

The L^AT_EX 2_ε Sources

Johannes Braams
David Carlisle
Alan Jeffrey
Leslie Lamport
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2027-07-01

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `latex`) at
<https://latex-project.org/bugs.html>.

Contents

01	ltdirchk.dtx	1
1	L^AT_EX System Dependent Initializations	1
2	Initialization	2
2.1	INITEX	3
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	5
3.2	UNIX (web2c)	6
3.3	UNIX (other)	7
3.4	MSDOS (emtex)	7
3.5	MSDOS (other)	7
3.6	VMS (DECUS T _E X, PD VMS 3.6)	7
3.7	VMS (???)	7
3.8	MACINTOSH (OzTeX 1.6)	8
3.9	MACINTOSH (other)	8
3.10	FAKE EXAMPLE	8
4	Setting \@currdir	9
5	Setting \input@path	10

6	Filename Parsing	11
7	T_EX Versions	13
8	ltxcheck.tex	13
02	lplain.dtx	14
1	Plain T_EX	14
03	ltvers.dtx	35
1	Version Identification	35
1.1	Declaring an all-new module	38
04	lualatex.dtx	40
1	Overview	40
2	Core T_EX functionality	40
3	Plain T_EX interface	41
4	Lua functionality	41
4.1	Allocators in Lua	41
4.2	Lua access to T _E X register numbers	42
4.3	Module utilities	43
4.4	Callback management	44
5	Implementation	45
5.1	Minimum LuaT _E X version	45
5.2	Older L ^A T _E X/Plain T _E X setup	45
5.2.1	Fixes to <code>etex.src/etex.sty</code>	45
5.2.2	luatex specific settings	46
5.3	Attributes	47
5.4	Category code tables	47
5.5	Named Lua functions	49
5.6	Custom whatsits	50
5.7	Lua bytecode registers	50
5.8	Lua chunk registers	50
5.9	Lua loader	51
5.10	Lua module preliminaries	52
5.11	Lua module utilities	53
5.11.1	Module tracking	53
5.11.2	Module messages	53
5.12	Accessing register numbers from Lua	54
5.13	Attribute allocation	55
5.14	Custom whatsit allocation	56

5.15	Bytecode register allocation	56
5.16	Lua chunk name allocation	57
5.17	Lua function allocation	57
5.18	Lua callback management	58
5.18.1	Housekeeping	58
5.18.2	Handlers	63
5.18.3	Public functions for callback management	65
05	ltxpl.dtx	72
1	expl3-dependent code	72
1.1	Loader	72
1.2	Using expl3 code	75
2	Document-level command names for expl3 functions	76
06	ltdfns.dtx	80
1	Definitions	80
1.1	Initex initializations	80
1.2	Saved versions of TeX primitives	80
1.3	Command definitions	81
1.4	Robust commands and protect	91
1.5	Acting on robust commands	98
1.5.1	Copying robust commands	100
1.5.2	Showing robust commands	103
1.5.3	Commands defined with <code>\DeclareRobustCommand</code>	103
1.5.4	Commands defined with <code>\newcommand</code> (with optional argument)	105
1.5.5	Showing environments	107
1.6	Internal defining commands	108
2	Discretionary Hyphenation	112
07	ltxcmd.dtx	115
1	Creating document commands	115
1.1	Overview	115
1.2	Describing argument types	115
1.3	Modifying argument descriptions	116
1.4	Creating document commands and environments	117
1.5	Optional arguments	118
1.6	Spacing and optional arguments	118
1.7	‘Embellishments’	119
1.8	Testing special values	119
1.9	Auto-converting to key–value format	121
1.10	Argument processors	122
1.11	Body of an environment	124
1.12	Fully-expandable document commands	125

1.13	Commands at the start of tabular cells	126
1.14	Using the verbatim argument types	126
1.15	Typesetting verbatim-like material	127
1.16	Verbatim environments	127
1.17	Performance	128
1.18	Details about argument delimiters	128
1.18.1	Character tokens	128
1.18.2	Control sequence tokens	128
1.19	Creating new argument processors	129
1.20	Variables and constants	130
1.21	Declaring commands and environments	133
1.22	Structure of <code>xparse</code> commands	139
1.23	Normalizing the argument specifications	143
1.24	Preparing the signature: general mechanism	152
1.25	Setting up a standard signature	153
1.26	Setting up expandable types	159
1.26.1	Copying a command and its internal structure	162
1.26.2	Showing the definition of a command	168
1.27	Grabbing arguments	173
1.28	Grabbing arguments expandably	191
1.29	Argument processors	196
1.30	Conversion to key–value form	198
1.31	Utilities	201
1.32	Access to the argument specification	205
1.33	Messages	205
1.34	User functions	211
08	lthooks.dtx	217
1	Introduction	217
2	Package writer interface	217
2.1	L ^A T _E X 2 _ε interfaces	217
2.1.1	Declaring hooks	217
2.1.2	Special declarations for generic hooks	218
2.1.3	Using hooks in code	219
2.1.4	Updating code for hooks	220
2.1.5	Hook names and default labels	223
2.1.6	The <code>top-level</code> label	225
2.1.7	Defining relations between hook code	225
2.1.8	Querying hooks	227
2.1.9	Displaying hook code	228
2.1.10	Debugging hook code	229
2.2	L3 programming layer (<code>expl3</code>) interfaces	229
2.3	On the order of hook code execution	232
2.4	The use of “reversed” hooks	234
2.5	Difference between “normal” and “one-time” hooks	235
2.6	Generic hooks provided by packages	235
2.7	Hooks with arguments	236

2.8	Private L ^A T _E X kernel hooks	238
2.9	Legacy L ^A T _E X 2 _ε interfaces	238
3	L^AT_EX 2_ε commands and environments augmented by hooks	239
3.1	Generic hooks	239
3.1.1	Generic hooks for all environments	240
3.1.2	Generic hooks for commands	241
3.1.3	Generic hooks provided by file loading operations	241
3.2	Hooks provided by <code>\begin{document}</code>	242
3.3	Hooks provided by <code>\end{document}</code>	242
3.4	Hooks provided by <code>\shipout</code> operations	244
3.5	Hooks provided for paragraphs	244
3.6	Hooks provided in NFSS commands	244
3.7	Hook provided by the mark mechanism	245
4	The Implementation	245
4.1	Debugging	245
4.2	Borrowing from internals of other kernel modules	246
4.3	Declarations	246
4.4	Providing new hooks	248
4.4.1	The data structures of a hook	248
4.4.2	On the existence of hooks	249
4.4.3	Setting hooks up	250
4.4.4	Disabling and providing hooks	256
4.5	Parsing a label	258
4.6	Adding or removing hook code	262
4.7	Setting rules for hooks code	283
4.8	Specifying code for next invocation	305
4.9	Using the hook	307
4.10	Querying a hook	313
4.11	Messages	317
4.12	L ^A T _E X 2 _ε package interface commands	320
4.13	Deprecated that needs cleanup at some point	323
4.14	Internal commands needed elsewhere	325
09	ltxcmdhooks.dtx	327
1	Introduction	327
2	Restrictions and operational details	328
2.1	Patching	329
2.1.1	Timing	329
2.2	Command copies	329
2.3	Grouping	330
2.4	Commands that look ahead	330
3	Package author interface	330
3.1	Arguments and redefining commands	331

4	The Implementation	332
4.1	Execution plan	332
4.2	Variables	333
4.3	Patching or delaying	334
4.4	Patching commands	336
4.4.1	Patching by expansion and redefinition	337
4.4.2	Patching by retokenization	345
4.5	Messages	351
10	ltsockets.dtx	353
1	Introduction	353
2	Configuration of the transformation process	353
2.1	The template mechanism	353
2.2	The hook mechanism	354
2.3	The socket mechanism	354
2.3.1	Examples	356
2.3.2	Details and semantics	357
2.4	Socket and plug names	359
2.4.1	Command syntax	359
2.4.2	Rationale for error handling	361
3	The Implementation	361
3.1	Debugging the socket structures	361
3.2	The L3 layer commands	362
3.3	Error messages	366
3.4	The L ^A T _E X 2 _ε interface commands	367
11	lttemplates.dtx	369
1	Introduction	369
2	What is a document?	369
3	Types, templates, and instances	370
4	Template types	370
5	Templates	370
6	Multiple choices	374
7	Instances	375
8	Document interface	376
9	Changing existing definitions	377
9.1	Expanding the values of keys	377

10	Creating equivalent keys	378
10.1	Modifying existing key values	378
11	Getting information about templates and instances	379
12	Debugging support	379
13	The implementation	379
13.1	Extensions for expl3	380
13.2	Variables and constants	381
13.3	Debugging support	383
13.4	Testing existence and validity	384
13.5	Saving and recovering property lists	386
13.6	Creating new template types	388
13.7	Design part of template declaration	388
13.8	Equivalent interfaces	392
13.9	Implementation part of template declaration	394
13.10	Editing template defaults	399
13.11	Creating instances of templates	403
13.12	Using templates directly	406
13.13	Assigning values to variables	407
13.14	Using instances	410
13.15	Assignment manipulation	411
13.16	Ad hoc key setting	411
13.17	Showing templates and instances	412
13.18	Messages	413
13.19	User functions	417
12	ltxalloc.dtx	421
1	Counters	421
13	ltxcntrl.dtx	423
1	Program control structure	423
14	ltxerror.dtx	427
1	Error handling and tracing	427
1.1	General commands	427
1.2	Specific errors	433
1.3	Tracing	437
15	ltxpar.dtx	438
1	Paragraphs	438
1.1	Implementation	438

16	ltpara.dtx	440
1	Introduction	440
1.1	The default processing done by the engine	440
2	The new mechanism implemented for L^AT_EX	442
2.1	The provided hooks	443
2.2	Altered and newly provided commands	444
2.3	Examples	445
2.3.1	Testing the mechanism	445
2.3.2	Mark the first paragraph of each <code>itemize</code>	447
2.4	Some technical notes	447
2.4.1	Glue items between paragraphs (found with <code>fancypar</code>)	447
3	The Implementation	448
3.1	Providing hooks for paragraphs	448
3.2	The error messages	456
17	ltmeta.dtx	457
1	Introduction	457
1.1	<code>\DocumentMetadata</code>	457
1.2	Storing and retrieving document properties	457
2	The Implementation	458
2.1	<code>\DocumentMetadata</code>	458
2.2	Document properties	459
2.3	Targets	460
18	ltspace.dtx	463
1	Spacing	463
1.1	User Commands	463
1.2	Chris' comments	463
1.3	Some immediate actions	465
1.4	The code	466
1.5	Vertical spacing	474
1.6	Horizontal space (and breaks)	479
19	ltlogos.dtx	484
1	Logos	484
20	ltfiles.dtx	485

1	File Handling	485
1.1	Safe Input Macros	499
1.2	Listing files	506
21	ltoutenc.dtx	509
1	Font encodings	509
1.1	Removing encoding-specific commands	511
1.2	The order of declarations	512
1.3	Docstrip modules	512
1.4	Definitions for the kernel	513
1.4.1	Declaration commands	513
1.4.2	Hyphenation	521
1.4.3	Miscellania	522
1.4.4	Default encodings	522
1.4.5	Math material	524
1.5	Definitions for the OT1 encoding	525
1.6	Definitions for the T1 encoding	528
1.7	Definitions for the OMS encoding	534
1.8	Definitions for the OML encoding	534
1.9	Definitions for the OT4 encoding	534
1.10	Definitions for the TS1 encoding	537
1.11	Definitions for the TU encoding	541
2	Package files	553
2.1	The fontenc package	553
22	ltcounts.dtx	556
1	Counters and Lengths	556
1.1	Document command and environment counter macros	556
23	ltnlength.dtx	569
1	Lengths	569
24	ltfssbas.dtx	571
1	Preliminary macros	571
2	Macros for setting up the tables	572
3	Selecting a new font	581
3.1	Macros for the user	581
3.2	Macros for loading fonts	587
4	Assigning math fonts to <i>versions</i>	594

25	ltfssaxes.dtx	601
1	Changing the font series	601
1.1	The series lookup table	601
1.2	Mapping rules for series changes	602
1.3	Changing to a new series	656
2	Changing the shape	660
2.1	Mapping rules for shape combinations	662
2.2	Changing to a new shape	666
3	Make sure we win ...	668
26	ltfssstrc.dtx	671
1	Introduction	671
2	A driver for this document	671
3	The Implementation	672
4	Handling Options	672
5	Macros common to fam.tex and tracefmt.sty	674
5.1	General font loading	674
5.2	Math fonts setup	680
5.2.1	Outline of algorithm for math font sizes	680
5.2.2	Code for math font size setting	681
5.2.3	Other code for math	682
6	Scaled font extraction	685
6.1	Sizefunctions	693
27	ltfsscnp.dtx	697
28	ltfssdcl.dtx	702
1	Interface Commands	702
29	ltfssini.dtx	735
1	NFSS Initialization	735
1.1	Providing math <i>versions</i>	735
2	Custom series settings for main document families	736
3	Supporting nested emphasis	755

4	Supporting \strongemph	758
4.1	Legacy	759
4.2	Miscellaneous	760
30	fontdef.dtx	766
1	Introduction	766
2	Customization	766
3	The docstrip modules	767
4	A driver for this document	767
5	The fonttext.ltx file	767
5.1	Encodings	768
5.2	Defaults	770
6	The fontmath.ltx file	772
6.1	The font encodings used	772
6.1.1	Symbolfont and Alphabet declarations	773
6.2	Math font sizes	773
6.3	The math symbol assignments	774
6.3.1	The letters	774
6.3.2	The digits	775
6.3.3	Punctuation, brace, etc. keys	775
6.3.4	Delimitercodes for characters	776
6.4	Symbols accessed via control sequences	776
6.4.1	Greek letters	776
6.4.2	Ordinary symbols	777
6.4.3	Large Operators	778
6.4.4	Binary symbols	778
6.4.5	Relations	779
6.4.6	Arrows	781
6.4.7	Punctuation symbols	781
6.4.8	Math accents	782
6.4.9	Radicals	782
6.4.10	Over and under something, etc	782
6.4.11	Delimiters	783
6.5	Math versions of text commands	784
6.6	Other special functions and parameters	785
6.6.1	Biggggg	785
6.6.2	The log-like functions	785
6.6.3	Parameters	785
7	Default cfg files	785
31	preload.dtx	787

1	Overview	787
2	Customization	787
3	Module switches for the DOCSTRIP program	787
4	A driver for this document	788
5	The code	788
32	lftntcmd.dtx	790
1	Introduction	790
2	The implementation	792
3	Initialization	798
33	lttextcomp.dtx	799
1	Sub-encodings	801
1.1	Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)	803
1.2	Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher	803
1.3	Unavailable in sub-encoding 3 and higher	806
1.4	Unavailable in sub-encoding 4 and higher	806
1.5	Unavailable in sub-encoding 5 (most older PS fonts) and higher	807
1.6	Unavailable in sub-encoding 6 and higher	807
1.7	Unavailable in sub-encoding 7 and higher	807
1.8	Unavailable in sub-encoding 8 and higher	807
1.9	Unavailable in Sub-encoding 9 (most missing)	808
2	Unicode engine specials	808
3	Font family sub-encodings setup	809
4	Legacy symbol support for lists and footnote symbols	813
5	The textcomp package	818
5.1	The old textcomp package code	819
5.1.1	Supporting oldstyle digits	828
5.1.2	Subset encoding defaults	828
6	The checkencodingssubset.tex file	830
34	ltpageno.dtx	839
1	Page Numbering	839

35	ltxref.dtx	840
1	Cross Referencing	840
1.1	Cross Referencing	840
36	ltproperties.dtx	849
1	Introduction	849
2	Design discussion	849
3	Handling unknown labels and properties	850
4	Rerun messages	850
5	Open points	850
6	Code interfaces	850
7	Auxiliary file interfaces	852
8	L^AT_EX 2_ε interface	853
9	Pre-declared properties	854
10	The Implementation	855
10.1	Reference commands	857
10.2	Tests and warnings	859
10.3	Predeclared properties	861
10.4	Messages	863
37	ltmiscen.dtx	864
1	Miscellaneous Environments	864
1.1	Environments	864
1.2	Center, Flushright, Flushleft	878
1.3	Verbatim	881
38	ltmath.dtx	889
1	Math setup	889
1.1	Math commands based on plain T _E X	889
1.1.1	The log-like functions	889
1.1.2	Biggggg	890
1.1.3	The UNSORTED Rest	890
1.2	Math Environments	897
1.3	External options to the standard document classes	904
1.3.1	Left equation numbering	904
1.3.2	Flush left equations	904

39	ltlists.dtx	907
1	List, and related environments	907
1.1	List and Trivlist	908
1.2	Vertical Spacing (skips)	909
1.3	Penalties	909
1.4	Horizontal Spacing (dimens)	909
1.5	Default Values	909
1.6	Itemize and Enumerate	922
40	ltboxes.dtx	925
1	L^AT_EX Box commands	925
1.1	Some low-level constructs	945
41	lftab.dtx	947
1	Tabbing, Tabular and Array Environments	947
1.1	tabbing	947
1.2	array and tabular environments	956
1.3	Hooks for tabulars	972
42	ltpictur.dtx	974
1	Picture Mode	974
1.1	Curves	1002
43	ltthm.dtx	1007
1	Theorem Environments	1007
44	ltsect.dtx	1012
1	Sectioning Commands	1012
1.1	The Title	1012
1.2	Sectioning	1013
1.2.1	Initializations	1020
1.3	Table of Contents etc.	1020
1.3.1	Convention	1020
1.3.2	Commands	1020
45	ltfloat.dtx	1026

1	Floats	1026
1.1	Floating Environments	1026
1.2	Footnotes	1040
46	ltxglo.dtx	1050
1	Index and Glossary Generation	1050
47	ltxbibl.dtx	1053
1	Bibliography Generation	1053
1.1	Default definitions	1057
48	ltxmarks.dtx	1058
1	Introduction	1058
2	Design-level and code-level interfaces	1059
2.1	Use cases for conditionals	1061
2.2	Understanding regions	1061
2.3	Debugging mark code	1063
3	Application examples	1063
4	Legacy L^AT_EX 2_ε interface	1063
4.1	Legacy design-level and document-level interfaces	1064
4.2	Legacy interface extensions	1064
5	Notes on the mechanism	1065
6	Public interfaces for packages such as <code>multicol</code>	1066
7	Internal functions for the standard output routine of L^AT_EX	1067
8	The Implementation	1068
8.1	Allocating new mark classes	1068
8.2	Updating mark structures	1070
8.3	Placing and retrieving marks	1078
8.4	Comparing mark values	1080
8.5	Messages	1081
8.6	Debugging the mark structures	1081
8.7	Designer-level interfaces	1083
9	L^AT_EX 2_ε integration	1084
9.1	Core L ^A T _E X 2 _ε integration	1084
9.2	Other L ^A T _E X 2 _ε output routines	1087
9.3	Rollback information	1088

49	ltpage.dtx	1089
1	Page styles and related commands	1089
1.1	Page Style Commands	1089
1.2	How a page style makes running heads and feet	1089
1.3	marking conventions	1089
50	ltclass.dtx	1094
1	Introduction	1094
2	User interface	1094
2.1	Option processing	1095
3	Class and Package interface	1096
3.1	Class name and version	1096
3.2	Package name and version	1096
3.3	Requiring other packages	1096
3.4	Declaring new options	1098
3.5	Safe Input Macros	1098
4	Implementation	1098
4.1	Hooks	1128
4.2	Providing shipment	1131
5	Package/class rollback mechanism	1139
6	After Preamble	1147
51	ltkeys.dtx	1149
1	Creating and using keyval options	1149
1.1	Implementation of ltkeys	1151
1.2	Key properties	1151
1.3	Main mechanism	1152
1.4	The document interfaces	1156
1.5	Option usage scope	1157
1.6	General key setting	1158
52	ltfilehook.dtx	1160

1	Introduction	1160
1.1	Provided hooks	1160
1.2	General hooks for file reading	1160
1.3	Hooks for package and class files	1161
1.4	Hooks for <code>\include</code> files	1162
1.5	High-level interfaces for <code>L^AT_EX</code>	1163
1.6	Kernel, class, and package interfaces for <code>L^AT_EX</code>	1164
1.7	A sample package for structuring the log output	1164
2	The Implementation	1165
2.1	Document and package-level commands	1165
2.2	<code>expl3</code> helpers	1166
2.3	Declaring the file-related hooks	1169
2.4	Patching <code>L^AT_EX</code> 's <code>\InputIfFileExists</code> command	1169
2.5	Declaring a file substitution	1171
2.6	Selecting a file (<code>\set@curr@file</code>)	1173
2.7	Replacing a file and detecting loops	1176
2.7.1	The Tortoise and Hare algorithm	1177
2.8	Preventing a package from loading	1179
2.9	High-level interfaces for <code>L^AT_EX</code>	1180
2.10	Internal commands needed elsewhere	1180
3	A sample package for structuring the log output	1181
4	Package emulations	1182
4.1	Package <code>atveryend</code> emulation	1182
53	ltshipout.dtx	1184
1	Introduction	1184
1.1	Overloading the <code>\shipout</code> primitive	1184
1.2	Provided hooks	1185
1.3	Legacy <code>L^AT_EX</code> commands	1187
1.4	Special commands for use inside the hooks	1188
1.5	Provided <code>LuaT_EX</code> callbacks	1188
1.6	Information counters	1189
1.7	Debugging shipout code	1189
2	Emulating commands from other packages	1190
2.1	Emulating <code>atbegshi</code>	1190
2.2	Emulating <code>everyshi</code>	1191
2.3	Emulating <code>atenddvi</code>	1191
2.4	Emulating <code>everypage</code>	1191
3	The Implementation	1192
3.1	Debugging	1193
3.2	Handling the end of job hook	1205
4	Legacy <code>L^AT_EX 2_ε</code> interfaces	1209

5	Internal commands needed elsewhere	1209
6	Package emulation for compatibility	1211
6.1	Package atendddvi emulation	1211
6.2	Package atbegshi emulation	1212
6.3	Package everyshi emulation	1213
54	ltoutput.dtx	1214
1	Output Routine and float handling	1214
1.1	Historical notes on the algorithm and commands	1214
1.2	Core definitions	1224
1.2.1	Definition of float boxes	1224
1.2.2	Page layout parameters	1225
1.2.3	Internal registers	1227
1.2.4	Page break commands	1227
2	The L^AT_EX output routine	1232
2.1	Hooks and replaceable code blocks	1232
2.1.1	Output routine hooks	1232
2.1.2	Replaceable code blocks (sockets)	1232
2.1.3	Tagging sockets	1234
2.1.4	Output routine commands	1235
2.2	The output routine configuration components	1246
2.2.1	Dealing with floats	1260
2.2.2	Kludgeins	1289
2.2.3	Float control	1291
2.2.4	Float placement parameters	1305
55	lttagging.dtx	1309
1	General support for tagged output	1309
2	Implementation	1311
2.1	Math collection	1312
2.2	Tagging sockets	1312
2.3	Generic sockets	1312
2.3.1	Tagging support for paragraph setup	1313
2.3.2	Tagging socket for targets	1317
2.3.3	Tagging Sockets for boxes	1317
2.3.4	Tagging Sockets for lists and blocks	1317
2.3.5	Tagging sockets for headings	1317
2.3.6	Tagging sockets for toc	1318
2.3.7	Tagging support for marginpar	1318
2.3.8	Tagging support for table/tabular packages	1318
2.3.9	Tagging Support for floats	1320
2.4	Tagging support for output routines	1320
2.5	Tagging support for math	1321
2.5.1	General sockets	1321

2.5.2	Sockets specific for luamml	1321
2.6	MathML intent attributes	1323
2.7	Symbolic structure names	1323
3	For lttab.dtx parked here for now	1323
3.1	Variables for row, column and span counting	1323
3.2	Tracing/debugging	1324
3.3	Interface commands	1325
56	lthyphen.dtx	1330
57	ltfinal.dtx	1332
1	Final settings	1332
1.1	Debugging	1332
1.2	Typesetting parameters	1332
1.3	Lccodes for hyphenation	1336
1.4	Hyphenation	1339
1.5	Font loading	1339
1.6	Input encoding	1342
1.7	Lccodes and uccodes	1346
1.8	Case changing	1347
1.9	Automatic insertion of \par tokens	1350
1.10	Applying Patch files	1350
1.11	Freeing Memory	1351
1.12	Initialize file list	1352
1.13	Initialize file list	1352
1.14	Preparation for supporting PDF in backends	1352
1.15	Do some temporary work for pre-release	1353
1.16	Some last minute initializations	1353
1.17	Dumping the format	1353
Change History		1357
Index		1445

File 01

ltdirchk.dtx

1 L^AT_EX System Dependent Initializations

This file implements the semi-automatic determination of various system dependent parts of the initialization. The actual definitions may be placed in a file `texsys.cfg`. Thus for operating systems for which the tests here do not result in acceptable settings, a ‘hand written’ `texsys.cfg` may be produced.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}<space>` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or `<space>`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, T_EX will try to load the expansion of `<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base` and `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS and Macintosh syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS T_EX versions. Currently if the UNIX, VMS or Macintosh parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` `\@TeXversion` is now set automatically by the initialization tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation is left for information. L^AT_EX does not set this variable exactly, the automatic tests set it to:

2 for any version, v , $v < 3.0$

3 for any version, v , $3.0 \leq v \leq 3.14$

`<undefined>` otherwise.

However these values are accurate enough for L^AT_EX to take appropriate action for these old T_EXs.

If your T_EX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this¹, L^AT_EX will not work around a bug in old T_EX versions, and so error messages will appear in a very strange format, with `^^J` appearing instead of line breaks:

```
LaTeX Error: \rubbish undefined.^^J^^JSee the LaTeX manual or LaTeX=
Companion
for explanation.^^JType  H <return>  for immediate help.
...

.3 \renewcommand{\rubbish}
      {}
```

However if you put `\def\@TeXversion{3.14}` in `texsys.cfg` the following format will be used:

```
LaTeX Error: \rubbish undefined.

ee the LaTeX manual or LaTeX Companion for explanation.
ype  H <return>  for immediate help.
.
...

.3 \renewcommand{\rubbish}
      {}
```

Note that this has an extra line ! . which does not appear in error messages that use the default settings with a current version of T_EX, but this should not cause any confusion we hope.

2 Initialization

As this file is read at a very early stage, some definitions that are normally considered to be part of the format must be made here.

Most such definitions are repeated later in the “right” place, usually (but not always) with different implementations. To be able to spot this more easily if you look into the file `latex.ltx` (which is stripped of comments) we add some comment lines to that effect that survive the stripping process by `docstrip`.

```
1 <*dircheck>
2 %% ---- START temporary definitions for bootstrapping; later overwritten ----
3 </dircheck>
```

¹Actually if your T_EX is really old, version 2, L^AT_EX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

2.1 INITEX

```

4 <*dircheck>
5 <*initex>
6 <initex>\ifnum\catcode'\{=1
7 <initex> \errmessage
8 <initex> {LaTeX must be made using an initex with no format preloaded}
9 <initex>\fi
10 \catcode'\{=1
11 \catcode'\}=2

```

If LuaTeX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of LuaTeX do not hide the primitives: a version check is not needed as the version itself will be missing in the case where action is needed!

```

12 \ifx\directlua\undefined
13 \else
14 \ifx\luatexversion\undefined

```

Enable e-TeX/pdfTeX/Umath primitives with their natural names

```

15 \directlua{tex.enableprimitives("",%
16 tex.extraprimitives('etex', 'pdftex', 'umath'))}

```

In current formats enable primitives with unprefix names. the `latexrelease` guards allow the primitives to be defined with a `\luatex` prefix if older formats are specified.

The unprefix forms are *not* undefined for improved compatibility with external packages when rolling back the format.

```

17 </initex>
18 </dircheck>
19 <*initex, latexrelease>
20 <latexrelease>\ifx\directlua\undefined\else
21 <latexrelease>\IncludeInRelease{2015/10/01}{\luatexluafunction}
22 <latexrelease> {LuaTeX (prefixed names)}%
23 \directlua{tex.enableprimitives("",%
24 tex.extraprimitives("omega", "aleph", "luatex"))}
25 <latexrelease>\EndIncludeInRelease
26 <latexrelease>\IncludeInRelease{0000/00/00}{\luatexluafunction}
27 <latexrelease> {LuaTeX (prefixed names)}%
28 <latexrelease>\directlua{
29 <latexrelease> tex.enableprimitives(
30 <latexrelease> "luatex",
31 <latexrelease> tex.extraprimitives("core","omega", "aleph", "luatex")
32 <latexrelease> )
33 <latexrelease>}
34 <latexrelease>\EndIncludeInRelease
35 <latexrelease>\fi
36 </initex, latexrelease>
37 <*dircheck>
38 <*initex>
39 \fi
40 \fi

```

A test can now be made for eTeX.

```

41 <initex>\ifx\eTeXversion\undefined
42 <initex> \errmessage
43 <initex> {LaTeX requires e-TeX}
44 <initex> \expandafter\endinput

```

```

45 <initex>\fi
    That distraction over, back to the basics of a format.
46 \catcode'\#=6
47 \catcode'\^=7
48 \chardef\active=13
49 \catcode'\@=11
50 \countdef\count@=255
51 \let\bgroup={ \let\egroup=}
52 \ifx\@input\undefined\let\@input\input\fi
53 \ifx\@end\undefined\let\@end\end\fi
54 \chardef\@inputcheck0
55 \chardef\sixt@n=16
56 \newlinechar'\^^J
57 \def\typeout{\immediate\write17}
58 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
59 \do\#\do\^{\do\_ \do\% \do\~ \do\^^I}
60 \def\@makeother#1{\catcode'#1=12\relax}
61 \def\space{ }
62 \def\@tempswafalse{\let\if@tempswa\iffalse}
63 \def\@tempswatrue{\let\if@tempswa\iftrue}
64 \let\if@tempswa\iffalse
65 \def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
66 \iterate \let\iterate\relax}
67 \let\repeat\fi
68 </initex>

```

2.2 Some bits of 2e

```

69 <*2kernel>
70 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
71 \long\def\@firstoftwo#1#2{#1}
72 \long\def\@secondoftwo#1#2{#2}

```

This is a special version of \ProvidesFile for initex use.

```

73 \def\ProvidesFile#1{%
74 \begingroup
75 \catcode'\ 10 %
76 \ifnum \endlinechar<256 %
77 \ifnum \endlinechar>\m@ne
78 \catcode\endlinechar 10 %
79 \fi
80 \fi
81 \@makeother\/%
82 \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]
83 \def\@providesfile#1[#2]{%
84 \wlog{File: #1 #2}%
85 \@addtofilelist{ #2}%
86 \endgroup}
87 \long\def\@addtofilelist#1{}
88 \def\@empty{}
89 \catcode'\%=12
90 \def\@percentchar{%}
91 \catcode'\%=14
92 \let\@currdir\@undefined
93 \let\input@path\@undefined

```

```

94 \let\filename@parse\@undefined

\strip@prefix

```

```

95 \def\strip@prefix#1>{}
96 </2ekernel>

```

(End of definition for \strip@prefix.)

3 texsys.cfg

As mentioned above, any site specific definitions required to describe the filename handling must be entered into a file `texsys.cfg`. If `texsys.cfg` can not be located by `\openin`, we write a default version out. The default version only contains comments, so we do not actually input the file in that case. The automatic tests later will, hopefully, correctly define the required macros.

The tricky code below checks to see if `texsys.cfg` exists. If it does not, all the text in this file between `START` and `END` is copied verbatim to a new file `texsys.cfg`. If `texsys.cfg` is found, then it is simply input. This is only done when this file is being used unstripped.

```

97 <docstrip>
98 \openin15=texsys.cfg
99 \ifeof15
100 \typeout{** Writing a default texsys.cfg}
101 \immediate\openout15=texsys.cfg
102 \begingroup
103 \catcode'\^M\active%
104 \let^M\par%
105 \def\reserved@a#1^M{%
106   \def\reserved@b{#1}%
107   \ifx\reserved@b\reserved@c\endgroup\else%
108     \immediate\write15{#1}%
109     \expandafter\reserved@a{fi}%
110   \def\reserved@d#1START^M{\let\do\makeother\dospecials\reserved@a}%
111   \catcode'\%=12
112   \def\reserved@c{END}
113   \reserved@d

```

START

3.1 texsys.cfg

This file contains the site specific definitions of the four macros

`\@currdir`, `\input@path`, `\filename@parse` and `\@TeXversion`.

As distributed it only contains comments, however this ‘empty’ file will work on many systems because of the automatic tests built into `ltdirchk.dtx`. You *are* allowed to edit this file to add definitions of these macros appropriate to your system.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}<space>` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or `<space>`. If

the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, TeX will try to load the expansion of

`<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`. One exception to this rule is that the input path should *always* contain the empty directory `{}` as this will allow ‘full pathnames’ to be used, and the ‘current directory’ to be searched.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base`, `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS TeX versions. Currently if the UNIX or VMS parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` You should not need to set this macro in `texsys.cfg`. LaTeX tests to set this automatically. See the comments in the opening section of `ltdirchk.dtx`.

The following sections give examples of definitions which might work on various systems. These are currently mainly untested as I only have access to a few systems, all of which do not need this file as the automatic tests work. All the code is commented out.

3.2 UNIX (web2c)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

114 `%\def\@currdir{./}`

115 `%\let\input@path\@undefined`

3.3 UNIX (other)

Apparently some commercial UNIX implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`.

```
116 % \def\currdir{./}
117 % \def\input@path{%
118 %   {/usr/local/lib/tex/inputs/distrib/}%
119 %   {/usr/local/lib/tex/inputs/contrib/}%
120 %   {/usr/local/lib/tex/inputs/local/}%
121 % }
```

3.4 MSDOS (emtex)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
122 % \def\currdir{./}
123 % \let\input@path\undefined
```

3.5 MSDOS (other)

Some PC implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`. This assumes the implementation uses UNIX style `/` as the directory separator.

```
124 % \def\currdir{./}
125 % \def\input@path{%
126 %   {c:/tex/inputs/distrib/}%
127 %   {c:/tex/inputs/contrib/}%
128 %   {c:/tex/inputs/local/}%
129 % }
```

3.6 VMS (DECUS T_EX, PD VMS 3.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
130 % \def\currdir{[] }
131 % \let\input@path\undefined
```

3.7 VMS (???)

Some VMS implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following:

```
132 % \def\currdir{[] }
133 % \def\input@path{%
134 %   {tex_inputs:}%
135 %   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
136 % }
```

3.8 MACINTOSH (OzTeX 1.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
137 % \def\@currdir{:}
138 % \let\input@path\@undefined
```

3.9 MACINTOSH (other)

Some Macintosh implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever folders are used on your machine): note that the directory names should end with `:`, and they should contain *no* spaces.

```
139 % \def\@currdir{:}
140 % \def\input@path{%
141 %   {Hard-Disk:Applications:TeX:TeX-inputs:}%
142 %   {Hard-Disk:Applications:TeX:My-inputs:}%
143 % }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name`. For maximum compatibility with macro sets, you want `name.ext` to be mapped to `<ext>name`, and `<area>name.ext` to be mapped to `<area.ext>name`. `\input` does this mapping automatically, but `\openin` does not, and does not look in the same places as `\input`. `<>name` is the desired ‘current directory’ syntax.

the following code would possibly work:

```
144 % \def\@dir#1#2 {%
145 %   \@d@r{#1}#2..\@nil}
146 % \def\@d@r#1#2.#3.#4\@nil{%
147 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
148 %
149 % \def\@currdir{\@dir{}}
150 % \def\input@path{%
151 %   {\@dir{area.one}}}%
152 %   {\@dir{area.two}}}%
153 % }
```

END

```
154 \immediate\closeout15
```

If `texsys.cfg` did exist, then input it.

```
155 \else
156 \typeout{** Using the existing texsys.cfg}
157 \closein15
158 \input texsys.cfg
159 \fi
160 </docstrip>
```

If the stripped version of this file is being used (in `latex2e.ltx`) then `texsys.cfg` should be there, so just input it.

```
161 <dircheck>\input texsys.cfg
```

4 Setting \@currdir

`\@currdir` This is a local definition of `\IfFileExists`. It tries to relocate `texsys.aux`. If it succeeds, then the `\@currdir` syntax has been determined. If all the tests fail then `\@currdir` will be set to `\@empty`, and `ltxcheck` will warn of this when it checks the format.

```
162 \begingroup
163 \count@ \time
164 \divide \count@ 60
165 \count2 = - \count@
166 \multiply \count2 60
167 \advance \count2 \time
```

The current date and time stamp.

```
168 \edef \today {%
\today 169 \the \year / \two@digits {\the \month} / \two@digits {\the \day} : %
170 \two@digits {\the \count@} : \two@digits {\the \count2}}
```

Create a file `texsys.aux` (hopefully in the current directory), then try to locate it again.

```
171 \immediate \openout15 = texsys.aux
172 \immediate \write15 {\today ^^J}
173 \immediate \closeout15 %
```

`#1` is the file to try, `#2` is what to do on success, `#3` on failure. Note that this definition is overwritten later on again!

```
174 \def \IfFileExists #1#2#3 {%
175 \openin \@inputcheck #1 %
176 \ifeof \@inputcheck
177 #3 \relax
178 \else
179 \read \@inputcheck to \reserved@a
180 \ifx \reserved@a \today
181 \typeout {#1 found} #2 \relax
182 \else
183 \typeout {BAD: old file \reserved@a (should be \today)} %
184 #3 \relax
185 \fi
186 \fi
187 \closein \@inputcheck}
188 \endlinechar = -1
```

If `\@currdir` has not been pre-defined in `texsys.cfg` then test for UNIX, VMS and Oz-TeX-Mac. syntax.

```
189 \ifx \@currdir \@undefined
190 \IfFileExists {./texsys.aux} {\gdef \@currdir {./}} %
191 {\IfFileExists {[]texsys.aux} {\gdef \@currdir {[]}} %
192 {\IfFileExists {:texsys.aux} {\gdef \@currdir {:}} {}}}
```

If it is still undefined at this point, all the above tests failed. Earlier versions interactively prompted for a definition at this point, but it seems impossible to reliably obtain information from users at this point in the installation. This version of the file produces

a format with no user-interaction. Later if the format is not suitable for the system, `texsys.cfg` may be edited and the format re-made.

```

193 \ifx\@currdir\undefined
194 \global\let\@currdir\empty
195 \typeout{^^J^^J%
196     !! No syntax for the current directory could be found^^J%
197 }%
198 \fi

```

Otherwise `\@currdir` was defined in `texsys.cfg`. In this case check that the syntax specified works on this system. (In case a complete L^AT_EX system has been copied from one system to another.) If the test fails, give up. The installer should remove or correct the offending `texsys.cfg` and try again.

```

199 \else
200 \IfFileExists{\@currdir texsys.aux}{}{%
201 \edef\reserved@a{\errhelp{%
202 texsys.cfg specifies the current directory syntax to be^^J%
203 \meaning\@currdir^^J%
204 but this does not work on this system.^^J%
205 Remove texsys.cfg and restart.}}\reserved@a
206 \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@@end}

```

The version of `\@currdir` in `texsys.cfg` looks OK.

```

207 \fi

208 \immediate\closeout15 %
209 \endgroup

210 \typeout{^^J^^J%
211     \noexpand\@currdir set to:
212     \expandafter\strip@prefix\meaning\@currdir.^^J%
213 }

```

(End of definition for \@currdir, \IfFileExists, and \today.)

Stop here if the file is being used unstripped.

```

214 <*docstrip>
215 \relax\endinput
216 </docstrip>

```

5 Setting `\input@path`

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable. The first-time installer of L^AT_EX 2_ε can not be expected to have enough information to supply the correct information to the prompts. Now the interaction is omitted. After the format is made the installer can attempt to run the test document `ltxcheck.tex` through L^AT_EX 2_ε. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

`\input@path` Now set up the `\input@path`.

`\input@path` should either be undefined, or a list of directories as described in the introduction.

```

217 \typeout{^^J%

```

```

218     Assuming \noexpand\openin and \noexpand\input^^J%
219     \ifx\input@path\@undefined
\input@path has not been pre-defined.
220     have the same search path.^^J%
221     \else
\input@path has been defined in texsys.cfg.
222     have different search paths.^^J%
223     LaTeX will use the path specified by \noexpand\input@path:^^J%
224     \fi
225     }

```

(End of definition for \input@path.)

6 Filename Parsing

\filename@parse Split a filename into its components.

```

226 \ifx\filename@parse\@undefined
227   \def\reserved@a{.}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like UNIX...
228   \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
229   \def\filename@parse#1{%
230     \let\filename@area\@empty
231     \expandafter\filename@path#1/\}

Search for the last /.

232   \def\filename@path#1/#2\{%
233     \ifx\#2\%
234       \def\reserved@a{\filename@simple#1.\}%
235     \else
236       \edef\filename@area{\filename@area#1/}%
237       \def\reserved@a{\filename@path#2\}%
238     \fi
239     \reserved@a}

240   \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like VMS...
241   \typeout{^^JDefining VMS style filename parser.^^J}
242   \def\filename@parse#1{%
243     \let\filename@area\@empty
244     \expandafter\filename@path#1]\}

Search for the last ].

245   \def\filename@path#1]\#2\{%
246     \ifx\#2\%
247       \def\reserved@a{\filename@simple#1.\}%
248     \else
249       \edef\filename@area{\filename@area#1]}\%
250       \def\reserved@a{\filename@path#2]\}%
251     \fi
252     \reserved@a}

253   \else\def\reserved@a{:}\ifx\@currdir\reserved@a

```

\filename@parse was not specified in texsys.cfg, but \@currdir looks like Macintosh...

```

254 \typeout{^^JDefining Mac style filename parser.^^J}
255 \def\filename@parse#1{%
256   \let\filename@area\@empty
257   \expandafter\filename@path#1:\}

```

Search for the last :.

```

258 \def\filename@path#1:#2\{%
259   \ifx\#2\%
260     \def\reserved@a{\filename@simple#1.\}%
261   \else
262     \edef\filename@area{\filename@area#1:}%
263     \def\reserved@a{\filename@path#2\}%
264   \fi
265   \reserved@a}
266 \else

```

\filename@parse was not specified in texsys.cfg. So just make a simple parser that always sets \filename@area to empty.

```

267 \typeout{^^JDefining generic filename parser.^^J}
268 \def\filename@parse#1{%
269   \let\filename@area\@empty
270   \expandafter\filename@simple#1.\}
271 \fi\fi\fi

```

\filename@simple is used by all three versions. Finally we can split off the extension.

```

272 </dircheck>
273 <*dircheck, latexrelease>
274 <latexrelease> \IncludeInRelease{2019/10/01}{\filename@simple}
275 <latexrelease> {Final dot for extension}%
276 \def\filename@simple#1.#2\{%
277   \ifx\#2\%
278     \let\filename@ext\relax
279     \edef\filename@base{#1}%
280   \else
281     \filename@dots{#1}#2\%
282   \fi}
283 \def\filename@dots#1#2.#3\{%
284   \ifx\#3\%
285     \def\filename@ext{#2}%
286     \edef\filename@base{#1}%
287   \else
288     \filename@dots{#1.#2}#3\%
289   \fi}
290 <latexrelease> \EndIncludeInRelease
291 <latexrelease> \IncludeInRelease{0000/00/00}{\filename@simple}
292 <latexrelease> {Final dot for extension}%
293 <latexrelease> \def\filename@simple#1.#2\{%
294 <latexrelease>   \ifx\#2\%
295 <latexrelease>     \let\filename@ext\relax
296 <latexrelease>   \else
297 <latexrelease>     \edef\filename@ext{\filename@dot#2\}%

```

```

298 <latexrelease>      \fi
299 <latexrelease>      \edef\filename@base{#1}}
300 <latexrelease>\EndIncludeInRelease
301 </dircheck,latexrelease>
302 <*dircheck>

      Remove a final dot, added earlier.
303   \def\filename@dot#1.\{#1}

304   \else

Otherwise, \filename@parse was specified in texsys.cfg.
305   \typeout{^^J^^J%
306     \noexpand\filename@parse was defined in texsys.cfg:^^J%
307     \expandafter\strip@prefix\meaning\filename@parse.^^J%
308   }
309   \fi

(End of definition for \filename@parse.)

```

7 T_EX Versions

`\@TeXversion` T_EX versions older than 3.141 require `\@TeXversion` to be set. This can be determined automatically due to a trick suggested by Bernd Raichle. Actually this will not always get the correct version number, e.g., T_EX3.14 would be detected as T_EX3, but L^AT_EX only needs to take account of T_EX's older than 3, or between 3 and 3.14.

```

310 \ifx\@TeXversion\@undefined
311   \ifx\@undefined\inputlineno
312     \def\@TeXversion{2}
313   \else
314     {\catcode'\^^J=\active
315      \def\reserved@a#1#2\@{\if#1\string^3\fi}
316      \edef\reserved@a{\expandafter\reserved@a\string^^J\@}
317      \ifx\reserved@a\@empty\else\gdef\@TeXversion{3}\fi}
318   \fi
319 \fi

(End of definition for \@TeXversion.)

320 %% ---- END temporary definitions for bootstrapping ----
321 </dircheck>

```

8 ltxcheck.tex

After the format has been made, and `article.cls` moved with the other files to the ‘standard input directory’ as specified in `install.txt`, the format may be checked by running the file `ltxcheck.tex`.

File 02

lplain.dtx

1 Plain T_EX

L^AT_EX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain T_EX format described in Appendix B of the T_EXBook. However, some of the user commands are not much use so, in order to save memory, we may remove them from the kernel into a package. Here is a list of the commands that may be removed (PROBABLY NOT COMPLETE).

```
\magstep      \magstephalf
\mathhexbox
\vglue        \vgl@
\hglue        \hgl@
```

This file is by now very small as most of it has been moved to more appropriate kernel files: it may disappear completely one day.

L^AT_EX font definitions are done using NFSS2 so none of PLAIN's font definitions are in L^AT_EX.

L^AT_EX has its own tabbing environment, so PLAIN's is disabled.

L^AT_EX uses its own output routine, so most of the plain one was removed.

```
1 (*2kernel)
2 \catcode'\{=1 % left brace is begin-group character
3 \catcode'\}=2 % right brace is end-group character
4 \catcode'\$=3 % dollar sign is math shift
5 \catcode'\&=4 % ampersand is alignment tab
6 \catcode'\#=6 % hash mark is macro parameter character
7 \catcode'\^=7 % circumflex and uparrow are for superscripts
8 \catcode'\_ =8 % underline and downarrow are for subscripts
9 \catcode'\^^I=10 % ascii tab is a blank space
10 \chardef\active=13 \catcode'\~=\active % tilde is active
11 \catcode'\^^L=\active \def^^L{\par}% ascii form-feed is \par
12 \message{catcodes,}
```

We had to define the \catcodes right away, before the message line, since \message uses the { and } characters. When INITEX (the T_EX initializer) starts up, it has defined the following \catcode values:

```
\catcode'\^^@=9 %  ascii null is ignored
\catcode'\^^M=5 %  ascii return is end-line
\catcode'\'=0 %    backslash is TeX escape character
\catcode'\%=14 %   percent sign is comment character
\catcode'\ =10 %   ascii space is blank space
\catcode'\^^?=15 %  ascii delete is invalid
\catcode'\A=11 ... \catcode'\Z=11 % uppercase letters
\catcode'\a=11 ... \catcode'\z=11 % lowercase letters
all others are type 12 (other)
```

Here is a list of the characters that have been specially catcoded:

```
13 \def\dospecials{\do\ \do\\\do\{\do\}\do\$ \do\&%
14 \do\# \do\^ \do\_ \do\% \do\~ \do\^^I}
```

(not counting ascii null, linefeed, formfeed, return, delete) Each symbol in the list is preceded by , which can be defined if you want to do something to every item in the list.

We make @ signs act like letters, temporarily, to avoid conflict between user names and internal control sequences of plain format.

```
15 \catcode'\@=11
```

To make the plain macros more efficient in time and space, several constant values are declared here as control sequences. If they were changed, anything could happen; so they are private symbols.

`\@ne` Small constants are defined using `\chardef`.

```
\tw@ 16 \chardef\@ne=1
\thr@@ 17 \chardef\tw@=2
\sixt@@n 18 \chardef\thr@@=3
\@cclv 19 \chardef\sixt@@n=16
        20 \chardef\@cclv=255
```

(End of definition for \@ne and others.)

`\@ccclvi` Constants above 255 defined using `\mathchardef`.

```
\@m 21 \mathchardef\@ccclvi=256
\@M 22 \mathchardef\@m=1000
\@MM 23 \mathchardef\@M=10000
      24 \mathchardef\@MM=20000
```

(End of definition for \@ccclvi and others.)

Allocation of registers

Here are macros for the automatic allocation of `\count`, `\box`, `\dimen`, `\skip`, `\muskip`, and `\toks` registers, as well as `\read` and `\write` stream numbers, `\fam` codes, `\language` codes, and `\insert` numbers.

```
25 \message{registers,}
```

When a register is used only temporarily, it need not be allocated; grouping can be used, making the value previously in the register return after the close of the group. The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more macro packages are being used at the same time.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

The following counters are reserved:

- 0 to 9 page numbering
- 10 count allocation
- 11 dimen allocation
- 12 skip allocation
- 13 muskip allocation
- 14 box allocation
- 15 toks allocation
- 16 read file allocation
- 17 write file allocation
- 18 math family allocation
- 19 language allocation
- 20 insert allocation
- 21 the most recently allocated number

22 constant -1

End of historical L^AT_EX 2.09 comments.

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, `\count 10` always contains the number of the highest-numbered counter that has been allocated, `\count 14` the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a `\count`, `\dimen`, `\skip`, and `\box` all with the same number; `\count 20` contains the lowest-numbered insert that has been allocated. Of course, `\box255` is reserved for `\output`; `\count255`, `\dimen255`, and `\skip255` can be used freely.

It is recommended that macro designers always use `\global` assignments with respect to registers numbered

1, 3, 5, 7, 9,

and always non-`\global` assignments with respect to registers

0, 2, 4, 6, 8, 255.

This will prevent “save stack buildup” that might otherwise occur.

```
26 \count10=22 % allocates \count registers 23, 24, ...
27 \count11=9 % allocates \dimen registers 10, 11, ...
28 \count12=9 % allocates \skip registers 10, 11, ...
29 \count13=9 % allocates \muskip registers 10, 11, ...
30 \count14=9 % allocates \box registers 10, 11, ...
31 \count15=9 % allocates \toks registers 10, 11, ...
32 \count16=-1 % allocates input streams 0, 1, ...
33 \count17=-1 % allocates output streams 0, 1, ...
34 \count18=3 % allocates math families 4, 5, ...
35 \count19=0 % allocates \language codes 1, 2, ...
36 \count20=255 % allocates insertions 254, 253, ...
```

`\insc@unt` The insertion counter and most recent allocation.

```
\allocationnumber
37 \countdef\insc@unt=20
38 \countdef\allocationnumber=21
```

(End of definition for \insc@unt and \allocationnumber.)

`\m@ne` The constant -1 .

```
39 \countdef\m@ne=22 \m@ne=-1
```

(End of definition for \m@ne.)

`\wlog` Write on log file (only)

```
40 \def\wlog{\immediate\write\m@ne}
```

(End of definition for \wlog.)

`\count@` Here are abbreviations for the names of scratch registers that don’t need to be allocated.

```
\dimen@
41 \countdef\count@=255
```

```
\dimen@i
42 \dimendef\dimen@=0
```

```
\dimen@ii
43 \dimendef\dimen@i=1 % global only
```

```
\skip@
44 \dimendef\dimen@ii=2
```

```
\toks@
45 \skipdef\skip@=0
```

```
46 \toksdef\toks@=0
```

(End of definition for \count@ and others.)

\newcount Now, we define \newcount, \newbox, etc. so that you can say \newcount\foo and \foo
 \newdimen will be defined (with \countdef) to be the next counter.
 \newskip To find out which counter \foo is, you can look at \allocationnumber.
 \newmuskip Since there's no \boxdef command, \chardef is used to define a \newbox,
 \newbox \newinsert, \newfam, and so on.
 \newtoks L^AT_EX change: remove \outer from \newcount and \newdimen (FMi) This is nec-
 \newread essary to use \newcount inside \if... later on. Also remove from \newskip, \newbox
 \newwrite \newwrite and \newfam (DPC) to save later redefinition. Rolling back the allocation
 \newfam routine runs us into trouble nowadays as the kernel itself will use up all of the standard
 \newlanguage pool. So we pretend we've always used the full range: this is needed for roll forward.

```

47 \def\newcount {\e@alloc\count \countdef {\count10}\insc@unt\float@count}
48 \def\newdimen {\e@alloc\dimen \dimendef {\count11}\insc@unt\float@count}
49 \def\newskip {\e@alloc\skip \skipdef {\count12}\insc@unt\float@count}
50 \def\newmuskip
51 {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}

```

For compatibility use \chardef in the classical range.

```

52 \def\newbox {\e@alloc\box
53 {\ifnum\allocationnumber<\@ccclvi
54 \expandafter\chardef
55 \else
56 \expandafter\e@alloc@chardef
57 \fi}
58 {\count14}\insc@unt\float@count}
59 \def\newtoks {\e@alloc\toks \toksdef{\count15}\m@ne\e@alloc@top}
60 \def\newread {\e@alloc\read \chardef{\count16}\m@ne\sixt@@n}

```

Skip \write18 due to its traditional use as a shell-escape.

```

61 \ifx\directlua@\undefined
62 \def\newwrite {\e@alloc\write \chardef{\count17}\m@ne\sixt@@n}
63 \else
64 \def\newwrite {\e@alloc\write
65 {\ifnum\allocationnumber=18
66 \advance\count17\@ne
67 \allocationnumber\count17 %
68 \fi
69 \global\chardef}%
70 {\count17}%
71 \m@ne
72 {128}}
73 \fi
74 \def\new@mathgroup
75 {\e@alloc\mathgroup\chardef{\count18}\m@ne\e@mathgroup@top}
76 \let\newfam\new@mathgroup
77 \ifx\directlua@\undefined
78 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne\@ccclvi}
79 \else
80 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne{16384}}
81 \fi

```

(End of definition for \newcount and others.)

`\e@alloc@chardef` The upper limit of extended registers, which leaves this number (eg `\dimen32767`) always unallocated by these macros. cf traditional `\dimen255`.

```

82 \ifx\directlua\undefined
83   \mathchardef\e@alloc@top=32767
84   \let\e@alloc@chardef\mathchardef
85 \else

```

luatex has 2^{16} registers.

```

86   \chardef\e@alloc@top=65535
87   \let\e@alloc@chardef\chardef
88 \fi

```

(End of definition for `\e@alloc@chardef` and `\e@alloc@top`.)

`\e@mathgroup@top` The upper limit of extended math groups (`\fam`) 16 in classic T_EX and e-T_EX, but 256 in Unicode TeX variants.

```

89 \ifx\Umathcode\undefined

```

classic and e tex have 16 fam (0–15).

```

90   \chardef\e@mathgroup@top=16
91 \else

```

xetex and luatex have 256 fam (0–255).

```

92   \chardef\e@mathgroup@top=256
93 \fi

```

(End of definition for `\e@mathgroup@top`.)

`\e@alloc` A modified version of `\alloc@` that takes the count register rather than just the final digit of its number (assuming `\count1x`). It also has an extra argument to give the top of the extended range.

```

      #1 #2      #3    #4    #5      #6
      \e@alloc type defcmd current top extended-top newname

```

Note that if just a single allocation range is required (not omitting a range up to 255 for inserts) then `-1` should be used for the first upper bound argument, `#4`.

```

94 \def\e@alloc#1#2#3#4#5#6{%
95   \global\advance#3\@ne
96   \e@ch@ck{#3}{#4}{#5}#1%
97   \allocationnumber#3\relax
98   \global#2#6\allocationnumber
99   \wlog{\string#6=\string#1\the\allocationnumber}}%

```

(End of definition for `\e@alloc`.)

`\e@ch@ck` Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

Allocate matching registers from the top of the extended range and add to `\@freelist`.

```

\extrafloats
100 \gdef\e@ch@ck#1#2#3#4{%
101   \ifnum#1<#2\else

```

If we've reached the classical top limit, bump to 256 or 266 for counts (count 256–265 are reserved by the allocation system).

```

102 \ifnum#1=#2\relax
103 \global#1@cclvi
104 \ifx\count#4\global\advance#1 10 \fi
105 \fi

```

Check we are below the extended limit.

```

106 \ifnum#1<#3\relax
107 \else
108 \errmessage{No room for a new \string#4}%
109 \fi
110 \fi}%
111 \let\float@count\e@alloc@top

```

`\extrafloats`

```

112 \ifx\numexpr\@undefined
In classic TeX use \newinsert to allocate float boxes.
113 \def\extrafloats#1{%
114 \count@#1\relax
115 \ifnum\count@>\z@
116 \newinsert\reserved@a
117 \global\expandafter\chardef
118 \csname bx@\the\allocationnumber\endcsname\allocationnumber
119 \@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
120 \advance\count@\m@ne
121 \expandafter\extrafloats
122 \expandafter\count@
123 \fi
124 }%
125 \else

```

In e-text take float boxes from the top of the extended range.

```

126 \def\extrafloats#1{%
127 \ifnum#1>\z@
128 \count@\numexpr\float@count-1\relax
129 \ifnum\count@<266 \ch@ck0@m@ne\insert\fi
130 \ch@ck0\count@\count
131 \ch@ck1\count@\dimen
132 \ch@ck2\count@\skip
133 \ch@ck4\count@\box
134 \global\e@alloc@chardef\float@count\count@
135 \global\expandafter\e@alloc@chardef
136 \csname bx@\the\float@count\endcsname\float@count
137 \@cons\@freelist{\csname bx@\the\float@count\endcsname}%
138 \expandafter\extrafloats\expandafter{\the\numexpr#1-1\expandafter}%
139 \fi}%
140 \fi

```

(End of definition for \e@ch@ck, \extrafloats, and \extrafloats.)

`\alloc@` Since `\e@alloc` was added in 2015, `\@alloc` has not been used, but was left as some legacy code calls it. However the original definition gives spurious errors once the “classic” registers run out, so it is now defined to call `\e@alloc` internally.

```
141 \def\alloc@#1#2#3#4{\e@alloc#2#3{\count1#1}#4\float@count}
```

(End of definition for \alloc@.)

`\newinsert` The highest register allowed with `\insert`.

```
142 \ifx\directlua\@undefined
143   \chardef\e@insert@top255
144 \else
145   \chardef\e@insert@top\e@alloc@top
146 \fi
```

If the classic registers are exhausted, take an insert from the free float list and use `\extrafloats` to add a new float to that list.

```
147 \def\newinsert#1{%
148   \@tempswafalse
149   \global\advance\insc@unt\m@ne
150   \ifnum\count10<\insc@unt
151   \ifnum\count11<\insc@unt
152   \ifnum\count12<\insc@unt
153   \ifnum\count14<\insc@unt
154     \@tempswatrue
155   \fi\fi\fi\fi
156   \if@tempswa
157     \allocationnumber\insc@unt
158   \else
159     \global\advance\insc@unt\@ne
160     \extrafloats\@ne
161     \@next\@currbox\@freelist
162     {\ifnum\@currbox<\e@insert@top
163       \allocationnumber\@currbox
164     \else
165       \ch@ck0\m@ne\insert
166     \fi}%
167     {\ch@ck0\m@ne\insert}%
168   \fi
169   \global\chardef#1\allocationnumber
170   \wlog{\string#1=\string\insert\the\allocationnumber}%
171 }
```

(End of definition for \newinsert.)

`\ch@ck`

```
172 \gdef\ch@ck#1#2#3{%
173   \ifnum\count1#1<#2\else
174     \errmessage{No room for a new #3}%
175   \fi}
```

(End of definition for \ch@ck.)

`\newhelp`

```
176 \def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
```

(End of definition for `\newhelp`.)

`\@inputcheck` Allocate read stream for testing and output stream that is never open and thus writes to
`\@unused` the terminal.

```

177 \newread\@inputcheck
178 \newwrite\@unused

```

(End of definition for `\@inputcheck` and `\@unused`.)

`\maxdimen` Here are some examples of allocation.

`\hideskip`

```

179 \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
180 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow

```

(End of definition for `\maxdimen` and `\hideskip`.)

`\p@`

`\z@`

```

181 \newdimen\p@ \p@=1pt % this saves macro space and time

```

`\z@skip`

```

182 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0

```

`\voidb@x`

```

183 \newskip\z@skip \z@skip=0pt plus 0pt minus 0pt
184 \newbox\voidb@x % permanently void box register

```

(End of definition for `\p@` and others.)

Assign initial values to T_EX's parameters

```

185 \message{parameters,}

```

All of T_EX's numeric parameters are listed here, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted.

```

186 \pretolerance=100
187 \tolerance=200 % INITEX sets this to 10000
188 \hbadness=1000
189 \vbadness=1000
190 \linepenalty=10
191 \hyphenpenalty=50
192 \exhyphenpenalty=50
193 \binoppenalty=700
194 \relpenalty=500
195 \clubpenalty=150
196 \widowpenalty=150
197 \displaywidowpenalty=50
198 \brokenpenalty=100
199 \predisplaypenalty=10000
200 % \postdisplaypenalty=0
201 % \interlinepenalty=0
202 % \floatingpenalty=0, set during \insert
203 % \outputpenalty=0, set before TeX enters \output
204 \doublehyphendemerits=10000
205 \finalhyphendemerits=5000
206 \adjdemerits=10000

```



```

207 % \looseness=0, cleared by TeX after each paragraph
208 % \pausing=0
209 % \holdinginserts=0
210 % \tracingonline=0
211 % \tracingmacros=0
212 % \tracingstats=0
213 % \tracingparagraphs=0
214 % \tracingpages=0
215 % \tracingoutput=0

```

In the past L^AT_EX used the default value of 1 for `\tracinglostchars` because this was the best it could do. This way one would at least get a warning in the `.log` file. e-T_EX improved on that and supported a value of 2 to show the warning on the terminal, so we could have changed the default when we made the e-T_EX extensions required—however, we overlooked that opportunity. In 2021 this parameter was improved on again and now also accepts the value 3 (error on the terminal). This made us realize that we should change the default. Using 3 would really be the best, but for compatibility reasons we only use 2. For recent LuaT_EX releases, we set 4: this is offset by 2 from the e-T_EX value as otherwise LuaT_EX only issues a warning for text in a box when the box is used, not saved. (Accepting values of 4 and 5 was updated in the T_EX Live 2025 release.)

```

216 \tracinglostchars=2
217 \ifdefined\luatexversion
218   \ifnum\luatexversion>120
219     \tracinglostchars=4
220   \fi
221 \fi
222 % \tracingcommands=0
223 % \tracingrestores=0

```

`\tracingstacklevels` For LuaT_EX, the `\tracingstacklevels` functionality was implemented as a callback, so here we just define the count register to hold the value of the parameter.

```

224 </2ekernel>
225 <*2ekernel | latexrelease>
226 <latexrelease> \IncludeInRelease{2021/06/01}{\tracingstacklevels}%
227 <latexrelease> {tracingstacklevels}%
228 \ifx\directlua\@undefined
229   % \tracingstacklevels=0 % added in 2021
230 \else
231   \newcount\tracingstacklevels
232   % Code for \tracingstacklevels defined in ltfinal.dtx
233 \fi
234 <latexrelease> \EndIncludeInRelease
235 <latexrelease>
236 <latexrelease> \IncludeInRelease{0000/00/00}{\tracingstacklevels}%
237 <latexrelease> {tracingstacklevels}%
238 <latexrelease> \ifx\directlua\@undefined
239 <latexrelease> \else
240 <latexrelease> \let\tracingstacklevels\@undefined
241 <latexrelease> \fi
242 <latexrelease> \EndIncludeInRelease
243 </2ekernel | latexrelease>
244 <*2ekernel>

```

(End of definition for \tracingstacklevels.)

```
245 \uchyph=1
246 % \lefthyphenmin=2 \righthyphenmin=3 set below
247 % \globaldefs=0
248 % \maxdeadcycles=25 % INITEX does this
249 % \hangafter=1 % INITEX does this, also TeX after each paragraph
250 % \fam=0
251 % \mag=1000 % INITEX does this
252 % \escapechar='\ % INITEX does this
253 \defaultthyphenchar='\-
254 \defaultskewchar=-1
255 % \endlinechar='\^M % INITEX does this
256 % \newlinechar=-1 \LaTeX\ sets this in ltdefs.dtx.
257 \delimiterfactor=901
258 % \time=now % TeX does this at beginning of job
259 % \day=now % TeX does this at beginning of job
260 % \month=now % TeX does this at beginning of job
261 % \year=now % TeX does this at beginning of job
```

In L^AT_EX we don't want box information in the transcript unless we do a full tracing.

```
262 \showboxbreadth=-1
263 \showboxdepth=-1
264 \errorcontextlines=-1
265 \hfuzz=0.1pt
266 \vfuzz=0.1pt
267 \overfullrule=5pt
268 \maxdepth=4pt
269 \splitmaxdepth=\maxdimen
270 \boxmaxdepth=\maxdimen
271 % \lineskiplimit=0pt, changed by \normalbaselines
272 \delimitershortfall=5pt
273 \nulldelimiterspace=1.2pt
274 \scriptspace=0.5pt
275 % \mathsurround=0pt
276 % \predisplaysize=0pt, set before TeX enters $$
277 % \displaywidth=0pt, set before TeX enters $$
278 % \displayindent=0pt, set before TeX enters $$
279 \parindent=20pt
280 % \hangindent=0pt, zeroed by TeX after each paragraph
281 % \hoffset=0pt
282 % \voffset=0pt
283 %
284 % \baselineskip=0pt, changed by \normalbaselines
285 % \lineskip=0pt, changed by \normalbaselines
286 \parskip=0pt plus 1pt
287 \abovedisplayskip=12pt plus 3pt minus 9pt
288 \abovedisplayshortskip=0pt plus 3pt
289 \belowdisplayskip=12pt plus 3pt minus 9pt
290 \belowdisplayshortskip=7pt plus 3pt minus 4pt
```

```

291 % \leftskip=0pt
292 % \rightskip=0pt

293 \topskip=10pt
294 \splittopskip=10pt

295 % \tabskip=0pt
296 % \spaceskip=0pt
297 % \xspaceskip=0pt

298 \parfillskip=0pt plus 1fil

\normalbaselineskip We also define special registers that function like parameters:
\normallineskip 299 \newskip\normalbaselineskip \normalbaselineskip=12pt
\normallineskiplimit 300 \newskip\normallineskip \normallineskip=1pt
301 \newdimen\normallineskiplimit \normallineskiplimit=0pt

(End of definition for \normalbaselineskip, \normallineskip, and \normallineskiplimit.)

\interfootlinepenalty
302 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100

(End of definition for \interfootlinepenalty.)
Definitions for preloaded fonts

\magstephalf
\magstep 303 \def\magstephalf{1095 }
304 \def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or
305 2074\or 2488\fi\relax}

(End of definition for \magstephalf and \magstep.)
Macros for setting ordinary text

\frenchspacing
\nonfrenchspacing 306 \def\frenchspacing{\sfcode'\.\@m \sfcode'\?\@m \sfcode'\!\@m
307 \sfcode'\:\@m \sfcode'\;\@m \sfcode'\,\@m}
308 \def\nonfrenchspacing{\sfcode'\.3000\sfcode'\?3000\sfcode'\!3000%
309 \sfcode'\:2000\sfcode'\;1500\sfcode'\,1250 }

(End of definition for \frenchspacing and \nonfrenchspacing.)

\normalbaselines
310 \def\normalbaselines{\lineskip\normallineskip
311 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}

(End of definition for \normalbaselines.)

\M Save a bit of space by using \let here.
\I 312 \def\^M{\ } % control <return> = control <space>
313 \let\^I\^M % same for <tab>

(End of definition for \M and \I.)

\lq
\rq 314 \def\lq{' }
315 \def\rq{' }

```

(End of definition for \lq and \rq.)

\lbrack
\rbra_{ck} ₃₁₆ \def\lbrack{[}
₃₁₇ \def\rbra_{ck}{]}

(End of definition for \lbrack and \rbra_{ck}.)

\aa These are not from plain.tex but they are similar to other commands found here and
\AA nowhere else, being alternate input forms for characters.

₃₁₈ \def \aa {\r a}
₃₁₉ \def \AA {\r A}

(End of definition for \aa and \AA.)

\endgraf
\endline ₃₂₀ \let\endgraf=\par
₃₂₁ \let\endline=\cr

(End of definition for \endgraf and \endline. These functions are documented on page [444](#).)

\space
₃₂₂ \def\space{ }

(End of definition for \space.)

\empty This probably ought to go altogether, but let it to the L^AT_EX version to save space.

₃₂₃ \let\empty\@empty

(End of definition for \empty.)

\null
₃₂₄ \def\null{\hbox{}}}

(End of definition for \null.)

\bgroup
\egroup ₃₂₅ \let\bgroup={
₃₂₆ \let\egroup=}

(End of definition for \bgroup and \egroup.)

\obeylines In \obeylines, we say \let^{^^}M=\obeyedline instead of \def^{^^}M{\obeyedline} since
\obeyspaces this allows, for example, \let\obeyedline=\cr \obeylines \halign{...
This is essentially a plain T_EX trick and in its original version where you had to use
to use \let\par=\cr not really a safe idea in L^AT_EX. If anybody used this trick this now
breaks (and one needs to use \obeyedline instead).

₃₂₇ </2ekernel>
₃₂₈ <*2ekernel | latexrelease>
₃₂₉ <[latexrelease]>\IncludeInRelease{2022/06/01}{\obeylines}%
₃₃₀ <[latexrelease]> {Add a redirection to obeylines and obeyspaces}%

If the active `^^M` escapes, e.g., into a `\write` (which is effectively in a different context) then we don't want the definition from `\obeylines` but rather a simple `\par` (in fact even the primitive one, not the L^AT_EX version `\para_end`: which is only defined later).

```
331 \begingroup
332 \catcode'\^^M=\active % these lines must end with %
333 \gdef\obeylines{\catcode'\^^M\active%
334 \let^^M\obeyedline%
```

The next line ending the definition is rather curious and it took me awhile to understand why rollback fails. The problem is the following: if `latexrelease` is used, then blocks of `\IncludeInRelease ... \EndIncludeInRelease` are bypassed at high speed by grabbing each as a delimited argument. However, in that case `^^M` is seen not as code but as line ending characters and in that mode T_EX discards everything from that point onwards to the real end of the line so it works like a comment — pretty strange really (and I think due to the fact that the original pascal compiler could have some garbage showing up after the normal line ending character. Thus we really have to make sure that any closing braces is not one the same line as an `^^M`, because otherwise it would get dropped and we end with unbalanced braces and never see the `\EndIncludeInRelease` — weird. In other places it doesn't matter because we aren't using the incomplete result.

```
335 }%
336 \global\let^^M\par % this is in case ^^M appears in a \write
337 \endgroup
```

`\obeyedline` The `\obeyedline` expands by default to `\par` with whatever definition `\par` has when it is executed. It can, however, be redefined (before calling `\obeylines`!) to achieve some special effects. If you want to alter this definition when already in the scope of `\obeylines`, it has no effect (because `\let` is used above). In that case simply make another call to `\obeylines` immediately. As you are in a restricted scope all that happens is that your redefinition is applied.

For the default definition we have to use `\def` not `\let` because the meaning of `\par` can change and we want to use the one that is current when `\obeylines` act.

There is a small subtlety here: in an `\edef` the active `^^M` stayed put (because it was equal to the primitive `\par`), now `\obeyedline` expands and you get what it contains, i.e., in that case `\par`, into the `\edef` or `\mark` unless we use `\protected` on it.

```
338 \protected\gdef\obeyedline{\par}
```

The definition of `\obeyspaces` is changed in the same way and now executes `\obeyedspace` for each active space.

```
\obeyedspace 339 \global\let\obeyedspace\space

340 \begingroup
341 \catcode'\ =\active%
342 \gdef\obeyspaces{\catcode'\ \active\let =\obeyedspace}%
```

An active space elsewhere generates `\space` by default (for example in a `\write`).

```
343 \global\let =\space%
344 \endgroup

345 </2ekernel | latexrelease>
346 <latexrelease>\EndIncludeInRelease
347 <latexrelease>\IncludeInRelease{0000/00/00}{\obeylines}%
348 <latexrelease> {Add a redirection to obeylines and obeyspaces}%
349 <latexrelease>
```

From 2019 onwards the commands are made robust (somewhat later in the kernel sources). So if we roll back they are robust, so when redefining them we have get rid of the robust payload first. Otherwise that is seen by the later rollback below, which then installs a fragile version of the new definition on top of the one we roll back to here, sigh. `\kernel@make@fragile` also changes its definition (later own) so this is done directly.

```
350 <latexrelease>\expandafter\let\csname obeylines \endcsname \@undefined
351 <latexrelease>\expandafter\let\csname obeyspace \endcsname \@undefined
352 <latexrelease>
353 <latexrelease>\begingroup
354 <latexrelease>\catcode'\^M=\active % these lines must end with %
355 <latexrelease> \gdef\obeylines{\catcode'\^M\active \let^M\par %
```

Closing brace on a separate line (see comment above).

```
356 <latexrelease> }%
```

Another pitfall: if we do a rollback `\par` is no longer the primitive, so the roll back definition needs `\let` to what is new the primitive.

```
357 <latexrelease> \global\let^M\RawParEnd % this is in case ^M appears in a \write
358 <latexrelease>\endgroup
359 <latexrelease>\def\obeyspaces{\catcode'\ \active}
360 <latexrelease>
361 <latexrelease>\let\obeyedline\@undefined
362 <latexrelease>\let\obeyedspace\@undefined
363 <latexrelease>\EndIncludeInRelease
364 <*2ekernel>
```

(End of definition for \obeylines and others.)

`\loop` We use Kabelschacht's method of doing loops, see TUB 8#2 (1987). (unless that breaks something :-). It turned out to need an extra `\relax`: see pr/642 (`\loop` could do one iteration too much in certain cases).

```
365 \long\def \loop #1\repeat{%
366   \def\iterate{#1\relax % Extra \relax
367     \expandafter\iterate\fi
368   }%
369   \iterate
370   \let\iterate\relax
371 }
```

This setting of `\repeat` is needed to make `\loop...\if...\repeat` skippable within another `\if...`

```
372 \let\repeat=\fi
```

(End of definition for \loop, \iterate, and \repeat.)

L^AT_EX defines `\smallskip`, etc. in `ltspace.dtx`.

```
\nointerlineskip
\offinterlineskip
373 \def\nointerlineskip{\prevdepth-\@m\p@}
374 \def\offinterlineskip{\baselineskip-\@m\p@
375   \lineskip\z@ \lineskiplimit\maxdimen}
```

(End of definition for \nointerlineskip and \offinterlineskip.)

```

\vglue
\hglue 376 \def\vglue{\afterassignment\vgl@\skip@=}
377 \def\vgl@{\par \dimen@\prevdepth \hrule \@height\z@
378 \nobreak\vskip\skip@ \prevdepth\dimen@}
379 \def\hglue{\afterassignment\hgl@\skip@=}
380 \def\hgl@{\leavevmode \count@\spacefactor \vrule \@width\z@
381 \nobreak\hskip\skip@ \spacefactor\count@}

(End of definition for \vglue and \hglue.)
LATEX defines ~ in ltdefs.dtx.

\slash This generates a / acting a bit like - but still allows hyphenation in the word part
preceding it (but not after).
382 \def\slash{/\penalty\exhyphenpenalty}

(End of definition for \slash.)

\break
\nobreak 383 \def\break{\penalty-\@M}
\allowbreak 384 \def\nobreak{\penalty \@M}
385 \def\allowbreak{\penalty \z@}

(End of definition for \break, \nobreak, and \allowbreak.)

\filbreak
\goodbreak 386 \def\filbreak{\par\vfil\penalty-200\vfilneg}
387 \def\goodbreak{\par\penalty-500 }

(End of definition for \filbreak and \goodbreak.)

\eject Define \eject as in plain TEX but define \supereject only in the compatibility file.
388 \def\eject{\par\break}

(End of definition for \eject.)

\removelastskip
389 \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}

(End of definition for \removelastskip.)

\smallbreak
\medbreak 390 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
\removelastskip\penalty-50\smallskip\fi}
\bigbreak 391 \def\medbreak{\par\ifdim\lastskip<\medskipamount
\removelastskip\penalty-100\medskip\fi}
392 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
\removelastskip\penalty-200\bigskip\fi}
393
394
395

(End of definition for \smallbreak, \medbreak, and \bigbreak.)

\m@th
396 \def\m@th{\mathsurround\z@}

(End of definition for \m@th.)

```

`\underbar` Due to L^AT_EX's redefinition of `\underline` plain T_EX's `\underbar` can be done in a simpler fashion (but do we need it at all?).

```

397 \def\underbar#1{\underline{\sbox\tw@{#1}\dp\tw@z@{\box\tw@}}}

```

(End of definition for `\underbar`.)

`\strutbox` L^AT_EX sets `\strutbox` in `\set@fontsize`.

```

\strut
398 \newbox\strutbox
399 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}

```

(End of definition for `\strutbox` and `\strut`.)

`\hidewidth` For alignment entries that can stick out.

```

400 \def\hidewidth{\hskip\hideskip}

```

(End of definition for `\hidewidth`.)

`\narrower`

```

401 \def\narrower{%
402   \advance\leftskip\parindent
403   \advance\rightskip\parindent}

```

(End of definition for `\narrower`.)

L^AT_EX defines `\ae` and similar commands elsewhere.

```

404 \chardef\%=' \%
405 \chardef\&=' \&
406 \chardef\#=' \#

```

Most text commands are actually encoding specific and therefore defined later, so commented out or removed from this file.

`\leavevmode` begins a paragraph, if necessary

```

407 \def\leavevmode{\unhbox\voidb@x}

```

(End of definition for `\leavevmode`.)

`\mathhexbox`

```

408 \def\mathhexbox#1#2#3{\mbox{$\m@th \mathchar"#1#2#3$}}

```

(End of definition for `\mathhexbox`.)

`\ialign`

```

409 \def\ialign{\everycr{}\tabskipz@skip\halign} % initialized \halign

```

(End of definition for `\ialign`.)

`\oalign`

```

\o@lign
410 \def\oalign#1{\leavevmode\vtop{\baselineskipz@skip \lineskip.25ex%
\oalign
411   \ialign{##\crrc#1\crrc}}}
412 \def\o@lign{\lineskiplimitz@ \oalign}
413 \def\oalign{\lineskiplimit-\maxdimen \oalign}

```

(End of definition for `\oalign`, `\o@lign`, and `\oalign`.)

`\sh@ft` The definition of this macro in plain.tex was improved in about 1997; but as a result its usage was changed and its new definition is not appropriate for L^AT_EX.

Since the version given here has been in use by L^AT_EX for many years it does not seem prudent to remove it now. As far as we can tell it has only been used to define `\b` and `\d` but this cannot be certain.

```
414 \def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font
415 \kern-.0156\dimen@} % compensate for slant in lowered accents
```

(End of definition for \sh@ft.)

`\ltx@sh@ft` This is the L^AT_EX version of the second incarnation of the plain macro `\sh@ft`, which takes a dimension as its argument. It shifts a pseudo-accent horizontally by an amount proportional to the product of its argument and the slant-per-point (`fontdimen 1`).

```
416 \def\ltx@sh@ft #1{%
417 \dimen@ #1%
418 \kern \strip@pt
419 \fontdimen1\font \dimen@
420 } % kern by #1 times the current slant
```

(End of definition for \ltx@sh@ft.)

L^AT_EX change: the text commands such as `\d`, `\b`, `\c`, `\copyright`, `\TeX` are now defined elsewhere.

L^AT_EX change: Make `\t` work in a moving argument. Now defined elsewhere.

`\hrulefill` L^AT_EX change: `\kern\z@` added to end of `\hrulefill` and `\dotfill` to make them work in ‘tabular’ and ‘array’ environments. (Change made 24 July 1987). L^AT_EX change: `\leavevmode` added at beginning of `\dotfill` and `\hrulefill` so that they work as expected in vertical mode.

```
421 \def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

The box in `\dotfill` originally contained (in plain.tex):

```
\mkern 1.5mu .\mkern 1.5mu;
```

the width of .44em differs from this by .04pt which is probably an acceptable difference within leaders.

```
422 \def\dotfill{%
423 \leavevmode
424 \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
425 \kern\z@}
```

(End of definition for \hrulefill and \dotfill.)

INITEX sets `\sfcode x=1000` for all x, except that `\sfcode X=999` for uppercase letters. The following changes are needed:

```
426 \sfcode'\)=0 \sfcode'\ '=0 \sfcode'\]=0
```

The `\nonfrenchspacing` macro will make further changes to `\sfcode` values.

Definitions related to output

`\magnification` doesn’t work in L^AT_EX.

```
def\magnification{\afterassignment\m@g\count@}
def\m@g{\mag\count@
\hsize6.5truein\vsiz8.9truein\dimen\footins8truein}
```

`\showoverfull` The following commands are used in debugging:

```
427 \def\showoverfull{\tracingonline\@ne}
```

(End of definition for \showoverfull.)

\showoutput
\loggingoutput

```
428 \gdef\loggingoutput{\tracingoutput\@ne
429   \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
430 \gdef\showoutput{\loggingoutput\showoverfull}
431 \endkernel
```

(End of definition for \showoutput and \loggingoutput.)

\tracingall
\loggingall

```
432 \<latexrelease>\IncludeInRelease{2021/06/01}{\loggingall}
433 \<latexrelease>          {\tracingstacklevels and \tracinglostchars=3}%
434 \*2ekernel | latexrelease)
435 \edef\loggingall{%
436   \tracingstats\tw@
437   \tracingpages\@ne
438   \tracingparagraphs\@ne
439   \tracinggroups\@ne
440   \tracingifs\@ne
441   \tracingscantokens\@ne
442   \tracingnesting\@ne
443   \errorcontextlines\maxdimen
444   \ifdefined\tracingstacklevels \tracingstacklevels\maxdimen \fi
445   \noexpand \loggingoutput
446   \tracingmacros\tw@
447   \tracingcommands\thr@@
448   \tracingrestores\@ne
449   \tracingassigns\@ne
450 }%
451 \def\tracingall{\showoverfull\loggingall}
452 \endkernel | latexrelease)
453 \<latexrelease>\EndIncludeInRelease
454 \<latexrelease>
455 \<latexrelease>\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
456 \<latexrelease>\ifx\tracingscantokens\@undefined
457 \<latexrelease>\gdef\loggingall{%
458   \<latexrelease> \tracingstats\tw@
459   \<latexrelease> \tracingpages\@ne
460   \<latexrelease> \tracinglostchars\@ne
461   \<latexrelease> \tracingparagraphs\@ne
462   \<latexrelease> \errorcontextlines\maxdimen
463   \<latexrelease> \loggingoutput
464   \<latexrelease> \tracingmacros\tw@
465   \<latexrelease> \tracingcommands\tw@
466   \<latexrelease> \tracingrestores\@ne
467   \<latexrelease> }%
468 \<latexrelease>\else
469 \<latexrelease>\gdef\loggingall{%
470   \<latexrelease> \tracingstats\tw@
471   \<latexrelease> \tracingpages\@ne
472   \<latexrelease> \tracinglostchars\tw@
473   \<latexrelease> \tracingparagraphs\@ne
474   \<latexrelease> \tracinggroups\@ne
```

```

475 <latexrelease> \tracingifs\@ne
476 <latexrelease> \tracingscantokens\@ne
477 <latexrelease> \tracingnesting\@ne
478 <latexrelease> \errorcontextlines\maxdimen
479 <latexrelease> \loggingoutput
480 <latexrelease> \tracingmacros\tw@
481 <latexrelease> \tracingcommands\thr@@
482 <latexrelease> \tracingrestores\@ne
483 <latexrelease> \tracingassigns\@ne
484 <latexrelease>}%
485 <latexrelease>\fi
486 <latexrelease>\gdef\tracingall{\showoverfull\loggingall}
487 <latexrelease>\EndIncludeInRelease
488 <latexrelease>
489 <latexrelease>\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
490 <latexrelease>\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
491 <latexrelease> \tracingpages\@ne\tracinglostchars\@ne
492 <latexrelease> \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
493 <latexrelease> \errorcontextlines\maxdimen\loggingoutput}
494 <latexrelease> \gdef\tracingall{\loggingall\showoverfull}
495 <latexrelease>\EndIncludeInRelease

```

(End of definition for \tracingall and \loggingall.)

\tracingnone

```

496 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
497 <latexrelease> \turn off etex tracing}%
498 <*2ekernel | latexrelease>
499 \edef\tracingnone{%
500 \tracingassigns\z@
501 \tracingrestores\z@
502 \tracingonline\z@
503 \tracingcommands\z@
504 \showboxdepth\m@ne
505 \showboxbreadth\m@ne
506 \tracingoutput\z@
507 \errorcontextlines\m@ne
508 \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi
509 \tracingnesting\z@
510 \tracingscantokens\z@
511 \tracingifs\z@
512 \tracinggroups\z@
513 \tracingparagraphs\z@
514 \tracingmacros\z@
515 \tracingpages\z@
516 \tracingstats\z@
517 }%
518 </2ekernel | latexrelease>
519 <latexrelease>\EndIncludeInRelease
520 <latexrelease>
521 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
522 <latexrelease> \turn off etex tracing}%
523 <latexrelease>\ifx\tracingscantokens\@undefined
524 <latexrelease>\def\tracingnone{%

```

```

525 <latexrelease> \tracingonline\z@
526 <latexrelease> \tracingcommands\z@
527 <latexrelease> \showboxdepth\m@ne
528 <latexrelease> \showboxbreadth\m@ne
529 <latexrelease> \tracingoutput\z@
530 <latexrelease> \errorcontextlines\m@ne
531 <latexrelease> \tracingrestores\z@
532 <latexrelease> \tracingparagraphs\z@
533 <latexrelease> \tracingmacros\z@
534 <latexrelease> \tracinglostchars\@ne
535 <latexrelease> \tracingpages\z@
536 <latexrelease> \tracingstats\z@
537 <latexrelease>}%
538 <latexrelease>\else
539 <latexrelease>\def\tracingnone{%
540 <latexrelease> \tracingassigns\z@
541 <latexrelease> \tracingrestores\z@
542 <latexrelease> \tracingonline\z@
543 <latexrelease> \tracingcommands\z@
544 <latexrelease> \showboxdepth\m@ne
545 <latexrelease> \showboxbreadth\m@ne
546 <latexrelease> \tracingoutput\z@
547 <latexrelease> \errorcontextlines\m@ne
548 <latexrelease> \tracingnesting\z@
549 <latexrelease> \tracingscantokens\z@
550 <latexrelease> \tracingifs\z@
551 <latexrelease> \tracinggroups\z@
552 <latexrelease> \tracingparagraphs\z@
553 <latexrelease> \tracingmacros\z@
554 <latexrelease> \tracinglostchars\@ne
555 <latexrelease> \tracingpages\z@
556 <latexrelease> \tracingstats\z@
557 <latexrelease>}%
558 <latexrelease>\fi
559 <latexrelease>\EndIncludeInRelease
560 <latexrelease>
561 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingnone}%
562 <latexrelease> {turn off etex tracing}%
563 <latexrelease>\let\tracingnone\@undefined
564 <latexrelease>\EndIncludeInRelease

```

(End of definition for \tracingnone.)

\hideoutput

```

565 <*2ekernel | latexrelease>
566 <latexrelease>\IncludeInRelease{2015/01/01}{\hideoutput}%
567 <latexrelease> {hide output from tracing}%
568 \def\hideoutput{%
569 \tracingoutput\z@
570 \showboxbreadth\m@ne
571 \showboxdepth\m@ne
572 \tracingonline\m@ne
573 }%
574 <latexrelease>\EndIncludeInRelease

```

```

575 <latexrelease>
576 <latexrelease> \IncludeInRelease{0000/00/00}{\hideoutput}%
577 <latexrelease> {hide output from tracing}%
578 <latexrelease> \let\hideoutput\@undefined
579 <latexrelease> \EndIncludeInRelease
580 </2ekernel | latexrelease>

(End of definition for \hideoutput.)
    LATEX change: \showhyphens Defined later.
    Punctuation affects the spacing.

581 <*2ekernel>
582 \nonfrenchspacing
583 </2ekernel>

```

File 03

ltvers.dtx

1 Version Identification

First we identify the date and version number of this release of L^AT_EX, and set `\everyjob` so that it is printed at the start of every L^AT_EX run.

`\fmtname` A `\patch@level` of 0 or higher denotes an official public release. A negative value indicates a candidate release that is not distributed.

`\fmtversion`

`\latexreleaseversion` If we put code updates into the kernel that are supposed to go into the next release we set the `\patch@level` to -1 and the `\fmtversion` / `\latexreleaseversion` to the dated of the next release (guessed, the real value is not so important and will get corrected when we make the release official).

`\patch@level`

If the `\patch@level` is already at -1 we do nothing here and use the `\fmtversion` date for any new `\IncludeInRelease` line when we add further code.

Finally, if we do make a public release we either just set the `\patch@level` to zero (if our initial guess was good) or we also change the date and then have to additionally change to that date on all the `\IncludeInRelease` statements that used the “guessed” date.

```
1 <*2ekernel>
2 \def\fmtname{LaTeX2e}
3 \edef\fmtversion
4 </2ekernel>
5 <latexrelease>\edef\latexreleaseversion
6 <*2ekernel|latexrelease>
7 {2027-07-01}
8 </2ekernel|latexrelease>
9 <*2ekernel>
10 \def\patch@level{0}
```

For more fine grain control there is the possibility to name the current development branch. This is only used when the `\patch@level` is negative (i.e., a pre-release format) and is intended to help us internally when we locally install a format out of some development branch.

```
\development@branch@name 11 \edef\development@branch@name{develop \the\year-\two@digits\month-\two@digits\day}
```

(End of definition for `\fmtname` and others.)

Check that the format being made is not too old. The error message complains about ‘more than 5 years’ but in fact the error is not triggered until 65 months.

This code is currently not activated as we don’t know if we already got to the last official 2e version (due to staff shortage or due to a successor (think positive:-)).

```
12 \iffalse
13 \def\reserved@a#1/#2/#3\@nil{%
14   \count@ \year
15   \advance\count@-#1\relax
16   \multiply\count@ by 12\relax
17   \advance\count@\month
18   \advance\count@-#2\relax}
19 \expandafter\reserved@a\fmtversion\@nil
```

\count@ is now the age of this file in months. Take a generous definition of ‘year’ so this message is not generated too often.

We store release info in the toks `\LaTeXReleaseInfo` to be used in `\everyjob` but also when `\end{document}` is executed. Instead of using `\typeout` we use `\show@release@info` so that we can write to the log only by changing that to `\wlog`.

```

68     \fi
69     \fi
70 \</2ekernel>

\IncludeInRelease
\EndIncludeInRelease
\@IncludeInRelease
\@IncludeInRelease
\@gobble@IncludeInRelease
\@check@IncludeInRelease

71 \<2ekernel>\let\@currname\@empty
72 \<*2ekernel | latexrelease>
73 \<latexrelease>\newif\if@includeinrelease
74 \<latexrelease>\@includeinreleasefalse
75 \def\IncludeInRelease#1{%
76   \if@includeinrelease
77     \PackageError{latexrelease}{mis-matched IncludeInRelease}%
78       {There is an \noexpand\EndIncludeInRelease missing}%
79   \@includeinreleasefalse
80   \fi
81   \ifnum0%
82     \ifx\new@moduledate\@empty\else 1\fi
83     \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
84     =11
85     \expandafter\@firstoftwo
86   \else
87     \expandafter\@secondoftwo
88   \fi
89   {\finish@module@release{#1}}%
90   {\kernel@ifnextchar[%
91     {\@IncludeInRelease{#1}}
92     {\@IncludeInRelease{#1}[#1]}}}%
93 \def\finish@module@release#1#2#3{%
94   \toks@{[#1] #3}%
95   \begingroup
96     \edef\x{\detokenize\expandafter{\new@modulename}}%
97     \edef\y{\detokenize{#2}}%
98   \expandafter\endgroup
99   \ifx\x\y \else
100     \@latex@error{\noexpand\IncludeInRelease dated #1 in a module is not
101       allowed.\MessageBreak Use a date at least equal to \new@moduledate
102       \space for complete rollback}\@ehd
103   \fi
104   \ifnum\expandafter\@parse@version\new@moduledate//00\@nil
105     >\expandafter\@parse@version\fmtversion//00\@nil
106     \GenericInfo{ }\{Applying: \the\toks@\}%
107   \else
108     \GenericInfo{ }\{Skipping: \the\toks@\}%
109     \expandafter\gobble@finish@module@release
110   \fi}
111 \long\def\gobble@finish@module@release#1\EndModuleRelease{%
112   \EndModuleRelease}

    If a specific date has not been specified in latexrelease use ‘#1’.
113 \def\@IncludeInRelease#1[#2]{\@IncludeInRelease{#2}}
114 \def\@IncludeInRelease#1#2#3{%
115   \toks@{[#1] #3}%
116   \expandafter\ifx\curname\string#2+\@currname+IIR\endcurname\relax

```


If we roll back and the first patch already match then applying that is actually reapplying what is already in the format, i.e., it is useless and possibly allocating new registers. However, it makes the logic simpler so this is the way it is for now. In theory we could always jump over the first patch because that is only really needed for rolling forward. So maybe one day ...

```

117 \ifnum\expandafter\@parse@version#1//00\@nil
118 >\expandafter\@parse@version\fmtversion//00\@nil
119 \GenericInfo{}\{Skipping: \the\toks@}%
120 \expandafter\expandafter\expandafter\@gobble@IncludeInRelease
121 \else
122 \GenericInfo{}\{Applying: \the\toks@}%
123 \@includeinreleasetrue
124 \expandafter\let\csname\string#2+\@currname+IIR\endcsname\@empty
125 \fi
126 \else
127 \GenericInfo{}\{Already applied: \the\toks@}%
128 \expandafter\@gobble@IncludeInRelease
129 \fi
130 }

131 \def\EndIncludeInRelease{%
132 \if@includeinrelease
133 \@includeinreleasefalse
134 \else
135 \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
136 \fi
137 \if@skipping@module
138 \expandafter\new@module@skip
139 \fi}

140 \long\def\@gobble@IncludeInRelease#1\EndIncludeInRelease{%
141 \@includeinreleasefalse
142 \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
143 \@end@check@IncludeInRelease}

144 \long\def\@check@IncludeInRelease#1\IncludeInRelease
145 #2#3\@end@check@IncludeInRelease{%
146 \ifx\@check@IncludeInRelease#2\else
147 \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
148 \fi
149 \if@skipping@module
150 \expandafter\new@module@skip
151 \fi}

```

(End of definition for `\IncludeInRelease` and others.)

1.1 Declaring an all-new module

```

\if@skipping@module \NewModuleRelease
\EndModuleRelease \new@module@skip
\new@modulename
\new@moduledate
152 \let\if@skipping@module\iffalse
153 \def\@skipping@moduletrue{\let\if@skipping@module\iftrue}
154 \def\@skipping@modulefalse{\let\if@skipping@module\iffalse}

```

When we have a whole new module, we can't roll back to a date where such module exists, otherwise hundreds of "command already defined" errors will pop up. But we can't skip it altogether either, because the module might have changes we still want applied, so a more detailed cherry-picking of code chunks have to be done.

```

155 \let\new@modulename\@empty
156 \let\new@moduledate\@empty
157 \def\NewModuleRelease#1#2#3{%
158   \ifx\new@modulename\@empty \else
159     \@latex@error{Nested \noexpand\NewModuleRelease forbidden.}\@ehd \fi
160   \edef\new@moduledate{#1}%
161   \edef\new@modulename{#2}%
162   \GenericInfo{}{BEGIN module: \new@modulename\space (\new@moduledate)}}%
163   \GenericInfo{}{ \@spaces\@spaces\@spaces\space#3\@gobble}%
164   \ifnum\sourceLaTeXdate<%
165     \expandafter\@parse@version\new@moduledate//00\@nil\relax
166     \ifnum\expandafter\@parse@version\fmtversion//00\@nil<%
167       \expandafter\@parse@version\new@moduledate//00\@nil\relax
168       \GenericInfo{}{Skipping module \new@modulename}%
169       \expandafter\expandafter
170       \expandafter\gobble@finish@module@release
171     \else
172       \GenericInfo{}{Applying module \new@modulename}
173       \@skipping@modulefalse
174     \fi
175   \else
176     \GenericInfo{}{Skipping module \new@modulename}
177     \@skipping@moduletrue
178     \expandafter\new@module@skip
179   \fi}
180 \long\def\new@module@skip#1\IncludeInRelease{%
181 \long\def\reserved@a##1\EndModuleRelease{}%
182 \ifrelax\detokenize\expandafter{\reserved@a#1{}}\EndModuleRelease}\relax
183 \else
184   \@latex@error{Missing mandatory \string\IncludeInRelease{0000/00/00}}\@ehc
185   \expandafter\@secondoftwo
186 \fi
187 \@gobble
188   {\@expandtwoargs\IncludeInRelease
189     {0000/00/00}{\new@modulename}%
190     {ERROR! Emergency recovery}%
191     #1}%
192 \IncludeInRelease}
193 \def\EndModuleRelease{%
194   \ifx\new@modulename\@empty
195     \@latex@error{Extra \string\EndModuleRelease.}\@eha
196   \else
197     \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}}%
198   \let\new@modulename\@empty
199   \let\new@moduledate\@empty
200   \@skipping@modulefalse
201 \fi}

(End of definition for \if@skipping@module and others.)

202 </2ekernel | latexrelease>

```

File 04

lualatex.dtx

1 Overview

LuaTeX adds a number of engine-specific functions to TeX. Several of these require set up that is best done in the kernel or need related support functions. This file provides *basic* support for LuaTeX at the L^AT_ΕX 2_ε kernel level plus as a loadable file which can be used with plain TeX and L^AT_ΕX.

This file contains code for both TeX (to be stored as part of the format) and Lua (to be loaded at the start of each job). In the Lua code, the kernel uses the namespace `luatexbase`.

The following `\count` registers are used here for register allocation:

```
\e@alloc@attribute@count Attributes (default 258)
\e@alloc@ccodetable@count Category code tables (default 259)
\e@alloc@luafunction@count Lua functions (default 260)
  \e@alloc@whatsit@count User whatsits (default 261)
  \e@alloc@bytecode@count Lua bytecodes (default 262)
  \e@alloc@luachunk@count Lua chunks (default 263)
```

(`\count 256` is used for `\newmarks` allocation and `\count 257` is used for `\newXeTeXintercharclass` with XeTeX, with code defined in `ltxfinal.dtx`). With any L^AT_ΕX 2_ε kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the L^AT_ΕX 2_ε kernel did not provide any functionality for the extended allocation area).

2 Core TeX functionality

The commands defined here are defined for possible inclusion in a future L^AT_ΕX format, however also extracted to the file `lualatex.tex` which may be used with older L^AT_ΕX formats, and with plain TeX.

```
\newattribute \newattribute{<attribute>}
  Defines a named \attribute, indexed from 1 (i.e. \attribute0 is never defined). Attributes initially have the marker value -7FFFFFFF ('unset') set by the engine.
\newcatcodetable \newcatcodetable{<catcodetable>}
  Defines a named \catcodetable, indexed from 1 (\catcodetable0 is never assigned). A new catcode table will be populated with exactly those values assigned by IniTeX (as described in the LuaTeX manual).
\newluafunction \newluafunction{<function>}
  Defines a named \luafunction, indexed from 1. (Lua indexes tables from 1 so \luafunction0 is not available).
\newluacmd \newluacmd{<function>}
  Like \newluafunction, but defines the command using \luadef instead of just assigning an integer.
```

<code>\newprotectedluacmd</code>	<code>\newluaundef{<function>}</code>	Like <code>\newluacmd</code> , but the defined command is not expandable.
<code>\newwhatsit</code>	<code>\newwhatsit{<whatsit>}</code>	Defines a custom <code>\whatsit</code> , indexed from 1.
<code>\newluabytecode</code>	<code>\newluabytecode{<bytecode>}</code>	Allocates a number for Lua bytecode register, indexed from 1.
<code>\newluachunkname</code>	<code>newluachunkname{<chunkname>}</code>	Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the register (without backslash) into the <code>lua.name</code> table to be used in stack traces.
<code>\catcodetable@initex</code>	Predefined category code tables with the obvious assignments. Note that the <code>latex</code> and	
<code>\catcodetable@string</code>	<code>atletter</code> tables set the full Unicode range to the codes predefined by the kernel.	
<code>\catcodetable@latex</code>	<code>\setattribute{<attribute>}{<value>}</code>	
<code>\catcodetable@atletter</code>	<code>\unsetattribute{<attribute>}</code>	
<code>\setattribute</code>	Set and unset attributes in a manner analogous to <code>\setlength</code> . Note that attributes	
<code>\unsetattribute</code>	take a marker value when unset so this operation is distinct from setting the value to zero.	

3 Plain T_EX interface

The `ltuatex` interface may be used with plain T_EX using `\input{lttuatex}`. This inputs `lttuatex.tex` which inputs `etex.src` (or `etex.sty` if used with L^AT_EX) if it is not already input, and then defines some internal commands to allow the `ltuatex` interface to be defined.

The `luatexbase` package interface may also be used in plain T_EX, as before, by inputting the package `\input luatexbase.sty`. The new version of `luatexbase` is based on this `ltuatex` code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

<code>new_attribute</code>	<code>luatexbase.new_attribute(<attribute>)</code>	Returns an allocation number for the <code><attribute></code> , indexed from 1. The attribute will be initialized with the marker value <code>-"7FFFFFFF</code> ('unset'). The attribute allocation sequence is shared with the T _E X code but this function does <i>not</i> define a token using <code>\attributedef</code> . The attribute name is recorded in the <code>attributes</code> table. A metatable is provided so that the table syntax can be used consistently for attributes declared in T _E X or Lua.
<code>new_whatsit</code>	<code>luatexbase.new_whatsit(<whatsit>)</code>	Returns an allocation number for the custom <code><whatsit></code> , indexed from 1.
<code>new_bytecode</code>	<code>luatexbase.new_bytecode(<bytecode>)</code>	Returns an allocation number for a bytecode register, indexed from 1. The optional <code><name></code> argument is just used for logging.
<code>new_chunkname</code>	<code>luatexbase.new_chunkname(<chunkname>)</code>	Returns an allocation number for a Lua chunk name for use with <code>\directlua</code> and <code>\latelua</code> , indexed from 1. The number is returned and also <code><name></code> argument is added to the <code>lua.name</code> array at that index.
<code>new_luafunction</code>	<code>luatexbase.new_luafunction(<functionname>)</code>	

Returns an allocation number for a lua function for use with `\luafunction`, `\lateluafunction`, and `\luadef`, indexed from 1. The optional `\functionname` argument is just used for logging.

`new_luacommand` `luatexbase.new_luacommand(\csname, \fn, \dots)`

Convenience wrapper combining `new_luafunction` with `token.set_lua`. Allocates a new lua function slot for `\csname` (as `new_luafunction` does), stores the Lua function `\fn` in it, then calls `token.set_lua(\csname, \fnalloc, \dots)` to bind `\csname` to that slot. Any further `\dots` arguments are passed on unchanged to `token.set_lua`.

These functions all require access to a named T_EX count register to manage their allocations. The standard names are those defined above for access from T_EX, e.g. `\e@alloc@attribute@count`, but these can be adjusted by defining the variable `\type_count_name` before loading `ltluatex.lua`, for example

```
local attribute_count_name = "attributetracker"
require("ltluatex")
```

would use a T_EX `\count` (`\countdef`'d token) called `attributetracker` in place of `\e@alloc@attribute@count`.

4.2 Lua access to T_EX register numbers

`registernumber` `luatexbase.registernumber(\name)`

Sometimes (notably in the case of Lua attributes) it is necessary to access a register *by number* that has been allocated by T_EX. This package provides a function to look up the relevant number using LuaT_EX's internal tables. After for example `\newattribute\myattrib, \myattrib` would be defined by (say) `\myattrib=\attribute15`, `luatexbase.registernumber("myattrib")` would then return the register number, 15 in this case. If the string passed as argument does not correspond to a token defined by `\attributedef`, `\countdef` or similar commands, the Lua value `false` is returned.

As an example, consider the input:

```
\newcommand\test[1]{%
\typeout{#1: \expandafter\meaning\csname#1\endcsname^^J
\space\space\space\space
\directlua{tex.write(luatexbase.registernumber("#1") or "bad input")}%
}}

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

\test{@tempdima}
\test{@tempdimb}

\test{strutbox}

\test{sixt@@n}
```

```

\attributedef\myattr=12
\myattr=200
\test{myattr}

```

If the demonstration code is processed with Lua^AT_EX then the following would be produced in the log and terminal output.

```

undefinedrubbish: \relax
      bad input
space: macro:->
      bad input
hbox: \hbox
      bad input
@MM: \mathchar"4E20
      20000
@tempdima: \dimen14
      14
@tempdimb: \dimen15
      15
strutbox: \char"B
      11
sixt@@n: \char"10
      16
myattr: \attribute12
      12

```

Notice how undefined commands, or commands unrelated to registers do not produce an error, just return **false** and so print **bad input** here. Note also that commands defined by `\newbox` work and return the number of the box register even though the actual command holding this number is a `\chardef` defined token (there is no `\boxdef`).

4.3 Module utilities

```
provides_module luatexbase.provides_module(<info>)
```

This function is used by modules to identify themselves; the `info` should be a table containing information about the module. The required field `name` must contain the name of the module. It is recommended to provide a field `date` in the usual L^AT_EX format `yyyy/mm/dd`. Optional fields `version` (a string) and `description` may be used if present. This information will be recorded in the log. Other fields are ignored. If the `version` begins with a digit, a `v` will be added at the start in the log.

```
module_info luatexbase.module_info(<module>, <text>)
```

```
module_warning luatexbase.module_warning(<module>, <text>)
```

```
module_error luatexbase.module_error(<module>, <text>)
```

These functions are similar to L^AT_EX's `\PackageError`, `\PackageWarning` and `\PackageInfo` in the way they format the output. No automatic line breaking is done, you may still use `\n` as usual for that, and the name of the package will be prepended to each output line.

Note that `luatexbase.module_error` raises an actual Lua error with `error()`, which currently means a call stack will be dumped. While this may not look pretty, at least it provides useful information for tracking the error down.

4.4 Callback management

<code>add_to_callback</code>	<code>luatexbase.add_to_callback(<callback>, <function>, <identifier>)</code> Registers the <code><function></code> into the <code><callback></code> with a textual <code><identifier></code> of the function. Functions are inserted into the callback in the order loaded. The <code><identifier></code> should be unique for each callback.
<code>remove_from_callback</code>	<code>luatexbase.remove_from_callback(<callback>, <identifier>)</code> Removes the callback function with <code><identifier></code> from the <code><callback></code> . The removed function and its identifier are returned as the results of this function.
<code>in_callback</code>	<code>luatexbase.in_callback(<callback>, <identifier>)</code> Checks if the <code><identifier></code> matches one of the functions added to the list for the <code><callback></code> , returning a boolean value.
<code>disable_callback</code>	<code>luatexbase.disable_callback(<callback>)</code> Sets the <code><callback></code> to <code>false</code> as described in the LuaTeX manual for the underlying <code>callback.register</code> built-in. Callbacks will only be set to false (and thus be skipped entirely) if there are no functions registered using the callback.
<code>callback_identifiers</code>	A list of the identifiers of functions registered to the specified callback is returned. <code>{}</code> is returned if there are no functions registered.
<code>callback_descriptions</code>	This is a historical alias for <code>callback_identifiers</code> .
<code>create_callback</code>	<code>luatexbase.create_callback(<name>, <type>, <default>)</code> Defines a user defined callback. The last argument is a default function or <code>false</code> .
<code>call_callback</code>	<code>luatexbase.call_callback(<name>, ...)</code> Calls a user defined callback with the supplied arguments.
<code>declare_callback_rule</code>	<code>luatexbase.declare_callback_rule(<name>, <first>, <relation>, <second>)</code> Adds an ordering constraint between two callback functions for callback <code><name></code> . The kind of constraint added depends on <code><relation></code> : before The callback function with identifier <code><first></code> will be executed before the function with identifier <code><second></code> . after The callback function with identifier <code><first></code> will be executed after the function with identifier <code><second></code> . incompatible-warning When both a callback function with identifier <code><first></code> and with identifier <code><second></code> is registered, then a warning is printed when the callback is executed. incompatible-error When both a callback function with identifier <code><first></code> and with identifier <code><second></code> is registered, then an error is printed when the callback is executed. unrelated Any previously declared callback rule between <code><first></code> and <code><second></code> gets disabled.

Every call to `declare_callback_rule` with a specific callback `<name>` and identifier `<first>` and `<second>` overwrites all previous calls with same callback and identifiers.

The callback functions do not have to be registered yet when the functions is called. Only the constraints for which both callback identifiers refer to callbacks registered at the time the callback is called will have an effect.

5 Implementation

```
1 <*2ekernel | tex | latexrelease>
2 <2ekernel | latexrelease>\ifx\directlua\@undefined\else
```

5.1 Minimum LuaTeX version

LuaTeX has changed a lot over time. In the kernel support for ancient versions is not provided: trying to build a format with a very old binary therefore gives some information in the log and loading stops. The cut-off selected here relates to the tree-searching behaviour of `require()`: from version 0.60, LuaTeX will correctly find Lua files in the `texmf` tree without ‘help’.

```
3 <latexrelease>\IncludeInRelease{2015/10/01}
4 <latexrelease>          {\newluafunction}{LuaTeX}%
5 \ifnum\luatexversion<60 %
6   \wlog{*****}
7   \wlog{* LuaTeX version too old for ltuatex support *}
8   \wlog{*****}
9   \expandafter\endinput
10 \fi
```

Two simple L^AT_EX macros from `ltdefns.dtx` have to be defined here because `ltdefns.dtx` is not loaded yet when `ltluatex.dtx` is executed.

```
11 \long\def\@gobble#1{}
12 \long\def\@firstofone#1{#1}
```

5.2 Older L^AT_EX/Plain T_EX setup

```
13 <*tex>
```

Older L^AT_EX formats don’t have the primitives with ‘native’ names: sort that out. If they already exist this will still be safe.

```
14 \directlua{tex.enableprimitives("",tex.extraprimitives("luatex"))}
15 \ifx\@alloc\@undefined
```

In pre-2014 L^AT_EX, or plain T_EX, load `etex.{sty,src}`.

```
16 \ifx\documentclass\@undefined
17   \ifx\loccount\@undefined
18     \input{etex.src}%
19   \fi
20   \catcode'\@=11 %
21   \outer\expandafter\def\csname newfam\endcsname
22     {\alloc@8\fam\chardef\et@xmaxfam}
23 \else
24   \RequirePackage{etex}
25   \expandafter\def\csname newfam\endcsname
26     {\alloc@8\fam\chardef\et@xmaxfam}
27   \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
28 \fi
```

5.2.1 Fixes to `etex.src/etex.sty`

These could and probably should be made directly in an update to `etex.src` which already has some LuaTeX-specific code, but does not define the correct range for LuaTeX.

2015-07-13 higher range in `luatex`.

```
29 \edef \et@xmaxregs {\ifx\directlua\@undefined 32768\else 65536\fi}
```


luatex/xetex also allow more math fam.

```

30 \edef \et@xmaxfam {\ifx\Umathcode\undefined\sixt@@n\else\@cclvi\fi}
31 \count 270=\et@xmaxregs % locally allocates \count registers
32 \count 271=\et@xmaxregs % ditto for \dimen registers
33 \count 272=\et@xmaxregs % ditto for \skip registers
34 \count 273=\et@xmaxregs % ditto for \muskip registers
35 \count 274=\et@xmaxregs % ditto for \box registers
36 \count 275=\et@xmaxregs % ditto for \toks registers
37 \count 276=\et@xmaxregs % ditto for \marks classes

```

and 256 or 16 fam. (Done above due to plain/L^AT_EX differences in l^Auatex.)

```

38 % \outer\def\newfam{\alloc@8\fam\chardef\et@xmaxfam}

```

End of proposed changes to etex.src

5.2.2 luatex specific settings

Switch to global cf `luatex.sty` to leave room for inserts not really needed for luatex but possibly most compatible with existing use.

```

39 \expandafter\let\csname newcount\expandafter\expandafter\endcsname
40     \csname globcount\endcsname
41 \expandafter\let\csname newdimen\expandafter\expandafter\endcsname
42     \csname globdimen\endcsname
43 \expandafter\let\csname newskip\expandafter\expandafter\endcsname
44     \csname globskip\endcsname
45 \expandafter\let\csname newbox\expandafter\expandafter\endcsname
46     \csname globbox\endcsname

```

Define `\e@alloc` as in L^AT_EX (the existing macros in `etex.src` are hard to extend to further register types as they assume specific 26x and 27x count range). For compatibility the existing register allocation is not changed.

```

47 \chardef\e@alloc@top=65535
48 \let\e@alloc\chardef\chardef
49 \def\e@alloc#1#2#3#4#5#6{%
50   \global\advance#3\@ne
51   \e@ch@ck{#3}{#4}{#5}#1%
52   \allocationnumber#3\relax
53   \global#2#6\allocationnumber
54   \wlog{\string#6=\string#1\the\allocationnumber}}%
55 \gdef\e@ch@ck#1#2#3#4{%
56   \ifnum#1<#2\else
57     \ifnum#1=#2\relax
58       #1\@cclvi
59       \ifx\count#4\advance#1 10 \fi
60     \fi
61     \ifnum#1<#3\relax
62     \else
63       \errmessage{No room for a new \string#4}%
64     \fi
65   \fi}%

```

Fix up allocations not to clash with `etex.src`.

```

66 \expandafter\csname newcount\endcsname\@alloc@attribute@count
67 \expandafter\csname newcount\endcsname\@alloc@ccodetable@count
68 \expandafter\csname newcount\endcsname\@alloc@luafunction@count
69 \expandafter\csname newcount\endcsname\@alloc@whatsit@count
70 \expandafter\csname newcount\endcsname\@alloc@bytecode@count
71 \expandafter\csname newcount\endcsname\@alloc@luachunk@count

```

End of conditional setup for plain T_EX / old L^AT_EX.

```

72 \fi
73 \</tex>

```

5.3 Attributes

`\newattribute` As is generally the case for the LuaT_EX registers we start here from 1. Notably, some code assumes that `\attribute0` is never used so this is important in this case.

```

74 \ifx\@alloc@attribute@count\@undefined
75   \countdef\@alloc@attribute@count=258
76   \@alloc@attribute@count=\z@
77 \fi
78 \def\newattribute#1{%
79   \@alloc@attribute\attributedef
80   \@alloc@attribute@count\m@ne\@alloc@top#1%
81 }

```

(End of definition for \newattribute.)

`\setattribute` Handy utilities.
`\unsetattribute`

```

82 \def\setattribute#1#2{#1=\numexpr#2\relax}
83 \def\unsetattribute#1{#1=-"7FFFFFFF\relax}

```

(End of definition for \setattribute and \unsetattribute.)

5.4 Category code tables

`\newcatcodetable` Category code tables are allocated with a limit half of that used by LuaT_EX for everything else. At the end of allocation there needs to be an initialization step. Table 0 is already taken (it's the global one for current use) so the allocation starts at 1.

```

84 \ifx\@alloc@ccodetable@count\@undefined
85   \countdef\@alloc@ccodetable@count=259
86   \@alloc@ccodetable@count=\z@
87 \fi
88 \def\newcatcodetable#1{%
89   \@alloc\catcodetable\chardef
90   \@alloc@ccodetable@count\m@ne{"8000}#1%
91   \initcatcodetable\allocationnumber
92 }

```

(End of definition for \newcatcodetable.)

```

\catcodetable@initex Save a small set of standard tables. The Unicode data is read here in using a parser sim-
\catcodetable@string plified from that in load-unicode-data: only the nature of letters needs to be detected.
\catcodetable@latex
\catcodetable@atletter
93 \newcatcodetable\catcodetable@initex
94 \newcatcodetable\catcodetable@string
95 \begingroup
96 \def\setrangecatcode#1#2#3{%
97 \ifnum#1>#2 %
98 \expandafter\@gobble
99 \else
100 \expandafter\@firstofone
101 \fi
102 {%
103 \catcode#1=#3 %
104 \expandafter\setrangecatcode\expandafter
105 {\number\numexpr#1 + 1\relax}{#2}{#3}
106 }%
107 }
108 \@firstofone{%
109 \catcodetable\catcodetable@initex
110 \catcode0=12 %
111 \catcode13=12 %
112 \catcode37=12 %
113 \setrangecatcode{65}{90}{12}%
114 \setrangecatcode{97}{122}{12}%
115 \catcode92=12 %
116 \catcode127=12 %
117 \savecatcodetable\catcodetable@string
118 \endgroup
119 }%
120 \newcatcodetable\catcodetable@latex
121 \newcatcodetable\catcodetable@atletter
122 \begingroup
123 \def\parseunicodedataI#1;#2;#3;#4\relax{%
124 \parseunicodedataII#1;#3;#2 First>\relax
125 }%
126 \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
127 \ifx\relax#4\relax
128 \expandafter\parseunicodedataIII
129 \else
130 \expandafter\parseunicodedataIV
131 \fi
132 {#1}#2\relax%
133 }%
134 \def\parseunicodedataIII#1#2#3\relax{%
135 \ifnum 0%
136 \if L#21\fi
137 \if M#21\fi
138 >0 %
139 \catcode"#1=11 %
140 \fi
141 }%
142 \def\parseunicodedataIV#1#2#3\relax{%
143 \read\unicoderead to \unicodedataline
144 \if L#2%

```

```

145     \count0="#1 %
146     \expandafter\parseunicodedataV\unicodedataline\relax
147   \fi
148 }%
149 \def\parseunicodedataV#1;#2\relax{%
150   \loop
151     \unless\ifnum\count0>"#1 %
152       \catcode\count0=11 %
153       \advance\count0 by 1 %
154   \repeat
155 }%
156 \def\storedpar{\par}%
157 \chardef\unicoderead=\numexpr\count16 + 1\relax
158 \openin\unicoderead=UnicodeData.txt %
159 \loop\unless\ifeof\unicoderead %
160   \read\unicoderead to \unicodedataline
161   \unless\ifx\unicodedataline\storedpar
162     \expandafter\parseunicodedataI\unicodedataline\relax
163   \fi
164 \repeat
165 \closein\unicoderead
166 \@firstofone{%
167   \catcode64=12 %
168   \savecatcodetable\catcodetable@latex
169   \catcode64=11 %
170   \savecatcodetable\catcodetable@atletter
171 }
172 \endgroup

```

(End of definition for \catcodetable@initex and others.)

5.5 Named Lua functions

\newluafunction Much the same story for allocating LuaTeX functions except here they are just numbers so they are allocated in the same way as boxes. Lua indexes from 1 so once again slot 0 is skipped.

```

173 \ifx\e@alloc@luafunction@count\undefined
174   \countdef\e@alloc@luafunction@count=260
175   \e@alloc@luafunction@count=\z@
176 \fi
177 \def\newluafunction{%
178   \e@alloc\luafunction\e@alloc@chardef
179   \e@alloc@luafunction@count\m@ne\e@alloc@top
180 }

```

(End of definition for \newluafunction.)

\newluacmd Additionally two variants are provided to make the passed control sequence call the function directly.

```

181 \def\newluacmd{%
182   \e@alloc\luafunction\luaodef
183   \e@alloc@luafunction@count\m@ne\e@alloc@top
184 }
185 \def\newprotectedluacmd{%

```

```

186 \e@alloc\luafunction{\protected\luaodef}
187 \e@alloc@luafunction@count\m@ne\e@alloc@top
188 }

```

(End of definition for \newluacmd and \newprotectedluacmd.)

5.6 Custom whatsits

\newwhatsit These are only settable from Lua but for consistency are definable here.

```

189 \ifx\e@alloc@whatsit@count\@undefined
190 \countdef\e@alloc@whatsit@count=261
191 \e@alloc@whatsit@count=\z@
192 \fi
193 \def\newwhatsit#1{%
194 \e@alloc\whatsit\e@alloc@chardef
195 \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
196 }

```

(End of definition for \newwhatsit.)

5.7 Lua bytecode registers

\newluabytocode These are only settable from Lua but for consistency are definable here.

```

197 \ifx\e@alloc@bytecode@count\@undefined
198 \countdef\e@alloc@bytecode@count=262
199 \e@alloc@bytecode@count=\z@
200 \fi
201 \def\newluabytocode#1{%
202 \e@alloc\luabytocode\e@alloc@chardef
203 \e@alloc@bytecode@count\m@ne\e@alloc@top#1%
204 }

```

(End of definition for \newluabytocode.)

5.8 Lua chunk registers

\newluachunkname As for bytecode registers, but in addition we need to add a string to the `lua.name` table to use in stack tracing. We use the name of the command passed to the allocator, with no backslash.

```

205 \ifx\e@alloc@luachunk@count\@undefined
206 \countdef\e@alloc@luachunk@count=263
207 \e@alloc@luachunk@count=\z@
208 \fi
209 \def\newluachunkname#1{%
210 \e@alloc\luachunk\e@alloc@chardef
211 \e@alloc@luachunk@count\m@ne\e@alloc@top#1%
212 \directlua{lua.name[\the\allocationnumber]="\csstring#1"}%
213 }

```

(End of definition for \newluachunkname.)

5.9 Lua loader

Lua code loaded in the format often has to be loaded again at the beginning of every job, so we define a helper which allows us to avoid duplicated code:

```
214 \def\now@and@everyjob#1{%
215   \everyjob\expandafter{\the\everyjob
216     #1%
217   }%
218   #1%
219 }
```

Load the Lua code at the start of every job. For the conversion of T_EX into numbers at the Lua side we need some known registers: for convenience we use a set of systematic names, which means using a group around the Lua loader.

```
220 <2kernel>\now@and@everyjob{%
221   \begingroup
222     \attributedef\attributezero=0 %
223     \chardef      \charzero      =0 %
```

Note name change required on older luatex, for hash table access.

```
224     \countdef      \CountZero    =0 %
225     \dimendef      \dimenzero     =0 %
226     \mathchardef   \mathcharzero =0 %
227     \muskipdef     \muskipzero    =0 %
228     \skipdef       \skipzero      =0 %
229     \toksdef       \tokszero      =0 %
230     \directlua{require("ltnatex")}
231   \endgroup
232 <2kernel>}%
233 <latexrelease>\EndIncludeInRelease

234 <latexrelease>\IncludeInRelease{0000/00/00}
235 <latexrelease>      {\newluafunction}{LuaTeX}%
236 <latexrelease>\let\@alloc@attribute@count\@undefined
237 <latexrelease>\let\newattribute\@undefined
238 <latexrelease>\let\setattribute\@undefined
239 <latexrelease>\let\unsetattribute\@undefined
240 <latexrelease>\let\@alloc@ccodetable@count\@undefined
241 <latexrelease>\let\newcatcodetable\@undefined
242 <latexrelease>\let\catcodetable@initex\@undefined
243 <latexrelease>\let\catcodetable@string\@undefined
244 <latexrelease>\let\catcodetable@latex\@undefined
245 <latexrelease>\let\catcodetable@atletter\@undefined
246 <latexrelease>\let\@alloc@luafunction@count\@undefined
247 <latexrelease>\let\newluafunction\@undefined
248 <latexrelease>\let\@alloc@luafunction@count\@undefined
249 <latexrelease>\let\newwhatsit\@undefined
250 <latexrelease>\let\@alloc@whatsit@count\@undefined
251 <latexrelease>\let\newluabytecode\@undefined
252 <latexrelease>\let\@alloc@bytecode@count\@undefined
253 <latexrelease>\let\newluachunkname\@undefined
254 <latexrelease>\let\@alloc@luachunk@count\@undefined
255 <latexrelease>\directlua{luatexbase.uninstall()}
256 <latexrelease>\EndIncludeInRelease
```

In `\everyjob`, if `luaotfload` is available, load it and switch to TU.

```

257 <latexrelease> \IncludeInRelease{2017/01/01}%
258 <latexrelease>           {\fontencoding}{TU in everyjob}%
259 <latexrelease> \fontencoding{TU}\let\encodingdefault\f@encoding
260 <latexrelease> \ifx\directlua\@undefined\else
261 <2ekernel> \everyjob\expandafter{%
262 <2ekernel>   \the\everyjob
263 <*2ekernel, latexrelease>
264   \directlua{%
265     if xpcall(function ()%
266               require('luaotfload-main')%
267               end, texio.write_nl) then %
268     local _void = luaotfload.main ()%
269     else %
270     texio.write_nl('Error in luaotfload: reverting to OT1')%
271     tex.print('\string\def\string\encodingdefault{OT1}')%
272     end %
273   }%
274   \let\f@encoding\encodingdefault
275   \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
276 </2ekernel, latexrelease>
277 <latexrelease> \fi
278 <2ekernel>   }
279 <latexrelease> \EndIncludeInRelease
280 <latexrelease> \IncludeInRelease{0000/00/00}%
281 <latexrelease>           {\fontencoding}{TU in everyjob}%
282 <latexrelease> \fontencoding{OT1}\let\encodingdefault\f@encoding
283 <latexrelease> \EndIncludeInRelease
284 <2ekernel | latexrelease> \fi
285 </2ekernel | tex | latexrelease>

```

5.10 Lua module preliminaries

```

286 <*lua>

```

Some set up for the Lua module which is needed for all of the Lua functionality added here.

luatexbase Set up the table for the returned functions. This is used to expose all of the public functions.

```

287 luatexbase      = luatexbase or { }
288 local luatexbase = luatexbase

```

(End of definition for luatexbase.)

Some Lua best practice: use local versions of functions where possible.

```

289 local string_format    = string.format
290 local string_gsub      = string.gsub
291 local tex_count        = tex.count
292 local tex_setcount     = tex.setcount
293 local texio_write_nl   = texio.write_nl
294 local flush_list       = node.flush_list
295 local token_set_lua    = token.set_lua

296 local luatexbase_warning
297 local luatexbase_error

```

5.11 Lua module utilities

5.11.1 Module tracking

modules To allow tracking of module usage, a structure is provided to store information and to return it.

```
298 local modules = modules or { }
```

(End of definition for modules.)

provides_module Local function to write to the log.

```
299 local function luatexbase_log(text)
300   texio_write_nl("log", text)
301 end
```

Modelled on `\ProvidesPackage`, we store much the same information but with a little more structure.

```
302 local function provides_module(info)
303   if not (info and info.name) then
304     luatexbase_error("Missing module name for provides_module")
305   end
306   local function spaced(text)
307     return text and (" " .. text) or ""
308   end
309   luatexbase_log(
310     "Lua module: " .. info.name
311     .. spaced(info.date)
312     .. spaced(info.version and string_gsub(info.version or "", "%d"), "v%1")
313     .. spaced(info.description)
314   )
315   modules[info.name] = info
316 end
317 luatexbase.provides_module = provides_module
```

(End of definition for provides_module.)

5.11.2 Module messages

There are various warnings and errors that need to be given. For warnings we can get exactly the same formatting as from `TEX`. For errors we have to make some changes. Here we give the text of the error in the `LATEX` format then force an error from Lua to halt the run. Splitting the message text is done using `\n` which takes the place of `\MessageBreak`.

First an auxiliary for the formatting: this measures up the message leader so we always get the correct indent.

```
318 local function msg_format(mod, msg_type, text)
319   local leader = ""
320   local cont
321   local first_head
322   if mod == "LaTeX" then
323     cont = string_gsub(leader, ".", " ")
324     first_head = leader .. "LaTeX: "
325   else
326     first_head = leader .. "Module " .. msg_type
```



```

327     cont = "(" .. mod .. ")"
328     .. string.gsub(first_head, ".", " ")
329     first_head = leader .. "Module " .. mod .. " " .. msg_type .. ":"
330 end
331 if msg_type == "Error" then
332     first_head = "\n" .. first_head
333 end
334 if string.sub(text,-1) ~= "\n" then
335     text = text .. " "
336 end
337 return first_head .. " "
338     .. string.gsub(
339         text
340     .. "on input line "
341         .. tex.inputlineno, "\n", "\n" .. cont .. " "
342     )
343     .. "\n"
344 end

module_info Write messages.
module_warning
module_error
345 local function module_info(mod, text)
346     texio_write_nl("log", msg_format(mod, "Info", text))
347 end
348 luatexbase.module_info = module_info
349 local function module_warning(mod, text)
350     texio_write_nl("term and log", msg_format(mod, "Warning", text))
351 end
352 luatexbase.module_warning = module_warning
353 local function module_error(mod, text)
354     error(msg_format(mod, "Error", text))
355 end
356 luatexbase.module_error = module_error

(End of definition for module_info, module_warning, and module_error.)
Dedicated versions for the rest of the code here.
357 function luatexbase_warning(text)
358     module_warning("luatexbase", text)
359 end
360 function luatexbase_error(text)
361     module_error("luatexbase", text)
362 end

```

5.12 Accessing register numbers from Lua

Collect up the data from the T_EX level into a Lua table: from version 0.80, LuaT_EX makes that easy.

```

363 local luaregisterbasetable = { }
364 local registermap = {
365     attributezero = "assign_attr"    ,
366     charzero      = "char_given"    ,
367     CountZero     = "assign_int"    ,
368     dimenzero     = "assign_dimen"  ,
369     mathcharzero  = "math_given"    ,

```

```

370 muskipzero    = "assign_mu_skip" ,
371 skipzero      = "assign_skip"    ,
372 tokszero      = "assign_toks"    ,
373 }
374 local createtoken
375 if tex.luatexversion > 81 then
376   createtoken = token.create
377 elseif tex.luatexversion > 79 then
378   createtoken = newtoken.create
379 end
380 local hashtokens = tex.hashtokens()
381 local luatexversion = tex.luatexversion
382 for i,j in pairs (registermap) do
383   if luatexversion < 80 then
384     luaregisterbasetable[hashtokens[i][1]] =
385       hashtokens[i][2]
386   else
387     luaregisterbasetable[j] = createtoken(i).mode
388   end
389 end

```

registernumber Working out the correct return value can be done in two ways. For older LuaTeX releases it has to be extracted from the `hashtokens`. On the other hand, newer LuaTeX's have `newtoken`, and whilst `.mode` isn't currently documented, Hans Hagen pointed to this approach so we should be OK.

```

390 local registernumber
391 if luatexversion < 80 then
392   function registernumber(name)
393     local nt = hashtokens[name]
394     if(nt and luaregisterbasetable[nt[1]]) then
395       return nt[2] - luaregisterbasetable[nt[1]]
396     else
397       return false
398     end
399   end
400 else
401   function registernumber(name)
402     local nt = createtoken(name)
403     if(luaregisterbasetable[nt.cmdname]) then
404       return nt.mode - luaregisterbasetable[nt.cmdname]
405     else
406       return false
407     end
408   end
409 end
410 luatexbase.registernumber = registernumber

```

(End of definition for registernumber.)

5.13 Attribute allocation

new_attribute As attributes are used for Lua manipulations its useful to be able to assign from this end.

```

411 local attributes=setmetatable(
412 {},

```

```

413 {
414 __index = function(t,key)
415 return registernumber(key) or nil
416 end}
417 )
418 luatexbase.attributes = attributes
419 local attribute_count_name =
420     attribute_count_name or "e@alloc@attribute@count"
421 local function new_attribute(name)
422     tex_setcount("global", attribute_count_name,
423         tex_count[attribute_count_name] + 1)
424     if tex_count[attribute_count_name] > 65534 then
425         luatexbase_error("No room for a new \\attribute")
426     end
427     attributes[name]= tex_count[attribute_count_name]
428     luatexbase_log("Lua-only attribute " .. name .. " = " ..
429         tex_count[attribute_count_name])
430     return tex_count[attribute_count_name]
431 end
432 luatexbase.new_attribute = new_attribute

```

(End of definition for new_attribute.)

5.14 Custom whatsit allocation

`new_whatsit` Much the same as for attribute allocation in Lua.

```

433 local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
434 local function new_whatsit(name)
435     tex_setcount("global", whatsit_count_name,
436         tex_count[whatsit_count_name] + 1)
437     if tex_count[whatsit_count_name] > 65534 then
438         luatexbase_error("No room for a new custom whatsit")
439     end
440     luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
441         tex_count[whatsit_count_name])
442     return tex_count[whatsit_count_name]
443 end
444 luatexbase.new_whatsit = new_whatsit

```

(End of definition for new_whatsit.)

5.15 Bytecode register allocation

`new_bytecode` Much the same as for attribute allocation in Lua. The optional `<name>` argument is used in the log if given.

```

445 local bytecode_count_name =
446     bytecode_count_name or "e@alloc@bytecode@count"
447 local function new_bytecode(name)
448     tex_setcount("global", bytecode_count_name,
449         tex_count[bytecode_count_name] + 1)
450     if tex_count[bytecode_count_name] > 65534 then
451         luatexbase_error("No room for a new bytecode register")
452     end

```

```

453   luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
454                 tex_count[bytecode_count_name])
455   return tex_count[bytecode_count_name]
456 end
457 luatexbase.new_bytecode = new_bytecode

```

(End of definition for new_bytecode.)

5.16 Lua chunk name allocation

new_chunkname As for bytecode registers but also store the name in the `lua.name` table.

```

458 local chunkname_count_name =
459     chunkname_count_name or "e@alloc@luachunk@count"
460 local function new_chunkname(name)
461     tex_setcount("global", chunkname_count_name,
462                 tex_count[chunkname_count_name] + 1)
463     local chunkname_count = tex_count[chunkname_count_name]
464     chunkname_count = chunkname_count + 1
465     if chunkname_count > 65534 then
466         luatexbase_error("No room for a new chunkname")
467     end
468     lua.name[chunkname_count]=name
469     luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
470                 chunkname_count .. "\n")
471     return chunkname_count
472 end
473 luatexbase.new_chunkname = new_chunkname

```

(End of definition for new_chunkname.)

5.17 Lua function allocation

new_luafunction Much the same as for attribute allocation in Lua. The optional `<name>` argument is used in the log if given.

```

474 local luafunction_count_name =
475     luafunction_count_name or "e@alloc@luafunction@count"
476 local function new_luafunction(name)
477     tex_setcount("global", luafunction_count_name,
478                 math.max(
479                     #(lua.get_functions_table()),
480                     tex_count[luafunction_count_name])
481                 + 1)
482     lua.get_functions_table()[tex_count[luafunction_count_name]] = false
483     if tex_count[luafunction_count_name] > 65534 then
484         luatexbase_error("No room for a new luafunction register")
485     end
486     luatexbase_log("Lua function " .. (name or "") .. " = " ..
487                 tex_count[luafunction_count_name])
488     return tex_count[luafunction_count_name]
489 end
490 luatexbase.new_luafunction = new_luafunction

```

(End of definition for new_luafunction.)

`new_luaccommand` A convenience wrapper around `new_luafunction` and `token.set_lua`: it allocates the lua function slot, stores $\langle fn \rangle$ there, and binds $\langle csname \rangle$ to it in one step.

```

491 local function new_luaccommand(csname, fn, ...)
492   local fntable = lua.get_functions_table()
493   local fnalloc = new_luafunction(csname)
494   fntable[fnalloc] = fn
495   token_set_lua(csname, fnalloc, ...)
496 end
497 luatexbase.new_luaccommand = new_luaccommand

```

(End of definition for new_luaccommand.)

5.18 Lua callback management

The native mechanism for callbacks in LuaTeX allows only one per function. That is extremely restrictive and so a mechanism is needed to add and remove callbacks from the appropriate hooks.

5.18.1 Housekeeping

The main table: keys are callback names, and values are the associated lists of functions. More precisely, the entries in the list are tables holding the actual function as `func` and the identifier `identifier`. Only callbacks with a non-empty list of functions have an entry in this list.

Actually there are two tables: `realcallbacklist` directly contains the entries as described above while `callbacklist` only directly contains the already sorted entries. Other entries can be queried through `callbacklist` too which triggers a resort.

Additionally `callbackrules` describes the ordering constraints: It contains two element tables with the identifiers of the constrained callback implementations. It can additionally contain a `type` entry indicating the kind of rule. A missing value indicates a normal ordering constraint.

```

498 local realcallbacklist = {}
499 local callbackrules = {}
500 local callbacklist = setmetatable({}, {
501   __index = function(t, name)
502     local list = realcallbacklist[name]
503     local rules = callbackrules[name]
504     if list and rules then
505       local meta = {}
506       for i, entry in ipairs(list) do
507         local t = {value = entry, count = 0, pos = i}
508         meta[entry.identifier], list[i] = t, t
509       end
510       local count = #list
511       local pos = count
512       for i, rule in ipairs(rules) do
513         local rule = rules[i]
514         local pre, post = meta[rule[1]], meta[rule[2]]
515         if pre and post then
516           if rule.type then
517             if not rule.hidden then
518               assert(rule.type == 'incompatible-warning' and luatexbase_warning

```

```

519         or rule.type == 'incompatible-error' and luatexbase_error)(
520             "Incompatible functions \" .. rule[1] .. "\" and \" .. rule[2]
521             .. "\" specified for callback \" .. name .. "\".")
522         rule.hidden = true
523     end
524 else
525     local post_count = post.count
526     post.count = post_count+1
527     if post_count == 0 then
528         local post_pos = post.pos
529         if post_pos ~= pos then
530             local new_post_pos = list[pos]
531             new_post_pos.pos = post_pos
532             list[post_pos] = new_post_pos
533         end
534         list[pos] = nil
535         pos = pos - 1
536     end
537     pre[#pre+1] = post
538 end
539 end
540 end
541 for i=1, count do -- The actual sort begins
542     local current = list[i]
543     if current then
544         meta[current.value.identifier] = nil
545         for j, cur in ipairs(current) do
546             local count = cur.count
547             if count == 1 then
548                 pos = pos + 1
549                 list[pos] = cur
550             else
551                 cur.count = count - 1
552             end
553         end
554         list[i] = current.value
555     else
556         -- Cycle occurred. TODO: Show cycle for debugging
557         -- list[i] = ...
558         local remaining = {}
559         for name, entry in next, meta do
560             local value = entry.value
561             list[#list + 1] = entry.value
562             remaining[#remaining + 1] = name
563         end
564         table.sort(remaining)
565         local first_name = remaining[1]
566         for j, name in ipairs(remaining) do
567             local entry = meta[name]
568             list[i + j - 1] = entry.value
569             for _, post_entry in ipairs(entry) do
570                 local post_name = post_entry.value.identifier
571                 if not remaining[post_name] then
572                     remaining[post_name] = name

```

```

573         end
574     end
575 end
576 local cycle = {first_name}
577 local index = 1
578 local last_name = first_name
579 repeat
580     cycle[last_name] = index
581     last_name = remaining[last_name]
582     index = index + 1
583     cycle[index] = last_name
584 until cycle[last_name]
585 local length = index - cycle[last_name] + 1
586 table.move(cycle, cycle[last_name], index, 1)
587 for i=2, length//2 do
588     cycle[i], cycle[length + 1 - i] = cycle[length + 1 - i], cycle[i]
589 end
590 error('Cycle occurred at ' .. table.concat(cycle, ' -> ', 1, length))
591 end
592 end
593 end
594 realcallbacklist[name] = list
595 t[name] = list
596 return list
597 end
598 })

```

Numerical codes for callback types, and name-to-value association (the table keys are strings, the values are numbers).

```

599 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
600 local types = {
601     list      = list,
602     data      = data,
603     exclusive = exclusive,
604     simple    = simple,
605     reverselist = reverselist,
606 }

```

Now, list all predefined callbacks with their current type, based on the LuaTeX manual version 1.01. A full list of the currently-available callbacks can be obtained using

```

\directlua{
    for i,_ in pairs(callback.list()) do
        texio.write_nl("- " .. i)
    end
}
\bye

```

in plain LuaTeX. (Some undocumented callbacks are omitted as they are to be removed.)

```

607 local callbacktypes = callbacktypes or {

```

Section 8.2: file discovery callbacks.

```

608     find_read_file      = exclusive,
609     find_write_file     = exclusive,
610     find_font_file      = data,

```

```

611 find_output_file = data,
612 find_format_file = data,
613 find_vf_file     = data,
614 find_map_file    = data,
615 find_enc_file    = data,
616 find_pk_file     = data,
617 find_data_file   = data,
618 find_opentype_file = data,
619 find_truetype_file = data,
620 find_type1_file  = data,
621 find_image_file  = data,

622 open_read_file   = exclusive,
623 read_font_file   = exclusive,
624 read_vf_file     = exclusive,
625 read_map_file    = exclusive,
626 read_enc_file    = exclusive,
627 read_pk_file     = exclusive,
628 read_data_file   = exclusive,
629 read_truetype_file = exclusive,
630 read_type1_file  = exclusive,
631 read_opentype_file = exclusive,

```

Not currently used by luatex but included for completeness. may be used by a font handler.

```

632 find_cidmap_file = data,
633 read_cidmap_file = exclusive,

```

Section 8.3: data processing callbacks.

```

634 process_input_buffer = data,
635 process_output_buffer = data,
636 process_jobname      = data,

```

Section 8.4: node list processing callbacks.

```

637 contribute_filter = simple,
638 buildpage_filter  = simple,
639 build_page_insert = exclusive,
640 pre_linebreak_filter = list,
641 linebreak_filter   = exclusive,
642 append_to_vlist_filter = exclusive,
643 post_linebreak_filter = reverselist,
644 hpack_filter       = list,
645 vpack_filter       = list,
646 hpack_quality      = exclusive,
647 vpack_quality      = exclusive,
648 pre_output_filter  = list,
649 process_rule       = exclusive,
650 hyphenate          = simple,
651 ligaturing         = simple,
652 kerning            = simple,
653 insert_local_par   = simple,
654 % mlist_to_hlist   = exclusive,
655 new_graf           = exclusive,

```

Section 8.5: information reporting callbacks.

```

656 pre_dump          = simple,

```



```

657 start_run          = simple,
658 stop_run           = simple,
659 start_page_number  = simple,
660 stop_page_number   = simple,
661 show_error_hook    = simple,
662 show_warning_message = simple,
663 show_error_message  = simple,
664 show_lua_error_hook = simple,
665 start_file          = simple,
666 stop_file           = simple,
667 call_edit           = simple,
668 finish_synctex      = simple,
669 wrapup_run          = simple,

```

Section 8.6: PDF-related callbacks.

```

670 finish_pdffile      = data,
671 finish_pdfpage       = data,
672 page_objnum_provider = data,
673 page_order_index     = data,
674 process_pdf_image_content = data,

```

Section 8.7: font-related callbacks.

```

675 define_font          = exclusive,
676 glyph_info           = exclusive,
677 glyph_not_found      = exclusive,
678 glyph_stream_provider = exclusive,
679 make_extensible       = exclusive,
680 font_descriptor_objnum_provider = exclusive,
681 input_level_string    = exclusive,
682 provide_charproc_data = exclusive,
683 }
684 luatexbase.callbacktypes=callbacktypes

```

Sometimes multiple callbacks correspond to a single underlying engine level callback. Then the engine level callback should be registered as long as at least one of these callbacks is in use. This is implemented though a shared table which counts how many of the involved callbacks are currently in use. The engine level callback is registered iff this count is not 0.

We add `mlist_to_hlist` directly to the list to demonstrate this, but the handler gets added later when it is actually defined.

All callbacks in this list are treated as user defined callbacks.

```

685 local shared_callbacks = {
686   mlist_to_hlist = {
687     callback = "mlist_to_hlist",
688     count = 0,
689     handler = nil,
690   },
691 }
692 shared_callbacks.pre_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist
693 shared_callbacks.post_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist

```

callback.register Save the original function for registering callbacks and prevent the original being used. The original is saved in a place that remains available so other more sophisticated code can override the approach taken by the kernel if desired.

```

694 local callback_register = callback_register or callback.register
695 function callback.register()
696   luatexbase_error("Attempt to use callback.register() directly\n")
697 end

```

(End of definition for `callback.register`.)

5.18.2 Handlers

The handler function is registered into the callback when the first function is added to this callback's list. Then, when the callback is called, the handler takes care of running all functions in the list. When the last function is removed from the callback's list, the handler is unregistered.

More precisely, the functions below are used to generate a specialized function (closure) for a given callback, which is the actual handler.

The way the functions are combined together depends on the type of the callback. There are currently 4 types of callback, depending on the calling convention of the functions the callback can hold:

simple is for functions that don't return anything: they are called in order, all with the same argument;

data is for functions receiving a piece of data of any type except node list head (and possibly other arguments) and returning it (possibly modified): the functions are called in order, and each is passed the return value of the previous (and the other arguments untouched, if any). The return value is that of the last function;

list is a specialized variant of *data* for functions filtering node lists. Such functions are called with a node list head as the first argument and may return either the head of a modified node list, or the boolean values **true** or **false**. The functions are chained the same way as for *data* except for the following cases. If a function returns **false**, then **false** is immediately returned and the following functions are *not* called. If a function returns **true**, then the same head is passed to the next function. If all functions return **true**, then the original head is returned, otherwise the return value of the last function not returning **true** is used.

reverselist is a specialized variant of *list* which executes functions in inverse order.

exclusive is for functions with more complex signatures; functions in this type of callback are *not* combined: An error is raised if a second callback is registered.

Handler for **data** callbacks.

```

698 local function data_handler(name)
699   return function(data, ...)
700     for _,i in ipairs(callbacklist[name]) do
701       data = i.func(data,...)
702     end
703     return data
704   end
705 end

```

Default for user-defined **data** callbacks without explicit default.

```

706 local function data_handler_default(value)
707   return value
708 end

```

Handler for exclusive callbacks. We can assume `callbacklist[name]` is not empty: otherwise, the function wouldn't be registered in the callback any more.

```

709 local function exclusive_handler(name)
710   return function(...)
711     return callbacklist[name][1].func(...)
712   end
713 end

```

Handler for list callbacks.

```

714 local function list_handler(name)
715   return function(head, ...)
716     local ret
717     for _,i in ipairs(callbacklist[name]) do
718       ret = i.func(head, ...)
719       if ret == false then
720         luatexbase_warning(
721           "Function '" .. i.identifier .. "' returned false\n"
722           .. "in callback '" .. name .. "'")
723       )
724       return false
725     end
726     if ret ~= true then
727       head = ret
728     end
729   end
730   return head
731 end
732 end

```

Default for user-defined list and reverselist callbacks without explicit default.

```

733 local function list_handler_default(head)
734   return head
735 end

```

Handler for reverselist callbacks.

```

736 local function reverselist_handler(name)
737   return function(head, ...)
738     local ret
739     local callbacks = callbacklist[name]
740     for i = #callbacks, 1, -1 do
741       local cb = callbacks[i]
742       ret = cb.func(head, ...)
743       if ret == false then
744         luatexbase_warning(
745           "Function '" .. cb.identifier .. "' returned false\n"
746           .. "in callback '" .. name .. "'")
747       )
748       return false
749     end
750     if ret ~= true then
751       head = ret
752     end
753   end
754   return head
755 end

```

```
756 end
```

Handler for simple callbacks.

```
757 local function simple_handler(name)
758   return function(...)
759     for _,i in ipairs(callbacklist[name]) do
760       i.func(...)
761     end
762   end
763 end
```

Default for user-defined simple callbacks without explicit default.

```
764 local function simple_handler_default()
765 end
```

Keep a handlers table for indexed access and a table with the corresponding default functions.

```
766 local handlers = {
767   [data]      = data_handler,
768   [exclusive] = exclusive_handler,
769   [list]      = list_handler,
770   [reverselist] = reverselist_handler,
771   [simple]     = simple_handler,
772 }
773 local defaults = {
774   [data]      = data_handler_default,
775   [exclusive] = nil,
776   [list]      = list_handler_default,
777   [reverselist] = list_handler_default,
778   [simple]     = simple_handler_default,
779 }
```

5.18.3 Public functions for callback management

Defining user callbacks perhaps should be in package code, but impacts on `add_to_callback`. If a default function is not required, it may be declared as `false`. First we need a list of user callbacks.

```
780 local user_callbacks_defaults = {}
```

`create_callback` The allocator itself.

```
781 local function create_callback(name, ctype, default)
782   local ctype_id = types[ctype]
783   if not name or name == ""
784   or not ctype_id
785   then
786     luatexbase_error("Unable to create callback:\n" ..
787                       "valid callback name and type required")
788   end
789   if callbacktypes[name] then
790     luatexbase_error("Unable to create callback '" .. name ..
791                       "':\ncallback is already defined")
792   end
793   default = default or defaults[ctype_id]
794   if not default then
```

```

795     luatexbase_error("Unable to create callback '" .. name ..
796                       "':\ndefault is required for '" .. ctype ..
797                       "' callbacks")
798   elseif type (default) ~= "function" then
799     luatexbase_error("Unable to create callback '" .. name ..
800                       "':\ndefault is not a function")
801   end
802   user_callbacks_defaults[name] = default
803   callbacktypes[name] = ctype_id
804 end
805 luatexbase.create_callback = create_callback

```

(End of definition for create_callback.)

call_callback Call a user defined callback. First check arguments.

```

806 local function call_callback(name,...)
807   if not name or name == "" then
808     luatexbase_error("Unable to create callback:\n" ..
809                       "valid callback name required")
810   end
811   if user_callbacks_defaults[name] == nil then
812     luatexbase_error("Unable to call callback '" .. name
813                       .. "':\nunknown or empty")
814   end
815   local l = callbacklist[name]
816   local f
817   if not l then
818     f = user_callbacks_defaults[name]
819   else
820     f = handlers[callbacktypes[name]](name)
821   end
822   return f(...)
823 end
824 luatexbase.call_callback=call_callback

```

(End of definition for call_callback.)

in_callback Look for a function identifier in a callback.

```

825 local function in_callback(name, identifier)
826   if not name
827     or name == ""
828     or not realcallbacklist[name]
829     or not callbacktypes[name]
830     or not identifier then
831     return false
832   end
833   for _, i in pairs(realcallbacklist[name]) do
834     if i.identifier == identifier then
835       return true
836     end
837   end
838   return false
839 end
840 luatexbase.in_callback = in_callback

```

(End of definition for `in_callback`.)

`add_to_callback` Add a function to a callback. First check arguments.

```
841 local function add_to_callback(name, func, identifier)
842   if not name or name == "" then
843     luatexbase_error("Unable to register callback:\n" ..
844                       "valid callback name required")
845   end
846   if not callbacktypes[name] or
847     type(func) ~= "function" or
848     not identifier or
849     identifier == "" then
850     luatexbase_error(
851       "Unable to register callback.\n\n"
852       .. "Correct usage:\n"
853       .. "add_to_callback(<callback>, <function>, <identifier>)"
854     )
855   end
```

Check if the specified identifier is already registered. This used to be allowed but will become an error in a future release.

```
856   if in_callback(name, identifier) then
857     luatexbase_warning(string_format(
858       "The identifier %s is already in use in callback %s.\n\nz
859       Please change the identifier of the newly registered callback \z
860       to avoid errors in the next LaTeX release", identifier, name))
861   end
```

Then test if this callback is already in use. If not, initialize its list and register the proper handler.

```
862   local l = realcallbacklist[name]
863   if l == nil then
864     l = { }
865     realcallbacklist[name] = l
```

Handle count for shared engine callbacks.

```
866   local shared = shared_callbacks[name]
867   if shared then
868     shared.count = shared.count + 1
869     if shared.count == 1 then
870       callback_register(shared.callback, shared.handler)
871     end
```

If it is not a user defined callback use the primitive callback register.

```
872   elseif user_callbacks_defaults[name] == nil then
873     callback_register(name, handlers[callbacktypes[name]](name))
874   end
875 end
```

Actually register the function and give an error if more than one `exclusive` one is registered.

```
876   local f = {
877     func      = func,
878     identifier = identifier,
879   }
880   if callbacktypes[name] == exclusive then
```

```

881     if #l == 1 then
882         luatexbase_error(
883             "Cannot add second callback to exclusive function\n'" ..
884             name .. "'")
885     end
886 end
887 table.insert(l, f)
888 callbacklist[name] = nil

```

Keep user informed.

```

889     luatexbase_log(
890         "Inserting '" .. identifier .. "' in '" .. name .. "'."
891     )
892 end
893 luatexbase.add_to_callback = add_to_callback

```

(End of definition for add_to_callback.)

declare_callback_rule Add an ordering constraint between two callback implementations

```

894 local function declare_callback_rule(name, desc1, relation, desc2)
895     if not callbacktypes[name] or
896         not desc1 or not desc2 or
897         desc1 == "" or desc2 == "" then
898         luatexbase_error(
899             "Unable to create ordering constraint. "
900             .. "Correct usage:\n"
901             .. "declare_callback_rule(<callback>, <identifier_a>, <identifier_b>)"
902         )
903     end
904     if relation == 'before' then
905         relation = nil
906     elseif relation == 'after' then
907         desc2, desc1 = desc1, desc2
908         relation = nil
909     elseif relation == 'incompatible-warning' or relation == 'incompatible-error' then
910     elseif relation == 'unrelated' then
911     else
912         luatexbase_error(
913             "Unknown relation type in declare_callback_rule"
914         )
915     end
916     callbacklist[name] = nil
917     local rules = callbackrules[name]
918     if rules then
919         for i, rule in ipairs(rules) do
920             if rule[1] == desc1 and rule[2] == desc2 or rule[1] == desc2 and rule[2] == desc1 then
921                 if relation == 'unrelated' then
922                     table.remove(rules, i)
923                 else
924                     rule[1], rule[2], rule.type = desc1, desc2, relation
925                 end
926             end
927         end
928     end
929     if relation ~= 'unrelated' then

```

```

930     rules[#rules + 1] = {desc1, desc2, type = relation}
931   end
932   elseif relation ~= 'unrelated' then
933     callbackrules[name] = {{desc1, desc2, type = relation}}
934   end
935 end
936 luatexbase.declare_callback_rule = declare_callback_rule

```

(End of definition for declare_callback_rule.)

remove_from_callback Remove a function from a callback. First check arguments.

```

937 local function remove_from_callback(name, identifier)
938   if not name or name == "" then
939     luatexbase_error("Unable to remove function from callback:\n" ..
940                       "valid callback name required")
941   end
942   if not callbacktypes[name] or
943     not identifier or
944     identifier == "" then
945     luatexbase_error(
946       "Unable to remove function from callback.\n\n"
947       .. "Correct usage:\n"
948       .. "remove_from_callback(<callback>, <identifier>)"
949     )
950   end
951   local l = realcallbacklist[name]
952   if not l then
953     luatexbase_error(
954       "No callback list for '" .. name .. "'\n")
955   end

```

Loop over the callback's function list until we find a matching entry. Remove it and check if the list is empty: if so, unregister the callback handler.

```

956   local index = false
957   for i,j in ipairs(l) do
958     if j.identifier == identifier then
959       index = i
960       break
961     end
962   end
963   if not index then
964     luatexbase_error(
965       "No callback '" .. identifier .. "' registered for '" ..
966       name .. "'\n")
967   end
968   local cb = l[index]
969   table.remove(l, index)
970   luatexbase_log(
971     "Removing '" .. identifier .. "' from '" .. name .. "'."
972   )
973   if #l == 0 then
974     realcallbacklist[name] = nil
975     callbacklist[name] = nil
976     local shared = shared_callbacks[name]
977     if shared then

```



```

978     shared.count = shared.count - 1
979     if shared.count == 0 then
980         callback_register(shared.callback, nil)
981     end
982     elseif user_callbacks_defaults[name] == nil then
983         callback_register(name, nil)
984     end
985 end
986 return cb.func,cb.identifier
987 end
988 luatexbase.remove_from_callback = remove_from_callback

```

(End of definition for remove_from_callback.)

disable_callback As we subvert the engine interface we need to provide a way to access this functionality.

```

989 local function disable_callback(name)
990     if(realcallbacklist[name] == nil) then
991         callback_register(name, false)
992     else
993         luatexbase_error("Callback list for " .. name .. " not empty")
994     end
995 end
996 luatexbase.disable_callback = disable_callback

```

(End of definition for disable_callback.)

callback_identifiers List the identifiers of functions registered for the given callback. This will sort the list if necessary.

```

997 local function callback_identifiers (name)
998     local d = {}
999     if not name
1000     or name == ""
1001     or not realcallbacklist[name]
1002     or not callbacktypes[name]
1003     then
1004         return d
1005     else
1006         for k, i in pairs(callbacklist[name]) do
1007             d[k]= i.identifier
1008         end
1009     end
1010     return d
1011 end
1012 luatexbase.callback_descriptions =callback_identifiers
1013 luatexbase.callback_identifiers = callback_identifiers

```

(End of definition for callback_identifiers.)

uninstall Unlike at the $\text{\TeX}$ level, we have to provide a back-out mechanism here at the same time as the rest of the code. This is not meant for use by anything other than `latexrelease`: as such this is *deliberately* not documented for users!

```

1014 local function uninstall()
1015     module_info(
1016         "luatexbase",

```

```

1017     "Uninstalling kernel luatexbase code"
1018   )
1019   callback.register = callback_register
1020   luatexbase = nil
1021 end
1022 luatexbase.uninstall = uninstall

```

(End of definition for uninstall.)

mlist_to_hlist To emulate these callbacks, the “real” `mlist_to_hlist` is replaced by a wrapper calling the wrappers before and after.

```

1023 create_callback('pre_mlist_to_hlist_filter', 'list')
1024 create_callback('mlist_to_hlist', 'exclusive', node.mlist_to_hlist)
1025 create_callback('post_mlist_to_hlist_filter', 'reverselist')
1026 function shared_callbacks.mlist_to_hlist.handler(head, display_type, need_penalties)
1027   local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
1028   if current == false then
1029     flush_list(head)
1030     return nil
1031   end
1032   current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
1033   local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
1034   if post == false then
1035     flush_list(current)
1036     return nil
1037   end
1038   return post
1039 end

```

(End of definition for mlist_to_hlist.)

```

1040 </lua>

```

Reset the catcode of @.

```

1041 <tex>\catcode'\@=\etacatcode\relax

```

File 05

ltxexpl.dtx

1 expl3-dependent code

1.1 Loader

```
\@kernel@after@enddocument
\@kernel@before@enddocument@afterlastpage
\@kernel@after@enddocument@afterlastpage
```

These three kernel hooks are used by the shipout code. They are defined early (i.e., here) because the lthooks code adds material to them.

```
1 <*2ekernel | latexrelease>
2 <latexrelease> \IncludeInRelease{2020/10/01}%
3 <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}
```

We initialize these kernel hooks only when they do not already exist. Otherwise they would be set to \@empty on rollback, which would be wrong because code that has been added to them may still have to be executed in the rollback situation. Instead code that writes to them needs to handle the rollback as needed. It is likely that we have to change that approach in the future, but for now it should do. (It is enough to test only for the existence of one hook, as all got added at the same time.)

```
4 \ifx\@kernel@after@enddocument\undefined
5   \let \@kernel@after@enddocument \@empty
6   \let \@kernel@after@enddocument@afterlastpage \@empty
```

For the similar reasons we also define those that are used in \document because they too get material added to in early modules.

```
\@kernel@before@begindocument
\@kernel@after@begindocument
```

```
7   \let \@kernel@before@begindocument \@empty
8   \let \@kernel@after@begindocument \@empty
9 \fi
```

This one actually came later so we need to have a separate check, but otherwise we don't bother with a separate rollback.

```
10 \ifx\@kernel@before@enddocument@afterlastpage\undefined
11   \let \@kernel@before@enddocument@afterlastpage \@empty
12 \fi
13 <latexrelease> \EndIncludeInRelease
14 <latexrelease> \IncludeInRelease{0000/00/00}%
15 <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}
16 <latexrelease> \let\@kernel@after@enddocument\undefined
17 <latexrelease> \let\@kernel@before@enddocument@afterlastpage\undefined
18 <latexrelease> \let\@kernel@after@enddocument@afterlastpage\undefined
19 <latexrelease> \let\@kernel@before@begindocument\undefined
20 <latexrelease> \let\@kernel@after@begindocument\undefined
21 </2ekernel | latexrelease>
22 <latexrelease> \EndIncludeInRelease
```

(End of definition for \@kernel@after@enddocument and others.)

First define some blank commands, so that in case something goes wrong while loading expl3, we won't get strange Undefined control sequence errors.

```
23 <*2ekernel | latexrelease>
24 <latexrelease> \IncludeInRelease{2020/10/01}%
25 <latexrelease> {\expl@sys@load@backend@}{Roll forward support}%
```

```

26 \def\reserved@a#1{\ifdefined#1\else\def#1{}\fi}
27 \reserved@a\@expl@sys@load@backend@@
28 \reserved@a\@expl@push@filename@@
29 \reserved@a\@expl@push@filename@aux@@
30 \reserved@a\@expl@pop@filename@@
31 <latexrelease> \EndIncludeInRelease
32 </2ekernel | latexrelease>

```

Create a hook for last-minute expl3 material.

```

33 <*2ekernel>
34 \def\@expl@finalise@setup@@{}
35 \def\@expl@finalize@setup@@{}
36 </2ekernel>

```

Now define some basics to support loading expl3. These macros can be defined here safely, because they are redefined later on by the kernel, so we define simpler versions just to suit our needs.

```

37 <*2ekernel>
38 \long\def\@gobble#1{}
39 \long\def\@firstofone#1{#1}
40 \long\def\@firstoftwo#1#2{#1}
41 \long\def\@secondoftwo#1#2{#2}
42 \long\def\IfFileExists#1{%
43   \openin\@inputcheck"#1" %
44   \ifeof\@inputcheck
45     \expandafter\@secondoftwo
46   \else
47     \closein\@inputcheck
48     \expandafter\@firstoftwo
49   \fi}
50 \long\def\@ifnextchar#1#2#3{%
51   \let\reserved@d=#1%
52   \def\reserved@a{#2}%
53   \def\reserved@b{#3}%
54   \futurelet\@let@token\@ifnch}
55 \def\@ifnch{%
56   \ifx\@let@token\reserved@d
57     \expandafter\reserved@a
58   \else
59     \expandafter\reserved@b
60   \fi}
61 </2ekernel>

```

If we are doing a rollback with a format containing expl3 we aren't reloading it as that creates havoc. This may need a refined version!

```

62 <*2ekernel | latexrelease>
63 <latexrelease> \IncludeInRelease{2020/10/01}%
64 <latexrelease> {expl3}{Pre-load expl3}%
65 \expandafter\ifx\csname tex\string _let:D\endcsname\relax
66   \expandafter\@firstofone
67 \else
68   \GenericInfo{}\{Skipping: expl3 code already part of the format}%
69 <2ekernel> \expandafter\endinput
70 <latexrelease> \expandafter\@gobble
71 \fi

```

Check for the required primitive/engine support and the existence of a loader.

```

72  {%
73    \IfFileExists{expl3.ltx}
74    {%
75      \ifnum0%
76        \ifdefined\pdffilesize 1\fi
77        \ifdefined\filesize 1\fi
78        \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
79        \ifdefined\kanjiskip 1\fi
80        >0 %
81        \expandafter\@firstofone
82    \else

```

In 2ekernel mode, an error is fatal and building the format is aborted. Use `\batchmode \read -1 to \tokenlist`, which errors with

! Emergency stop. (cannot \read from terminal in nonstop modes)

and aborts the T_EX run. In latexrelease mode, raise an error and do nothing. Both ways, the error message shows the minimum expl3 engine requirements.

```

83 <2ekernel>          \def~{ }\def\MessageBreak{^^J~~~~~}%
84 <2ekernel>          \errmessage{LaTeX Error:
85 <latexrelease>      \@latex@error{%
86                    LaTeX requires the e-TeX primitives and additional\MessageBreak
87                    functionality available in the engines:\MessageBreak
88                    - pdfTeX v1.40\MessageBreak
89                    - XeTeX v0.99992\MessageBreak
90                    - LuaTeX v0.95\MessageBreak
91                    - e-(u)pTeX mid-2012\MessageBreak
92                    or later%
93 <latexrelease>      }\@ehd \expandafter\@gobble
94 <2ekernel>          }\batchmode \read -1 to \reserved@a
95                    \fi
96                }
97            {%
98 <*2ekernel>
99            \errmessage{LaTeX requires expl3}%
100            \batchmode \read -1 to \reserved@a
101 </2ekernel>

```

We do not support a roll forward across 2019. You need to start with 2019 if you want to get to 2020 or beyond.

```

102 <*latexrelease>
103     \@latex@warning@no@line
104     {You need a format that already contains a recent\MessageBreak
105     expl3 as part of the kernel, e.g. at least a kernel\MessageBreak
106     from 2019 to roll forward to that date!\MessageBreak
107     --- I'm giving up!\MessageBreak\MessageBreak
108     Note that manually loading the expl3 package\MessageBreak
109     from your distribution is not enough}%
110     \batchmode \read -1 to \reserved@a
111 </latexrelease>
112     }%
113     {\input expl3.ltx }%
114 }

```

```

115 <latexrelease>\EndIncludeInRelease
116 <latexrelease>

```

To support roll-forward for the case where `xparse` is fully integrated into the kernel, we do not need to repeat the complex test above as we can simply look for the marker command.

```

117 <latexrelease>\IncludeInRelease{2020/02/02}%
118 <latexrelease>          {expl3}{Pre-load expl3}%
119 <latexrelease>\IfFileExists{expl3.ltx}
120 <latexrelease>  {%
121 <latexrelease>    \ifnum0%
122 <latexrelease>      \ifdefined\pdffilesize 1\fi
123 <latexrelease>      \ifdefined\filesize 1\fi
124 <latexrelease>      \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
125 <latexrelease>      >0 %
126 <latexrelease>    \else
127 <latexrelease>      \message{Skipping expl3-dependent extensions}
128 <latexrelease>      \expandafter\@gobbletwo
129 <latexrelease>    \fi
130 <latexrelease>  }
131 <latexrelease>  {%
132 <latexrelease>    \message{Skipping expl3-dependent extensions}%
133 <latexrelease>    \@gobbletwo
134 <latexrelease>  }%
135 <latexrelease>\input{expl3.ltx}
136 <latexrelease>\EndIncludeInRelease

```

We need to be sure that `expl3` is sufficiently new that all of the required functions exist. As the mechanism for date checking isn't in place yet, that is done using a marker function. We use the same abort trick as earlier.

```

137 \ifcsname\detokenize{tl_if_head_eq_charcode:eNTF}\endcsname
138 \else
139   \def~{ } \def\MessageBreak{^^J~~~~~~}%
140   \errmessage{LaTeX Error:
141     L3 Programming layer too old.\MessageBreak
142     You need to update your installation of 'l3kernel'.\MessageBreak
143     LaTeX will abort!}%
144   \batchmode \read -1 to \reserved@a
145 \fi

```

Now in `latexrelease` mode, redefine a few commands to avoid “already defined” errors.

```

146 <latexrelease>\@ifundefined{ExplSyntaxOff}{\}{\latexrelease@postltxexpl}

```

1.2 Using `expl3` code

In order to ease the implementation of some new features in $\text{\LaTeX} 2_{\epsilon}$ we may (temporarily) use some coding based on the `expl3`-code. Such macros will eventually vanish and may be changed unannounced. They are there for internal use in the $\text{\LaTeX} 2_{\epsilon}$ kernel and are not meant to be used in third-party packages. These macros will always have the `@expl@` prefix in their name.

The rest of the name matches the `expl3` name but with all underscores replaced by `@s` and the `:` replaced by `@@`, e.g.,

```
\cs_new_eq:NN \@expl@tl@trim@spaces@apply@@nN \tl_trim_spaces_apply:nN
```

if that `expl3` command is needed in places that are others coded in L^AT_EX 2_ε conventions.

In this file, each release of LaTeX adds an `\IncludeInRelease` block, in which the macros copied for that release were defined. In case a rollback is requested, the entire block is changed.

Each macro copied has a `\changes` entry to explain when and why it was copied, so that further to that may spot it easily.

Here `\cs_gset_eq:NN` is used, instead of the `new` variant because if different releases use that same name for different purposes, each can copy the macro without worrying about redefinitions.

```
147 <latexrelease>\IncludeInRelease{2020/10/01}{\@expl@cs@to@str@@N}%
148 <latexrelease>          {expl3 macros added for the 2020-10-01 release}%
```

The `expl3` activation needs to be inside the release guards as otherwise rolling forward is broken in old kernels that do not have `expl3` loaded.

```
149 \ExplSyntaxOn

150 \cs_gset_eq:NN \@expl@cs@to@str@@N \cs_to_str:N
151 \cs_gset_eq:NN \@expl@str@if@eq@@nnTF \str_if_eq:nnTF

152 \cs_gset_eq:NN \@expl@cs@prefix@spec@@N \cs_prefix_spec:N
153 \cs_if_exist:NTF \cs_parameter_spec:N
154   { \cs_gset_eq:NN \@expl@cs@parameter@spec@@N \cs_parameter_spec:N }
155   { \cs_gset_eq:NN \@expl@cs@parameter@spec@@N \cs_argument_spec:N }
156 \cs_gset_eq:NN \__kernel_cs_parameter_spec:N \@expl@cs@parameter@spec@@N
157 \cs_gset_eq:NN \@expl@cs@replacement@spec@@N \cs_replacement_spec:N

158 \cs_gset_eq:NN \@expl@str@map@function@@NN \str_map_function:NN
159 \cs_gset_eq:NN \@expl@char@generate@@nn \char_generate:nn

160 \ExplSyntaxOff
```

Here we can't assume that `expl3` is available. It will be if we roll back but if this code is executed rolling forward it needs to be pure 2_ε.

```
161 <latexrelease>\EndIncludeInRelease
162 <latexrelease>\IncludeInRelease{0000/00/00}{\@expl@cs@to@str@@N}%
163 <latexrelease>          {expl3 macros added for the 2020-10-01 release}%
164 <latexrelease>\let \@expl@cs@to@str@@N \@undefined
165 <latexrelease>\let \@expl@str@if@eq@@nnTF \@undefined
166 <latexrelease>\let \@expl@cs@prefix@spec@@N \@undefined
167 <latexrelease>\let \@expl@cs@parameter@spec@@N \@undefined
168 <latexrelease>\let \@expl@cs@replacement@spec@@N \@undefined
169 <latexrelease>\let \@expl@str@map@function@@NN \@undefined
170 <latexrelease>\EndIncludeInRelease
171 </2kernel | latexrelease>
```

2 Document-level command names for `expl3` functions

Current home for L3 programming layer functions that we make directly available at the document level. This section may need to be moved later (after `\NewDocumentCommand` is defined in case we want to use that in the setup).

`\fpeval` The expandable command `\fpeval` takes as its argument a floating point expres-

sion and produces a result using the normal rules of mathematics. As this command is expandable it can be used where $\TeX$ requires a number and for example within a low-level $\edef$ operation to give a purely numerical result. See `usrguide3` for further explanation.

`\inteval` The expandable command `\inteval` takes as its argument an integer expression and produces a result using the normal rules of mathematics. The operations recognized are `\dimeval` $+$, $-$, $*$ and $/$ plus parentheses. Division occurs with *rounding*, and ties are rounded away from zero. As this command is expandable it can be used where $\TeX$ requires a number and for example within a low-level $\edef$ operation to give a purely numerical result. See `usrguide3` for further explanation. `\dimeval` and `\skipeval` are similar, but generate fixed and rubber length values, respectively.

`\fpeval` A document level wrapper around the code level function for floating point calculations.

```
\inteval
\dimeval
\skipeval
172 <*2ekernel | latexrelease>
173 <latexrelease>\IncludeInRelease{2022/06/01}%
174 <latexrelease>                {\fpeval}{fp and int calculations}%
175 \ExplSyntaxOn
176 \cs_new_eq:NN \fpeval \fp_eval:n
And a few more, this time wrappers around the e $\TeX$  primitives.
177 \cs_new_eq:NN \inteval \int_eval:n
178 \cs_new_eq:NN \dimeval \dim_eval:n
179 \cs_new_eq:NN \skipeval \skip_eval:n
180 \ExplSyntaxOff
```

(End of definition for `\fpeval` and others.)

```
181 </2ekernel | latexrelease>
182 <latexrelease>\EndIncludeInRelease
183 <latexrelease>\IncludeInRelease{0000/00/00}%
184 <latexrelease>                {\fpeval}{fp and int calculations}%
185 <latexrelease>
186 <latexrelease>\let\fpeval\@undefined
187 <latexrelease>\let\inteval\@undefined
188 <latexrelease>\let\dimeval\@undefined
189 <latexrelease>\let\skipeval\@undefined
190 <latexrelease>\EndIncludeInRelease
```

`\UseName` When declaring new commands with `\NewDocumentCommand` or `\NewCommandCopy`
`\ExpandArgs` or similar, it is sometimes necessary to “construct” the csname. As a general mechanism the L3 programming layer has `\exp_args:N...` for this, but there is no mechanism for it if `\ExplSyntaxOn` is not active. We therefore offer a few of these commands also with CamelCase names.

`\UseName` A document wrapper for changing arguments to cs names for use with `\NewDocumentCommand`
`\ExpandArgs` and similar functions.

```
191 <*2ekernel | latexrelease>
192 <latexrelease>\IncludeInRelease{2022/06/01}%
193 <latexrelease>                {\ExpandArgs}{Some pre-expansion commands}%
194 \ExplSyntaxOn
195 \cs_new_eq:NN \UseName \use:c
```



```

196 \cs_new:Npn \ExpandArgs #1
197 {
198   \cs_if_exist_use:cF { exp_args:N #1 }
199   { \msg_expandable_error:nnn { kernel } { unknown-arg-expansion } {#1} }
200 }
201 \msg_new:nnn { kernel } { unknown-arg-expansion }
202 { Unknown~arg~expansion~"#1" }
203 \ExplSyntaxOff

```

(End of definition for \UseName and \ExpandArgs.)

```

204 </2ekernel | latexrelease>
205 <latexrelease>\EndIncludeInRelease
206 <latexrelease>\IncludeInRelease{0000/00/00}%
207 <latexrelease>          {\ExpandArgs}{Some pre-expansion commands}%
208 <latexrelease>
209 <latexrelease>\let\UseName\@undefined
210 <latexrelease>\let\ExpandArgs\@undefined
211 <latexrelease>\EndIncludeInRelease

```

\IfExplAtLeastTF A pretty simple set of wrappers. \IfExplAtLeastTF was already introduced in 2023
 \IfExplAtLeastT but since we pretend that these commands were always there, even in rollback, things
 \IfExplAtLeastF can be kept simple and we just alter the rollback date.

\IfExplAtLeastTF

```

212 <*2ekernel | latexrelease>
213 <latexrelease>\IncludeInRelease{2025/06/01}%
214 <latexrelease>          {\IfExplAtLeastTF}{Test for expl3 date}%
215 \def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
216 \def\IfExplAtLeastT#1#2{\IfExplAtLeastTF{#1}{#2}\@firstofone\@gobble}
217 \def\IfExplAtLeastF#1{\IfExplAtLeastTF{#1}{}}

```

(End of definition for \IfExplAtLeastTF.)

We make sure the commands are always available.

```

218 </2ekernel | latexrelease>
219 <latexrelease>\EndIncludeInRelease
220 <latexrelease>\IncludeInRelease{0000/00/00}%
221 <latexrelease>          {\IfExplAtLeastTF}{Test for expl3 date}%
222 <latexrelease>
223 <latexrelease>\def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
224 <latexrelease>\def\IfExplAtLeastT#1#2{\IfExplAtLeastTF{#1}{#2}\@firstofone\@gobble}
225 <latexrelease>\def\IfExplAtLeastF#1{\IfExplAtLeastTF{#1}{}}
226 <latexrelease>\EndIncludeInRelease

```

\expandableinput

```

227 <*2ekernel | latexrelease>
228 <latexrelease>\IncludeInRelease{2025/06/01}%
229 <latexrelease>          {\expandableinput}{Expandable input}%
230 \ExplSyntaxOn
231 \cs_new_eq:NN \expandableinput \file_input_raw:n
232 \ExplSyntaxOff

```

(End of definition for \expandableinput.)

```

233 </2ekernel | latexrelease>
234 <latexrelease>\EndIncludeInRelease

```

```
235 <latexrelease>\IncludeInRelease{0000/00/00}%  
236 <latexrelease>                {\expandableinput}{Expandable input}%  
237 <latexrelease>  
238 <latexrelease>\let\expandableinput\@undefined  
239 <latexrelease>\EndIncludeInRelease
```

File 06

ltdefns.dtx

1 Definitions

This section contains commands used in defining other macros.

```
1 <*2ekernel>
```

1.1 Initex initializations

`\two@digits` Prefix a number less than 10 with ‘0’.

```
2 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(End of definition for \two@digits.)

`\typeout` Display something on the terminal.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease> \IncludeInRelease{2020/10/01}%
6 <latexrelease> \typeout{Allow "par" in \typeout}%
7 \protected\long\def\typeout#1{\begingroup
8   \set@display@protect
9   \def\par{^^J^^J}%
10  \immediate\write\@unused{#1}\endgroup}
11 </2ekernel | latexrelease>
12 <latexrelease> \EndIncludeInRelease
13 <latexrelease> \IncludeInRelease{0000/00/00}%
14 <latexrelease> \typeout{Allow "par" in \typeout}%
15 <latexrelease>
16 <latexrelease> \def\typeout#1{\begingroup\set@display@protect
17 <latexrelease> \immediate\write\@unused{#1}\endgroup}
18 <latexrelease> \EndIncludeInRelease
19 <*2ekernel>
```

(End of definition for \typeout.)

`\newlinechar` A char to be used as new-line in output to files.

```
20 \newlinechar‘^^J
```

(End of definition for \newlinechar.)

1.2 Saved versions of TeX primitives

The TeX primitive `\foo` is saved as `\@@foo`. The following primitives are handled in this way:

`\@@par`

```
21 \let\@@par=\par
22 %\let\@@input=\input      %%% moved earlier
23 %\let\@@end=\end          %%%
```

(End of definition for \@@par.)

`\@@hyph` Save original primitive definition.

```

24 \let\@@hyph=\-

```

(End of definition for `\@@hyph`.)

`\@@italiccorr` Save the original italic correction.

```

25 \let\@@italiccorr=\/

```

(End of definition for `\@@italiccorr`.)

`\@height` The following definitions save token space. E.g., using `\@height` instead of `height` saves

`\@depth` 5 tokens at the cost in time of one macro expansion.

```

\@width 26 \def\@height{height} \def\@depth{depth} \def\@width{width}
\@minus 27 \def\@minus{minus}
\@plus 28 \def\@plus{plus}

```

The next one is another 100 tokens worth.

```

29 \def\hb@xt@{\hbox to}

```

(End of definition for `\@height` and others.)

```

30 \message{hacks,}

```

`\hb@xt@`

1.3 Command definitions

This section defines the following commands:

`\@namedef` `{\NAME}`
Expands to `\def\NAME`, except name can contain any characters.

`\@nameuse` `{\NAME}`
Expands to `\NAME`.

`\@ifnextchar` `X{\YES}{\NO}`
Expands to `\YES` if next character is an ‘X’, and to `\NO` otherwise. (Uses `\reserved@a-`
`\reserved@c`.) NOTE: GOBBLES ANY SPACE FOLLOWING IT.

`\@ifstar` `{\YES}{\NO}`
Gobbles following spaces and then tests if next the character is a ‘*’. If it is, then it
gobbles the ‘*’ and expands to `\YES`, otherwise it expands to `\NO`.

`\@dblarg` `{\CMD}{\ARG}`
Expands to `\{\CMD\}[\ARG]{\ARG}`. Use `\@dblarg\CS` when `\CS` takes arguments
`[ARG1]{ARG2}`, where default is `ARG1 = ARG2`.

`\@ifundefined` `{\NAME}{\YES}{\NO}`
: If `\NAME` is undefined then it executes `\YES`, otherwise it executes `\NO`. More precisely,
true if `\NAME` either undefined or = `\relax`.

`\@ifdefinable` `\NAME{\YES}` Executes `\YES` if the user is allowed to define `\NAME`, otherwise it
gives an error. The user can define `\NAME` if `\@ifundefined{\NAME}` is true, ‘`NAME`’ ≠
‘`relax`’ and the first three letters of ‘`NAME`’ are not ‘`end`’, and if `\endNAME` is not defined.

`\newcommand` `*{\FOO}[\i]{\TEXT}`
User command to define `\FOO` to be a macro with `i` arguments (`i = 0` if missing) having
the definition `\TEXT`. Produces an error if `\FOO` already defined.

Normally the command is defined to be `\long` (ie it may take multiple paragraphs
in its argument). In the star-form, the command is not defined as `\long` and a blank line
in any argument to the command would generate an error.

`\renewcommand` `*{\FOO}[\i]{\TEXT}`

Same as `\newcommand`, except it checks if `\F00` already defined.

`\newenvironment` $\star{\langle FOO\rangle}[\langle i\rangle]\{\langle DEF1\rangle\}\{\langle DEF2\rangle\}$
equivalent to:
`\newcommand{\F00}[i]{DEF1} \def{\endF00}{DEF2}`
(or the appropriate star forms).

`\renewenvironment` Obvious companion to `\newenvironment`.

`\@cons` : See description of `\output` routine.

`\@car` `\@car T1 T2 ... Tn\@nil == T1` (unexpanded)

`\@cdr` `\@cdr T1 T2 ... Tn\@nil == T2 ... Tn` (unexpanded)

`\typeout` $\{\langle message\rangle\}$
Produces a warning message on the terminal.

`\typein` $\{\langle message\rangle\}$
Types message, asks the user to type in a command, then executes it

`\typein` $[\langle\backslash CS\rangle]\{\langle MSG\rangle\}$
Same as above, except defines `\CS` to be the input instead of executing it.

`\typein`

```

31 \def\typein{%
32   \let\@typein\relax
33   \@testopt\@xtypein\@typein}

34 \ifx\directlua\@undefined

35 \def\@xtypein[#1]#2{%
36   \typeout{#2}%
37   \advance\endlinechar\@M
38   \read\@inputcheck to#1%
39   \advance\endlinechar-\@M
40   \@typein}%

41 \else

42 \def\@xtypein[#1]#2{%
43   \typeout{#2}%
44   \begingroup \endlinechar\m@ne
45   \read\@inputcheck to#1%
46   \expandafter\endgroup
47   \expandafter\def\expandafter#1\expandafter{#1}%
48   \@typein}%

49 \fi

```

(End of definition for `\typein`.)

`\@namedef`

```

50 \def\@namedef#1{\expandafter\def\csname #1\endcsname}

```

(End of definition for `\@namedef`.)

`\@nameuse`

```

51 \def\@nameuse#1{\csname #1\endcsname}

```

(End of definition for `\@nameuse`.)

```

\@cons
52 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}

(End of definition for \@cons.)

\@car
\@cdr
53 \def\@car#1#2\@nil{#1}
54 \def\@cdr#1#2\@nil{#2}

(End of definition for \@car and \@cdr.)

\@carcube \@carcube T1 ... Tn\@nil = T1 T2 T3 , n > 3
55 \</2ekernel>
56 \<latexrelease>\IncludeInRelease{2020/10/01}{\@carcube}{Make \@carcube long}%
57 \<*2ekernel|latexrelease>
58 \long\def\@carcube#1#2#3#4\@nil{#1#2#3}
59 \</2ekernel|latexrelease>
60 \<latexrelease>\EndIncludeInRelease
61 %
62 \<latexrelease>\IncludeInRelease{0000/00/00}{\@carcube}{Undo: Make \@carcube long}%
63 \<latexrelease>\def\@carcube#1#2#3#4\@nil{#1#2#3}
64 \<latexrelease>\EndIncludeInRelease
65 \<*2ekernel>

(End of definition for \@carcube.)

\@onlypreamble This macro adds its argument to the list of commands stored in \@preamblecmds
\@preamblecmds to be disabled after \begin{document}. These commands are redefined to generate
\@notprerr at this point.
66 \def\@preamblecmds{}
67 \def\@onlypreamble#1{%
68   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
69     \@preamblecmds\do#1}}
70 \@onlypreamble\@onlypreamble
71 \@onlypreamble\@preamblecmds

(End of definition for \@onlypreamble and \@preamblecmds.)

\@star@or@long Look ahead for a *. If present reset \l@ngrel@x so that the next definition, #1, will be
non-long.
72 \</2ekernel>
73 \<*2ekernel|latexrelease>
74 \<latexrelease>\IncludeInRelease{2025/11/01}%
75 \<latexrelease> {\@star@or@long}{Macros without args are short}%
76 \def\@star@or@long#1{%

By default commands defined with \newcommand and friends are not \protected so
\pr@tectedrel@x is set to \relax.
77 \let\pr@tectedrel@x\relax
78 \ifstar
79 {\let\l@ngrel@x\relax#1}%
80 {\let\l@ngrel@x\long#1}}

```

```

81 </2ekernel | latexrelease>
82 <latexrelease>\EndIncludeInRelease
83 <latexrelease>\IncludeInRelease{0000/00/00}%
84 <latexrelease>    {\@star@or@long}{Macros without args are short}%
85 <latexrelease>
86 <latexrelease>\def\@star@or@long#1{%
87 <latexrelease>    \ifstar
88 <latexrelease>        {\let\l@ngrel@x\relax#1}%
89 <latexrelease>        {\let\l@ngrel@x\long#1}}
90 <latexrelease>
91 <latexrelease>\EndIncludeInRelease
92 <*2ekernel>

```

(End of definition for \@star@or@long.)

\l@ngrel@x This is either `\relax` or `\long` depending on whether the `*`-form of a definition command is being executed.

```
93 \let\l@ngrel@x\relax
```

(End of definition for \l@ngrel@x.)

\pr@tectedrel@x Same for the protection status.

```
94 \let\pr@tectedrel@x\relax
```

(End of definition for \pr@tectedrel@x.)

\newcommand User level `\newcommand`.

```
95 \def\newcommand{\@star@or@long\new@command}
```

\new@command

```
96 \def\new@command#1{%
97     \testopt{\@newcommand#1}0}
```

(End of definition for \newcommand and \new@command.)

\@newcommand Handling arguments for `\newcommand`.

```

\@argdef 98 \def\@newcommand#1[#2]{%
\@xargdef 99     \kernel@ifnextchar [{\@xargdef#1[#2]}%
100         {\@argdef#1[#2]}}

```

Define `#1` if it is definable.

Both here and in `\@xargdef` the replacement text is absorbed as an argument because if we are not allowed to make the definition we have to get rid of it completely.

```

101 \long\def\@argdef#1[#2]#3{%
102     \@ifdefinable #1{\@yargdef#1\@ne{#2}{#3}}

```

Handle the second optional argument.

```

103 \long\def\@xargdef#1[#2] [#3]#4{%
104     \@ifdefinable#1{%

```

Define the actual command to be:

```
\def\foo{\@protected@testopt\foo\\foo{default}}
```

where `\\foo` is a c`sname` generated from applying `\csname` and `\string` to `\foo`, ie the actual name contains a backslash and therefore can't clash easily with existing command names. "Default" is the contents of the second optional argument of `(re)newcommand`.

```
105 \expandafter\def\expandafter#1\expandafter{%
106     \expandafter
107     \@protected@testopt
108     \expandafter
109     #1%
110     \csname\string#1\endcsname
111     {#3}}%
```

Now we define the internal macro ie `\\foo` which is supposed to pick up all arguments (optional and mandatory).

```
112 \expandafter\@yargdef
113 \csname\string#1\endcsname
114 \tw@
115 {#2}%
116 {#4}}}
```

(End of definition for `\@newcommand`, `\@argdef`, and `\@xargdef`.)

`\@testopt` This macro encapsulates the most common call to `\@ifnextchar`, saving several tokens each time it is used in the definition of a command with an optional argument. `#1` The code to execute in the case that there is a `[` need not be a single token but can be any sequence of commands that 'expects' to be followed by `[`. If this command were only used in `\newcommand` definitions then `#1` would be a single token and the braces could be omitted from `{#1}` in the definition below, saving a bit of memory.

```
117 \long\def\@testopt#1#2{%
118     \kernel@ifnextchar[#{1}{#1[{#2}]}}}
```

(End of definition for `\@testopt`.)

`\@protected@testopt` Robust version of `\@testopt`. The extra argument (`#1`) must be a single token. If protection is needed the call expands to `\protect` applied to this token, and the 2nd and 3rd arguments are discarded (by `\@x@protect`). Otherwise `\@testopt` is called on the 2nd and 3rd arguments.

This method of making commands robust avoids the need for using up two c`sname`s per command, the price is the extra expansion time for the `\ifx` test.

```
119 \def\@protected@testopt#1{%
120     \ifx\protect\@typeset@protect
121         \expandafter\@testopt
122     \else
123         \@x@protect#1%
124     \fi}
```

(End of definition for `\@protected@testopt`.)

`\@yargdef` These generate a primitive argument specification, from a L^AT_EX [`<digit>`] form; in fact `<digit>` can be anything such that `\number <digit>` is single digit.

`\@yargdef` Reorganized slightly so that `\renewcommand{\reserved@a}[1]{foo}` works. I am not sure this is worth it, as a following `\newcommand` would over-write the definition of `\reserved@a`.

Recall that L^AT_EX2.09 goes into an infinite loop with
`\renewcommand[1]{\@tempa}{foo}`
(DPC 6 October 93).

Reorganized again (DPC 1999). Rather than make a loop to construct the argument spec by counting, just extract the required argument spec by using a delimited argument (delimited by the digit). This is faster and uses less tokens. The coding is slightly odd to preserve the old interface (using `#2 = \tw@` as the flag to surround the first argument with `[]`). But the new method did not allow for the number of arguments `#3` not being given as an explicit digit; hence (further expansion of this argument and use of) `\number` was added later in 1999.

It is not clear why these are still `\long`.

```

125 \long \def \@yargdef #1#2#3{%
126   \ifx#2\tw@
127     \def\reserved@b##11{####1}%
128   \else
129     \let\reserved@b\@gobble
130   \fi
131   \expandafter
132     \@yargd@f \expandafter{\number #3}#1%
133 }

134 </2ekernel>
135 <*2ekernel | latexrelease>
136 <latexrelease>\IncludeInRelease{2025/11/01}%
137 <latexrelease>  {\@yargd@f}{Macros without args are short}%
138 <latexrelease>
139 \long \def \@yargd@f#1#2{%
140   \def \reserved@a ##1#1##2##{%
141     \expandafter\def\expandafter#2\reserved@b ##1#1%
142   }%

```

If the command needs to be `\protected` then `\pr@tectedrel@x` has the appropriate value at this point.

```

143   \pr@tectedrel@x

```

If the command has no arguments then we don't want to to be `\long` (even if `\l@ngrel@x` has been set to `\long`).

```

144   \ifnum#1>\z@ \l@ngrel@x \fi
145   \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
146 }

147 </2ekernel | latexrelease>
148 <latexrelease>\EndIncludeInRelease
149 <latexrelease>\IncludeInRelease{0000/00/00}%
150 <latexrelease>  {\@yargd@f}{Macros without args are short}%
151 <latexrelease>
152 <latexrelease>\long \def \@yargd@f#1#2{%
153 <latexrelease>  \def \reserved@a ##1#1##2##{%
154 <latexrelease>    \expandafter\def\expandafter#2\reserved@b ##1#1%
155 <latexrelease>  }%
156 <latexrelease>  \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
157 <latexrelease>}
158 <latexrelease>
159 <latexrelease>\EndIncludeInRelease
160 <*2ekernel>

```

(End of definition for \@yargdef and \@yargdef.)

\@reargdef

```
161 \long\def\@reargdef#1[#2]{%
162   \@yargdef#1\@one{#2}}
```

(End of definition for \@reargdef.)

\renewcommand Check the command name is already used. If not give an error message. Then temporarily disable \@ifdefinable then call \newcommand. (Previous version \let#1=\relax but this does not work too well if #1 is \@tempa-e.)

```
163 \def\renewcommand{\@star@or@long\renew@command}
```

\renew@command

```
164 \def\renew@command#1{%
165   \begingroup
166   \escapechar\m@ne\xdef\@gtempa{\expandafter\string\@car#1?\@nil}}%
167   \endgroup
168   \expandafter\@ifundefined\@gtempa
169   {\@latex@error{Command \string#1 undefined}\@ehc}%
170   \relax
171   \let\@ifdefinable\@rc@ifdefinable
172   \new@command#1}
```

(End of definition for \renewcommand and \renew@command.)

\@ifdefinable Test if user is allowed to define a command.

\@@ifdefinable

\@rc@ifdefinable

```
173 \long\def\@ifdefinable #1#2{%
174   \edef\reserved@a{\expandafter\@gobble\string #1}%
175   \@ifundefined\reserved@a
176   {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}}%
177   \ifx \reserved@b\@qend \@notdefinable\else
178   \ifx \reserved@a\@qrelax \@notdefinable\else
179     #2%
180   \fi
181   \fi}%
182   \@notdefinable}
```

Saved definition of \@ifdefinable.

```
183 \let\@@ifdefinable\@ifdefinable
```

Version of \@ifdefinable for use with \renewcommand. Does not do the check this time, but restores the normal definition.

```
184 \long\def\@rc@ifdefinable#1#2{%
185   \let\@ifdefinable\@@ifdefinable
186   #2}
```

(End of definition for \@ifdefinable, \@@ifdefinable, and \@rc@ifdefinable.)

\newenvironment

Define a new user environment. #1 is the environment name. #2# Grabs all the tokens up to the first {. These will be any optional arguments. They are not parsed at this point, but are just passed to \@newenv which will eventually call \newcommand. Any optional arguments will then be parsed by \newcommand as it defines the command that executes the ‘begin code’ of the environment.

This #2# trick removed with version 1.2i as it fails if a { occurs in the optional argument. Now use \@ifnextchar directly.

```

187 \def\newenvironment{\@star@or@long\new@environment}

\new@environment 188 \def\new@environment#1{%
189   \@testopt{\@newenva#1}0}

190 \def\@newenva#1[#2]{%
\@newenva 191   \kernel@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}]

192 \def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][#3]}}
\@newenvb (End of definition for \newenvironment and others.)

\renewenvironment Redefine an environment. For \renewenvironment disable \@ifdefinable and then call
\newenvironment. It is OK to \let the argument to \relax here as there should not
be a @temp... environment.
193 \def\renewenvironment{\@star@or@long\renew@environment}

\renew@environment 194 \def\renew@environment#1{%
195   \@ifundefined{#1}%
196     {\@latex@error{Environment #1 undefined}\@ehc
197     }\relax
198   \expandafter\let\csname#1\endcsname\relax
199   \expandafter\let\csname end#1\endcsname\relax
200   \new@environment{#1}}

(End of definition for \renewenvironment and \renew@environment.)

\@newenv The internal version of \newenvironment.
Call \newcommand to define the <begin-code> for the environment. \def is used for
the <end-code> as it does not take arguments. (but may contain \pars)
Make sure that an attempt to define a ‘graf’ or ‘group’ environment fails by temporarily letting the undefined \... (begin code) to the definition of \end... and as a result we get an error if that has a definition.
201 \long\def\@newenv#1#2#3#4{%
202   \@ifundefined{#1}%
203     {\expandafter\let\csname#1\expandafter\endcsname
204       \csname end#1\endcsname}%
205     \relax
206   \expandafter\new@command
207     \csname #1\endcsname#2{#3}%
208     \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}

(End of definition for \@newenv.)

```

`\newif` And here's a different sort of allocation: For example, `\newif\iffoo` creates `\footrue`, `\foofalse` to go with `\iffoo`.

```
209 \def\newif#1{%
210   \count@\escapechar \escapechar\m@ne
211   \let#1\iffalse
212   \@if#1\iftrue
213   \@if#1\iffalse
214   \escapechar\count@}
```

```
\@if 215 \def\@if#1#2{%
216   \expandafter\def\csname\expandafter\@gobbletwo\string#1%
217           \expandafter\@gobbletwo\string#2\endcsname
218           {\let#1#2}}
```

(End of definition for \newif and \@if.)

`\providecommand` `\providecommand` takes the same arguments as `\newcommand`, but discards them if #1 is already defined. Otherwise it just acts like `\newcommand`. This implementation currently leaves any discarded definition in `\reserved@a` (and possibly `\reserved@a`) this wastes a bit of space, but it will be reclaimed as soon as these scratch macros are redefined.

```
219 \def\providecommand{\@star@or@long\provide@command}
```

```
\provide@command 220 \def\provide@command#1{%
221   \begingroup
222   \escapechar\m@ne\xdef\@gtempa{\expandafter\string\@car#1?\@nil}}%
223   \endgroup
224   \expandafter\@ifundefined\@gtempa
225   {\def\reserved@a{\new@command#1}}%
226   {\def\reserved@a{\renew@command\reserved@a}}%
227   \reserved@a}%

```

(End of definition for \providecommand and \provide@command.)

`\CheckCommand` `\CheckCommand` takes the same arguments as `\newcommand`. If the command already exists, with the same definition, then nothing happens, otherwise a warning is issued. Useful for checking the current state before a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: `\newcommand\xxx[2][a]{(#1)(#2)}` then `\CheckCommand\xxx[2][b]{(#1)(#2)}` would *not* generate a warning, but, for instance `\CheckCommand\xxx[2]{(#1)(#2)}` would.

```
228 \def\CheckCommand{\@star@or@long\check@command}
```

`\CheckCommand` is only available in the preamble part of the document.

```
229 \@onlypreamble\CheckCommand
```

```
\check@command 230 \def\check@command#1#2#{\@check@c#1{#2}}
231 \@onlypreamble\check@command
```

(End of definition for \CheckCommand and \check@command.)

`\@check@c` `\CheckCommand` itself just grabs all the arguments we need, without actually looking for [optional argument forms. Now define `\reserved@a`. If `\reserved@a` is then defined, compare it with the “`\#1`” otherwise compare `\reserved@a` with `\#1`.

```

232 \long\def\@check@c#1#2#3{%
233   \expandafter\let\csname\string\reserved@a\endcsname\relax
234   \renewcommand\reserved@a#2{#3}%
235   \@ifundefined{\string\reserved@a}%
236     {\@check@eq#1\reserved@a}%
237     {\expandafter\@check@eq
238       \csname\string#1\expandafter\endcsname
239       \csname\string\reserved@a\endcsname}}
240 \@onlypreamble\@check@c

```

(End of definition for `\@check@c`.)

`\@check@eq` Complain if `\#1` and `\#2` are not `\ifx` equal.

```

241 \def\@check@eq#1#2{%
242   \ifx#1#2\else
243     \@latex@warning@no@line
244       {Command \noexpand#1 has
245       changed.\MessageBreak
246       Check if current package is valid}%
247   \fi}
248 \@onlypreamble\@check@eq

```

(End of definition for `\@check@eq`.)

`\@gobble` The `\@gobble` macro is used to get rid of its argument.

```

\@gobbletwo 249 \long\def \@gobble #1{}
\@gobblethree 250 \long\def \@gobbletwo #1#2{}
\@gobblefour 251 \long\def \@gobblethree #1#2#3{}
252 \long\def \@gobblefour #1#2#3#4{}

```

(End of definition for `\@gobble` and others.)

There are also `\@gobble@om`, `\@gobble@som`, `\@gobble@with@sphack@om`, and `\@gobble@with@sphack@som`. They accept an optional and a mandatory argument, possibly preceded by a star. In all cases the expansion is empty or just manipulates the spaces around the command. Used to disable commands such as `\index` or `\label` in certain situations. Since they are defined with `\DeclareDocumentCommand`, which is not yet available at this point, the actual definition happens in `ltsect.dtx`.

`\@firstofone` Some argument-grabbers.

```

\@firstoftwo 253 \long\def\@firstofone#1{#1}
\@secondoftwo 254 \long\def\@firstoftwo#1#2{#1}
255 \long\def\@secondoftwo#1#2{#2}

```

`\@iden` is another name for `\@firstofone` for compatibility reasons.

```

256 \let\@iden\@firstofone

```

(End of definition for `\@firstofone` and others.)

`\@iden`

`\@thirdofthree` Another grabber now used in the encoding specific section.

```

257 \long\def\@thirdofthree#1#2#3{#3}

```

(End of definition for \@thirdofthree.)

`\@expandtwoargs` A macro to totally expand two arguments to another macro

```
258 </2ekernel>
259 <latexrelease>\IncludeInRelease{2022/11/01}%
260 <latexrelease>      {\@expandtwoargs}{protected edef}%
261 <*2ekernel | latexrelease>
262 \def\@expandtwoargs#1#2#3{%
263 \protected@edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
264 </2ekernel | latexrelease>
265 <latexrelease>\EndIncludeInRelease
266 <latexrelease>\IncludeInRelease{00/00/00}%
267 <latexrelease>      {\@expandtwoargs}{protected edef}%
268 <latexrelease>\def\@expandtwoargs#1#2#3{%
269 <latexrelease>\edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
270 <latexrelease>\EndIncludeInRelease
271 <*2ekernel>
```

(End of definition for \@expandtwoargs.)

`\@backslashchar` A category code 12 backslash.

```
272 \edef\@backslashchar{\expandafter\@gobble\string\\}
```

(End of definition for \@backslashchar.)

1.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in L^AT_EX's commands. Whilst typesetting documents, L^AT_EX makes use of many of T_EX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different occasions when these commands are not safe. These are called 'moving arguments' by L^AT_EX, and consist of:

- writing information to a file, such as indexes or tables of contents.
- writing information to the screen.
- inside an `\edef`, `\message`, `\mark`, or other command which evaluates its argument fully.

The method L^AT_EX uses for making fragile commands robust is to precede them with `\protect`. This can have one of four possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edefs`, `\marks` and other commands which evaluate their arguments fully. More precisely, whenever the content of an `\edef` or `\xdef` etc. can contain arbitrary user input not under the direct control of the programmer, one should use `\protected@edef` instead of `\edef`, etc., so that `\protect` has a suitable definition and the user input will not break if it contains fragile commands.

`\@unexpandable@protect`

```
273 \def\@unexpandable@protect{\noexpand\protect\noexpand}
```

(End of definition for \@unexpandable@protect.)

```
\DeclareRobustCommand
\declare@robustcommand
\declare@robustcommand@auxi
\declare@robustcommand@auxii
\declare@robustcommand@auxiii
```

This is a package-writers command, which has the same syntax as `\newcommand`, but which declares a protected command. It does this by having

```
\DeclareRobustCommand\foo
define \foo to be \protect\foo<space>,
and then use \newcommand\foo<space>.
```

Since the internal command is `\foo<space>`, when it is written to an auxiliary file, it will appear as `\foo`.

We have to be a bit cleverer if we're defining a short command, such as `\_`, in order to make sure that the auxiliary file does not include a space after the command, since `\_ a` and `\_a` aren't the same. In this case we define `\_` to be:

```
\x@protect\_ \protect\_<space>
```

which expands to:

```
\ifx\protect\@typeset@protect\else
\x@protect@\_
\fi
\protect\_<space>
```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `\_<space>`, and otherwise `\x@protect@` gobbles everything up and expands to `\protect\_`. None of that works with an active char (as we have only one character token to play with), so we resort to using the engine mechanism there. (The braces in passing `#1` there are defensive in case someone has passed more than one token!)

Note: setting `\protect` to any value other than `\relax` whilst in 'typesetting' mode will cause commands to go into an infinite loop! In particular, setting `\protect` to `\@empty` will cause `\_` to loop forever. It will also break lots of other things, such as protected `\ifmmodes` inside `\haligns`. If you really have to do such a thing, then please set `\@typeset@protect` to be `\@empty` as well. (This is what the code for `\patterns` does, for example.)

More fun with `\expandafter` and `\csname`.

```
274 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
275 \def\declare@robustcommand#1{%
276   \ifx\@undefined#1\else\ifx\relax#1\else
277     \@latex@info{Redefining \string#1}%
278     \fi\fi
279   \ifcat\noexpand~\noexpand#1%
280     \expandafter\declare@robustcommand@auxi
281   \else
282     \expandafter\declare@robustcommand@auxiii
283   \fi
284   {#1}%
285 }
```

For active chars, we cannot use an auxiliary so have to ‘pack’ everything we need in one definition. To allow for use in file names, etc., we need `\ifincsname` in the definition, so we use two steps to grab that. We also take advantage of the need for an extra auxiliary to do some expansion to give us the string we will want.

```

286 \protected\long\def\declare@robustcommand@auxi#1#2#{%
287   \expandafter\declare@robustcommand@auxii\expandafter{\string#1}#1{#2}%
288 }

289 </2ekernel>
290 <*2ekernel | latexrelease>
291 <latexrelease> \IncludeInRelease{2025/11/01}%
292 <latexrelease>   {\declare@robustcommand@auxii}{Macros without args are short}%
293 \protected\long\def\declare@robustcommand@auxii#1#2#3#4{%

```

Here we want a `\protected` command.

```

294   \let\pr@tectedrel@x\protected
295   \let\@ifdefinable\@rc@ifdefinable
296   \new@command#2#3{%
297     \ifincsname
298     \expandafter\@firstoftwo
299     \else
300     \expandafter\@secondoftwo
301     \fi
302     {#1}{#4}%
303   }%
304 }

305 </2ekernel | latexrelease>
306 <latexrelease> \EndIncludeInRelease
307 <latexrelease> \IncludeInRelease{0000/00/00}%
308 <latexrelease>   {\declare@robustcommand@auxii}{Macros without args are short}%
309 <latexrelease>
310 <latexrelease> \protected\long\def\declare@robustcommand@auxii#1#2#3#4{%
311 <latexrelease>   \ifx\l@ngrel@x\relax
312 <latexrelease>     \let\l@ngrel@x\protected
313 <latexrelease>   \else
314 <latexrelease>     \def\l@ngrel@x{\protected\long}%
315 <latexrelease>   \fi
316 <latexrelease>   \let\@ifdefinable\@rc@ifdefinable
317 <latexrelease>   \new@command#2#3{%
318 <latexrelease>     \ifincsname
319 <latexrelease>     \expandafter\@firstoftwo
320 <latexrelease>     \else
321 <latexrelease>     \expandafter\@secondoftwo
322 <latexrelease>     \fi
323 <latexrelease>     {#1}{#4}%
324 <latexrelease>   }%
325 <latexrelease> }
326 <latexrelease>
327 <latexrelease> \EndIncludeInRelease
328 <*2ekernel>

329 \protected\def\declare@robustcommand@auxiii#1{%
330   \edef\reserved@a{\string#1}%
331   \def\reserved@b{#1}%
332   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%

```



```

333 \edef#1{%
334     \ifx\reserved@a\reserved@b
335         \noexpand\x@protect
336         \noexpand#1%
337     \fi
338     \noexpand\protect
339     \expandafter\noexpand\csname
340         \expandafter\@gobble\string#1 \endcsname
341 }%
342 \let\@ifdefinable\@rc@ifdefinable
343 \expandafter\new@command\csname
344     \expandafter\@gobble\string#1 \endcsname
345 }

```

```

346 \def\x@protect#1{%
347     \ifx\protect\@typeset@protect\else
348         \x@protect#1%
349     \fi
350 }
351 \def\@x@protect#1\fi#2#3{%
352     \fi\protect#1%
353 }

```

(End of definition for `\DeclareRobustCommand` and others.)

`\@typeset@protect` We set `\@typeset@protect` to `\relax` rather than `\@empty` to make sure that the protection mechanism stops the look-ahead and expansion performed at the start of `\halign` cells.

```
354 \let\@typeset@protect\relax
```

(End of definition for `\@typeset@protect`.)

`\set@display@protect` These macros set `\protect` appropriately for typesetting or displaying.

```

\set@typeset@protect 355 \def\set@display@protect{\let\protect\string}
356 \def\set@typeset@protect{\let\protect\@typeset@protect}

```

(End of definition for `\set@display@protect` and `\set@typeset@protect`.)

`\protected@edef` The commands `\protected@edef` and `\protected@xdef` perform ‘safe’ `\edefs` and `\xdefs`, saving and restoring `\protect` appropriately. For cases where restoring `\protect` doesn’t matter, there’s an ‘unsafe’ `\unrestored@protected@xdef`, useful if you know what you’re doing!

`\protected@xdef`
`\unrestored@protected@xdef`
`\restore@protect`

```

357 \def\protected@edef{%
358     \let\@protect\protect
359     \let\protect\@unexpandable@protect
360     \afterassignment\restore@protect
361     \edef
362 }
363 \def\protected@xdef{%
364     \let\@protect\protect
365     \let\protect\@unexpandable@protect
366     \afterassignment\restore@protect
367     \xdef

```

```

368 }
369 \def\unrestored@protected@xdef{%
370   \let\protect\@unexpandable@protect
371   \xdef
372 }
373 \def\restore@protect{\let\protect\@protect}

```

(End of definition for \protected@edef and others.)

\protect The normal meaning of \protect

```

374 \set@typeset@protect

```

(End of definition for \protect.)

\MakeRobust This macro makes an existing fragile macro robust, but only if it hasn't been robust in the past, i.e., it checks for the existence of the macro `\<name>`_␣ and if that exists it assumes that `\<name>` is already robust. In that case either undefine the inner macro first or use `\DeclareRobustCommand` to define it in a robust way directly. We could probably test the top-level definition to have the right kind of structure, but this is somewhat problematical as we then have to distinguish between `\long` macros and others and also take into account that sometimes the top-level is deliberately done manually (like with `\begin`).

The macro firstly checks if the control sequence in question exists at all.

```

375 \</2ekernel>
376 \<latexrelease>\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%
377 \<*2ekernel | latexrelease>
378 \def\MakeRobust#1{%
379   \count@=\escapechar
380   \escapechar='\\
381   \@ifundefined{\expandafter\@gobble\string#1}{%
382     \@latex@error{Command '\string#1' undefined.%
383     \MessageBreak There is nothing here to make robust}%
384     \@eha
385   }%

```

Then we check if the macro is already robust. We do this by testing if the internal name for a robust macro is defined, namely `\foo`_␣. If it is already defined do nothing, otherwise set `\foo`_␣ equal to `\foo` and redefine `\foo` so that it acts like a macro defined with `\DeclareRobustCommand`. We use `\@kernel@rename@newcommand` to copy `\foo` over to `\foo`_␣, including a possible default optional argument.

```

386   {%
387     \@ifundefined{\expandafter\@gobble\string#1\space}%
388     {%
389       \expandafter\@kernel@rename@newcommand
390       \csname\expandafter\@gobble\string#1\space\endcsname
391       #1%
392       \edef\reserved@a{\string#1}%
393       \def\reserved@b{#1}%
394       \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
395       \xdef#1{%
396         \ifx\reserved@a\reserved@b
397           \noexpand\x@protect\noexpand#1%
398         \fi
399         \noexpand\protect\expandafter\noexpand

```

```

400     \csname\expandafter\@gobble\string#1\space\endcsname}%
401   }%
402   {\@latex@info{Command '\string#1' is already robust}}%
403 }%
404 \escapechar=\count@
405 }%

```

This macro renames a command, possibly with an optional argument (defined with `\newcommand`) from #2 to #1, by renaming the internal macro `\\#2` to `\\#1` and defining `\\#1` appropriately, then undefining `\\#2` and `\\#2`. The `\afterassignment` trick is to make both definitions in `\@copy@newcommand` global (which are local by default).

In case the macro was defined with `\newcommand` and an optional argument, to replicate exactly the behaviour of `\DeclareRobustCommand` we have to move also the internal `\\foo` to `\\foo_`. In that case, #1 will be a parameterless macro (`\robust@command@chk@safe` checks that), and `\@if@newcommand` will return true (both defined below in this file). If so, we can use `\@copy@newcommand` rather than plain `\let` to copy the command over. `\@kernel@rename@newcommand` does this test and carries out the renaming.

```

406 \def\@kernel@rename@newcommand#1#2{%
407   \robust@command@chk@safe#2%
408   {\@if@newcommand#2%
409     {\afterassignment\global
410       \global\@copy@newcommand#1#2%
411       \global\let#2\@undefined
412       \global\expandafter\let\csname\string#2\endcsname\@undefined}%
413     {\global\let#1=#2}}%
414   {\global\let#1=#2}}

415 \</2ekernel|latexrelease>
416 \<latexrelease>\EndIncludeInRelease
417 %
418 \<latexrelease>\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
419 \<latexrelease>\def\MakeRobust#1{%
420 \<latexrelease>   \@ifundefined{\expandafter\@gobble\string#1}{%
421 \<latexrelease>     \@latex@error{The control sequence '\string#1' is undefined!%
422 \<latexrelease>       \MessageBreak There is nothing here to make robust}%
423 \<latexrelease>     \@eha
424 \<latexrelease>   }%
425 \<latexrelease>   {%
426 \<latexrelease>     \@ifundefined{\expandafter\@gobble\string#1\space}%
427 \<latexrelease>     {%
428 \<latexrelease>       \global\expandafter\let\csname
429 \<latexrelease>         \expandafter\@gobble\string#1\space\endcsname=#1%
430 \<latexrelease>       \edef\reserved@a{\string#1}%
431 \<latexrelease>       \def\reserved@b{#1}%
432 \<latexrelease>       \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
433 \<latexrelease>       \xdef#1{%
434 \<latexrelease>         \ifx\reserved@a\reserved@b
435 \<latexrelease>           \noexpand\x@protect\noexpand#1%
436 \<latexrelease>         \fi
437 \<latexrelease>         \noexpand\protect\expandafter\noexpand
438 \<latexrelease>         \csname\expandafter\@gobble\string#1\space\endcsname}%
439 \<latexrelease>       }%

```

```

440 <latexrelease>    {\@latex@info{The control sequence ‘\string#1’ is already robust}}}%
441 <latexrelease>    }%
442 <latexrelease>}%
443 <latexrelease>\let\@kernel@rename@newcommand\@undefined
444 <latexrelease>\EndIncludeInRelease
445 %
446 <latexrelease>\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
447 <latexrelease>\def\MakeRobust#1{%
448 <latexrelease>    \ifundefined{\expandafter\@gobble\string#1}{%
449 <latexrelease>        \@latex@error{The control sequence ‘\string#1’ is undefined!%
450 <latexrelease>        \MessageBreak There is nothing here to make robust}%
451 <latexrelease>        \@eha
452 <latexrelease>    }%
453 <latexrelease>    {%
454 <latexrelease>        \ifundefined{\expandafter\@gobble\string#1\space}%
455 <latexrelease>        {%
456 <latexrelease>            \expandafter\let\csname
457 <latexrelease>            \expandafter\@gobble\string#1\space\endcsname=#1%
458 <latexrelease>            \edef\reserved@a{\string#1}%
459 <latexrelease>            \def\reserved@b{#1}%
460 <latexrelease>            \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
461 <latexrelease>            \edef#1{%
462 <latexrelease>                \ifx\reserved@a\reserved@b
463 <latexrelease>                    \noexpand\x@protect\noexpand#1%
464 <latexrelease>                \fi
465 <latexrelease>                \noexpand\protect\expandafter\noexpand
466 <latexrelease>                \csname\expandafter\@gobble\string#1\space\endcsname}%
467 <latexrelease>            }%
468 <latexrelease>        {\@latex@info{The control sequence ‘\string#1’ is already robust}}}%
469 <latexrelease>    }%
470 <latexrelease>}%
471 <latexrelease>\let\@kernel@rename@newcommand\@undefined
472 <latexrelease>\EndIncludeInRelease
473 %
474 <latexrelease>\IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
475 <latexrelease>\let\MakeRobust\@undefined
476 <latexrelease>\let\@kernel@rename@newcommand\@undefined
477 <latexrelease>\EndIncludeInRelease
478 <*2ekernel>

```

(End of definition for \MakeRobust and \@kernel@rename@newcommand.)

\kernel@make@fragile The opposite of **\MakeRobust** except that it doesn't do many checks as it is internal to the kernel. Why does one want such a thing? Only for compatibility reasons if **latexrelease** requests a rollback of the kernel. For this reason we pretend that this command existed in all earlier versions of L^AT_EX i.e., we are not rolling it back since we need it precisely then. But we have to get it into the **latexrelease** file so that a roll forward is possible too.

```

479 </2ekernel>
480 <*2ekernel | latexrelease>
481 <latexrelease>\IncludeInRelease{2020/10/01}%
482 <latexrelease>        {\kernel@make@fragile}{Undo robustness}%
483 <def\kernel@make@fragile#1{%
484     \ifundefined{\expandafter\@gobble\string#1\space}%

```

If not robust do nothing.

```

485     {}%
Otherwise copy \foo_ back to \foo. Then use \@kernel@rename@newcommand to check
and copy \foo_ back to \foo in case the command has an optional argument. If so,
also undefine \foo_, and at the end undefine \foo_.
486     {%
487         \global\expandafter\let\expandafter #1\csname
488             \expandafter@gobble\string#1\space\endcsname
489         \expandafter\@kernel@rename@newcommand
490         \csname\expandafter@gobble\string#1\expandafter\endcsname
491         \csname\expandafter@gobble\string#1\space\endcsname
492         \global\expandafter\let\csname
493             \expandafter@gobble\string#1\space\endcsname\undefined
494     }%
495 }
496 \<latexrelease>\EndIncludeInRelease
497 %
498 \<latexrelease>\IncludeInRelease{0000/00/00}%
499 \<latexrelease>          {\kernel@make@fragile}{Undo robustness}%
500 \<latexrelease>\def\kernel@make@fragile#1{%
501 \<latexrelease>  \@ifundefined{\expandafter@gobble\string#1\space}%
502 \<latexrelease>    {}%
503 \<latexrelease>    {%
504 \<latexrelease>      \global\expandafter\let\expandafter #1\csname
505 \<latexrelease>      \expandafter@gobble\string#1\space\endcsname
506 \<latexrelease>      \global\expandafter\let\csname
507 \<latexrelease>      \expandafter@gobble\string#1\space\endcsname\undefined
508 \<latexrelease>    }%
509 \<latexrelease>}
510 \<latexrelease>\EndIncludeInRelease
511 \</2ekernel | latexrelease>
512 \<*2ekernel>

```

(End of definition for \kernel@make@fragile.)

1.5 Acting on robust commands

```

513 \</2ekernel>
514 \<latexrelease>\IncludeInRelease{2020-10-01}{\robust@command@act}
515 \<latexrelease>  {Add \robust@command@act}%
516 \<*2ekernel | latexrelease>

```

With most document level commands being robust now there is more of a requirement to have a standard way of aliasing (or copying) a command to a new name, for example to save an original definition before changing a command. `\DeclareCommandCopy` is analogous to $\text{\TeX}$ ’s `\let`, except that it copes with the different types of robust commands defined by $\text{\LaTeX}$ ’s mechanisms.

A couple of “types of robustness” are defined by the $\text{\LaTeX} 2_\epsilon$ kernel, namely robust commands defined with `\DeclareRobustCommand` and commands with optional arguments defined with `\newcommand`. However there are other types of robust commands that are frequently used, which are not defined in the $\text{\LaTeX} 2_\epsilon$ kernel, like commands defined with `xparse`’s `\NewDocumentCommand` and `etoolbox`’s `\newrobustcmd`.

In this section we will define a generic extensible machinery to act on robust commands. This code will then be used to test if a command is robust, considered the different types of robustness, and then either copy that definition, if `\DeclareCommandCopy` (or similar) is used, or show the definition of the command, if `\ShowCommand` is used.

`\robust@command@act` The looping machinery is generic and knows nothing about what is to be done for each case. The syntax of the main macro `\robust@command@act` is:

```
\robust@command@act⟨action-list⟩⟨robust-cmd⟩
⟨fallback-action⟩⟨act-arg⟩
```

`⟨action-list⟩` is a token list of the form:

```
{⟨if-type-1⟩ ⟨act-type-1⟩}
{⟨if-type-2⟩ ⟨act-type-2⟩}
...
```

`\robust@command@act` will iterate over the `⟨action-list⟩`, evaluating each `⟨if-type-n⟩` `⟨robust-cmd⟩` `{⟨true⟩}{⟨false⟩}`. If the `⟨if-type-n⟩` conditional returns `⟨true⟩`, then `⟨act-type-n⟩⟨act-arg⟩` is executed, and the loop ends. If the conditional returns `⟨false⟩`, then `⟨if-type-n + 1⟩` is executed in the same way, until either one of the conditionals return `⟨true⟩`, or the end of the `⟨action-list⟩` is reached. If the end is reached, then `⟨fallback-action⟩⟨act-arg⟩` is executed before `\robust@command@act` exits.

`\robust@command@act` will start by using `\robust@command@act@chk@args` to check if the `⟨robust-cmd⟩` (#2) is a parameterless (possibly `\protected`) macro. If it is not, the command is not a robust command: these always start with a parameterless user-level macro; in that case, `\robust@command@act@end` is used to short-circuit the process and do the `⟨fallback-action⟩` (#3). This first test is necessary because later on we need to be able to expand the `⟨robust-cmd⟩` without the risk of it Breaking Badly, and as a bonus, this speeds up the process in case we used `\NewCommandCopy` in a “normal” macro.

```
517 \long\def\robust@command@act#1#2#3#4{%
518   \robust@command@chk@safe#2%
519   {\expandafter\robust@command@act@loop
520    \expandafter#2%
521    #1{\@nnil\@nnil}%
522    \robust@command@act@end}%
523   {\robust@command@act@end}%
524   {#3}{#4}}%
```

If `\robust@command@act@chk@args` branched to false, then `\robust@command@act@loop` will loop over the list of items in the `⟨action-list⟩` (#1), and process each item as described earlier. If the `⟨if-type-n⟩` command expands to `⟨true⟩` then `\robust@command@act@do` is used to execute `⟨act-type-n⟩` on the `⟨act-arg⟩`, otherwise the loop resumes with the next item.

```
525 \long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
526 \long\def\robust@command@act@loop@aux#1#2#3{%
527   \ifx\@nnil#2%
528   \else
529     #2{#1}%
530     {\robust@command@act@do{#3}}%
531     {\expandafter\robust@command@act@loop\expandafter#1}%
532   \fi}
```

```

533 \long\def\robust@command@act@do#1%
534   \fi#2%
535   \robust@command@act@end#3#4{%
536   \fi
537   #1#4}

```

If the end is reached and no action was taken, then do $\langle fallback-action \rangle \langle act-arg \rangle$.

```

\robust@command@act@end 538 \long\def\robust@command@act@end#1#2{#1#2}

```

```

\robust@command@chk@safe 539 \long\def\robust@command@chk@safe#1{%
\robust@command@act@chk@args 540   \begingroup
541   \escapechar='\\
542   \expandafter\endgroup\expandafter
543   \robust@command@act@chk@args\meaning#1:->\@nil}
544 \def\robust@command@act@chk@args#1:->#2\@nil{%
545   \@expl@str@if@eq@nnTF{#1}{macro}%
546   {\@firstoftwo}%
547   {\@expl@str@if@eq@nnTF{#1}{\protected macro}%
548   {\@firstoftwo}%
549   {\@secondoftwo}}}

550 </2ekernel | latexrelease>
551 <latexrelease>\EndIncludeInRelease
552 <latexrelease>\IncludeInRelease{0000-00-00}{\robust@command@act}
553 <latexrelease> {Add \robust@command@act}%
554 <latexrelease>\let\robust@command@act\@undefined
555 <latexrelease>\let\robust@command@act@loop\@undefined
556 <latexrelease>\let\robust@command@act@loop@aux\@undefined
557 <latexrelease>\let\robust@command@act@do\@undefined
558 <latexrelease>\let\robust@command@act@end\@undefined
559 <latexrelease>\let\robust@command@chk@safe\@undefined
560 <latexrelease>\let\robust@command@act@chk@args\@undefined
561 <latexrelease>\EndIncludeInRelease
562 <*2ekernel>

```

(End of definition for `\robust@command@act` and others.)

1.5.1 Copying robust commands

```

563 </2ekernel>
564 <latexrelease>\IncludeInRelease{2020-10-01}{\DeclareCommandCopy}
565 <latexrelease> {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
566 <*2ekernel | latexrelease>

```

`\NewCommandCopy` `\NewCommandCopy` starts by checking if `#1` is already defined, and raises an error if so, otherwise the definition is carried out. `\RenewCommandCopy` does (almost) the opposite. `\DeclareCommandCopy` If the command is *not* defined, then an error is raised. But the definition is carried out anyhow, so the behaviour is consistent with `\renewcommand`.

A `\ProvideCommandCopy` isn't defined because it's not reasonably useful. `\provide...` commands mean “define this if there's no other definition”, but copying a command (usually) implies that the command being copied is defined, so `\ProvideCommandCopy` doesn't make a lot of sense. But more importantly, the most common use case of copying a command is to redefine it later, while preserving the old definition, as in:

```

\ProvideCommandCopy \A \B
\renewcommand \B { ... \A ... }

```

then, if \A is already defined the first line is skipped, and in this case \B won't work as expected.

The three versions call the internal `\declare@commandcopy` with the proper action. `\@firstofone` will carry out the copy. The only case when the copy is not made is the `<false>` case for `\NewCommandCopy`, in which the command already exists and the definition is aborted.

```

567 \def\NewCommandCopy{%
568   \declare@commandcopy
569     {\@firstofone}%
570     {\@firstoftwo\@notdefinable}}
571 \def\RenewCommandCopy{%
572   \declare@commandcopy
573     {\@latexerror{Command \backslashchar\reserved@a\space undefined}\@ehc
574     \@firstofone}%
575     {\@firstofone}}
576 \def\DeclareCommandCopy{%
577   \declare@commandcopy
578     {\@firstofone}%
579     {\@firstofone}}

```

Start by checking if the command is already defined. The proper action is taken by each specific command above. If all's good, then `\robust@command@act` is called with the proper arguments as described earlier, with `\@declarecommandcopylisthook` as the `<action-list>` and `\declare@commandcopy@let` as the `<fallback-action>`.

```

\declare@commandcopy
\declare@commandcopy@do
580 \long\def\declare@commandcopy#1#2#3#4{%
581   \edef\reserved@a{\expl@cs@to@str@N#3}%
582   \@ifundefined\reserved@a{#1}{#2}%
583   {\declare@commandcopy@do{#3}{#4}}}
584 \long\def\declare@commandcopy@do#1#2{%
585   \robust@command@act
586   \@declarecommandcopylisthook#2%
587   \declare@commandcopy@let{#1#2}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

\@declarecommandcopylisthook
588 \def\@declarecommandcopylisthook{%
589   {\@if@DeclareRobustCommand \@copy@DeclareRobustCommand}%
590   {\@if@newcommand \@copy@newcommand}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

591 \long\def\declare@commandcopy@let#1#2{\let#1=#2\relax}
\declare@commandcopy@let

```

Now the rollback code.

```

592 </2ekernel | latexrelease>
593 <latexrelease>\EndIncludeInRelease
594 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
595 <latexrelease> {Undefine \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
596 <latexrelease>\let\NewCommandCopy\@undefined

```



```

597 <latexrelease>\let\RenewCommandCopy\@undefined
598 <latexrelease>\let\DeclareCommandCopy\@undefined
599 <latexrelease>\let\declare@commandcopy\@undefined
600 <latexrelease>\let\@declarecommandcopylisthook\@undefined
601 <latexrelease>\let\declare@commandcopy@let\@undefined
602 <latexrelease>\EndIncludeInRelease
603 <*2ekernel>

(End of definition for \NewCommandCopy and others.)

604 </2ekernel>
605 <latexrelease>\IncludeInRelease{2023-06-01}{\DeclareEnvironmentCopy}
606 <latexrelease> {Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}%
607 <*2ekernel | latexrelease>

```

`\NewEnvironmentCopy` If `\#1` or `\end#1` already exist one gets an error message talking about the problematical command (not the environment). The remainder of the $\text{\LaTeX}$ run is probably badly broken and it is unlikely that continuing it gives reasonable results.

```

608 \def\NewEnvironmentCopy{%
609   \declare@environmentcopy
610     {\@firstofone}%
611     {\@firstoftwo\@notdefinable}}
612 \def\RenewEnvironmentCopy{%
613   \declare@environmentcopy
614     {\@latex@error{Environment \reserved@a\space undefined}\@ehc
615     \@firstofone}%
616     {\@firstofone}}
617 \def\DeclareEnvironmentCopy{%
618   \declare@environmentcopy
619     {\@firstofone}%
620     {\@firstofone}}
621 \long\def\declare@environmentcopy#1#2#3#4{%
622   \edef\reserved@a{\@ifundefined{#3}{end#3}{#3}}%
623   \@ifundefined\reserved@a
624     {\def\reserved@a{#3}#1}%
625     {\def\reserved@a{#3}#2}%
626     {\ExpandArgs{cc}\declare@commandcopy@do{#3}{#4}%
627     \ExpandArgs{cc}\declare@commandcopy@do{end#3}{end#4}}}

```

Now the rollback code.

```

628 </2ekernel | latexrelease>
629 <latexrelease>\EndIncludeInRelease
630 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareEnvironmentCopy}
631 <latexrelease> {Undefine \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}
632 <latexrelease>\let\NewEnvironmentCopy\@undefined
633 <latexrelease>\let\RenewEnvironmentCopy\@undefined
634 <latexrelease>\let\DeclareEnvironmentCopy\@undefined
635 <latexrelease>\EndIncludeInRelease
636 <*2ekernel>

```

(End of definition for `\NewEnvironmentCopy`, `\RenewEnvironmentCopy`, and `\DeclareEnvironmentCopy`.)

1.5.2 Showing robust commands

`\ShowCommand` Most of the machinery defined for `\NewCommandCopy` can be used to show the definition of a robust command, in a similar fashion to `\texdef`. The difference is that after the command is detected to have a given type of robustness, rather than making a copy, we use a separate routine to show its definition.

With all the machinery in place, `\ShowCommand` itself is quite simple: we use `\robust@command@act` to iterate through the `\@showcommandlisthook` list, and if nothing is found, fallback to `\show`.

```

637 </2ekernel>
638 <latexrelease>\IncludeInRelease{2020-10-01}{\ShowCommand}%
639 <latexrelease> {Add \ShowCommand}%
640 <*2ekernel | latexrelease>

641 \long\def\ShowCommand#1{%
642   \robust@command@act
643   \@showcommandlisthook#1%
644   \show#1}

```

`\@showcommandlisthook` The initial definition of `\@showcommandlisthook` contains the same tests as used for copying, but `\@show@...` commands instead of `\@copy@...`. Same as before, it is initialized to cope with `\DeclareRobustCommand` and `\newcommand` with optional arguments.

```

645 \def\@showcommandlisthook{%
646   {\@if@DeclareRobustCommand \@show@DeclareRobustCommand}%
647   {\@if@newcommand \@show@newcommand}}

```

Now the rollback code.

```

648 </2ekernel | latexrelease>
649 <latexrelease>\EndIncludeInRelease
650 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowCommand}
651 <latexrelease> {Undefine \ShowCommand}%
652 <latexrelease>\let\ShowCommand\@undefined
653 <latexrelease>\let\@showcommandlisthook\@undefined
654 <latexrelease>\EndIncludeInRelease
655 <*2ekernel>

```

(End of definition for \ShowCommand and \@showcommandlisthook.)

```

656 </2ekernel>
657 <latexrelease>\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}
658 <latexrelease> {Add \@if@DeclareRobustCommand, \@if@newcommand,
659 <latexrelease>       \@copy@DeclareRobustCommand, \@copy@newcommand,
660 <latexrelease>       \@show@DeclareRobustCommand, \@show@newcommand}%
661 <*2ekernel | latexrelease>

```

1.5.3 Commands defined with \DeclareRobustCommand

`\@if@DeclareRobustCommand` Now that we provided a generic way to copy one macro to another, we need to define a way to check if a command is one of L^AT_EX 2_ε's robust types. These tests are heavily based on Heiko's `\LetLtxMacro`, but chopped into separate macros.

The command `\@if@DeclareRobustCommand` checks if a command `\cmd` was defined by `\DeclareRobustCommand`. The test returns true if the expansion of `\cmd` is exactly `\protect\cmd_`.

```

662 \long\def\@if@DeclareRobustCommand#1{%

```

```

663 \begingroup
664 \escapechar='\\
665 \edef\reserved@a{\string#1}%
666 \edef\reserved@b{\detokenize{#1}}%
667 \xdef\@gtempa{%
668     \ifx\reserved@a\reserved@b
669         \noexpand\x@protect
670         \noexpand#1%
671     \fi
672     \noexpand\protect
673     \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
674 \endgroup
675 \ifx\@gtempa#1\relax
676     \expandafter\@firstoftwo
677 \else
678     \expandafter\@secondoftwo
679 \fi}

```

If a command was defined by `\DeclareRobustCommand` (that is, `\@if@DeclareRobustCommand` returns true), then to make a copy of `\cmd` into `\foo` we define the latter such that it expands to `\protect\foo`, then make `\foo` equal to `\cmd`.

There is one detail we need to take care of: if a command was defined with `\DeclareRobustCommand` it may still have an optional argument, in which case there is one more macro layer before the actual definition of the command. We use `\@if@newcommand` to check that and `\@copy@newcommand` to do the copying.

```

680 \long\def\@copy@DeclareRobustCommand#1#2{%
681     \begingroup
682     \escapechar='\\
683     \edef\reserved@a{\string#1}%
684     \edef\reserved@b{\detokenize{#1}}%
685     \edef\reserved@a{%
686     \endgroup
687     \def\noexpand#1{%
688         \ifx\reserved@a\reserved@b
689             \noexpand\x@protect
690             \noexpand#1%
691         \fi
692         \noexpand\protect
693         \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname}%
694     \noexpand\copy@kernel@robust@command
695     \expandafter\noexpand\csname\@expl@cs@to@str@@N#1 \endcsname
696     \expandafter\noexpand\csname\@expl@cs@to@str@@N#2 \endcsname}%
697     \reserved@a}
698 \long\def\copy@kernel@robust@command#1#2{%
699     \robust@command@chk@safe#2%
700     {\@if@newcommand#2%
701         {\@copy@newcommand}%
702         {\declare@commandcopy@let}}
703     {\declare@commandcopy@let}%
704     #1#2}

```

Showing the command is pretty simple. This command prints the top-level expansion as $\TeX$'s `\show` would, but with robust macro: rather than just `macro:`, then

```

\@show@DeclareRobustCommand
\show@kernel@robust@command
\@show@macro

```

a blank line and then `\show` the inner command. For a macro defined with, say, `\DeclareRobustCommand\foo[1]{bar}`, it will print:

```
> \foo=robust macro:
->\protect \foo .

> \foo =\long macro:
#1->bar.
```

If the inner command is defined with an optional argument, then `\@show@newcommand` is also used.

The value of `\escapechar` is deliberately not enforced, so `\ShowCommand` behaves more like `\show`.

```
705 \long\def\@show@DeclareRobustCommand#1{%
706   \typeout{> \string#1=robust macro:}%
707   \typeout{->\@expl@cs@replacement@spec@@N#1.^~J}%
708   \expandafter\show@kernel@robust@command
709     \csname\@expl@cs@to@str@@N#1 \endcsname}
710 \long\def\show@kernel@robust@command#1{%
711   \robust@command@chk@safe#1%
712     {\@if@newcommand#1%
713       {\@show@newcommand}%
714       {\@show@macro}}%
715     {\@show@macro}%
716   #1}
717 \let\@show@macro\show
```

(End of definition for `\@if@DeclareRobustCommand` and others.)

1.5.4 Commands defined with `\newcommand` (with optional argument)

`\@if@newcommand` A command `\cmd` (or `\cmdL`, if it was defined with `\DeclareRobustCommand`) with an optional argument will expand to `\@protected@testopt\cmd\cmd{<opt>}`. To check that we look at the first three tokens in the expansion of `\cmd`, and return true or false accordingly.

This test *requires* that the command be a parameterless macro, otherwise it will not work (and probably break). This is ensured with `\robust@command@chk@safe` before calling `\@if@newcommand`.

```
718 \long\def\@if@newcommand#1{%
719   \edef\reserved@a{%
720     \noexpand\@protected@testopt
721     \noexpand#1%
722     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname}%
723   \edef\reserved@b{%
724     \unexpanded\expandafter\expandafter\expandafter
725       {\expandafter\@carcube#1{-}{-}{-}\@nil}}%
726   \ifx\reserved@a\reserved@b
727     \expandafter\@firstoftwo
728   \else
729     \expandafter\@secondoftwo
730   \fi}
```

Then, if a command `\cmd` takes an optional argument, we copy it to `\foo` by defining the latter to expand to `\@protected@testopt\foo\foo{<opt>}`.

```
\@copy@newcommand
731 \long\def\@copy@newcommand#1#2{%
732   \edef#1{\noexpand\@protected@testopt
733     \noexpand#1%
734     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@N#1\endcsname
735     \unexpanded\expandafter\expandafter\expandafter
736       {\expandafter\@gobblethree#2}}%
737   \expandafter
738   \let\csname\@backslashchar\@expl@cs@to@str@N#1\expandafter\endcsname
739     \csname\@backslashchar\@expl@cs@to@str@N#2\endcsname}
```

`\@show@newcommand`
`\@show@newcommand@aux`

A command being `\shown` here is guaranteed to have an optional argument. Start by showing the top-level expansion of the command (using `\typeout` to avoid TeX asking for interaction and extra context lines), then call `\@show@newcommand@aux` with the internal command, which contains the actual definition, and with the expansion of the command to extract the default value of the optional argument.

```
740 \long\def\@show@newcommand#1{%
741   \typeout{> \string#1=robust macro:}%
742   \typeout{->\@expl@cs@replacement@spec@N#1.^^J}%
743   \expandafter\@show@newcommand@aux
744     \csname\@backslashchar\@expl@cs@to@str@N#1\expandafter\endcsname
745     \expandafter{#1}\@show@tokens}
```

For a macro defined with, say, `\newcommand\foo[1][opt]{bar}`, it will print:

```
> \foo=robust macro:
->\@protected@testopt \foo \foo {opt}.

> \foo=\long macro:
> default #1=opt.
[#1]->bar.
```

If the command was defined with `\DeclareRobustCommand`, then another pair of lines show the top-level expansion `\protect_\foo_\_`.

```
746 \long\def\@show@newcommand@aux#1#2#3{%
747   \typeout{> \string#1=\@expl@cs@prefix@spec@N#1macro:}%
748   #3{default \string##1=\expandafter\detokenize\@gobblethree#2.^^J%
749     \@expl@cs@parameter@spec@N#1->\@expl@cs@replacement@spec@N#1}}
```

This macro prints the contents of the token list (macro) `#1` using `\showtokens`. The `\expandafter` gymnastics ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

```
\@show@tokens
750 \long\def\@show@tokens#1{%
751   \edef\reserved@a{#1}%
752   \showtokens\expandafter
753     \expandafter\expandafter{\expandafter\reserved@a}}
```

Now the rollback code.

```
754 </2ekernel | latexrelease>
755 <latexrelease>\EndIncludeInRelease
756 <latexrelease>\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
757 <latexrelease> {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
```

```

758 <latexrelease> \copy@DeclareRobustCommand, \copy@newcommand,
759 <latexrelease> \show@DeclareRobustCommand, \show@newcommand}%
760 <latexrelease>\let\@if@DeclareRobustCommand\@undefined
761 <latexrelease>\let\@copy@DeclareRobustCommand\@undefined
762 <latexrelease>\let\@show@DeclareRobustCommand\@undefined
763 <latexrelease>\let\@if@newcommand\@undefined
764 <latexrelease>\let\@copy@newcommand\@undefined
765 <latexrelease>\let\@show@newcommand\@undefined
766 %
767 <latexrelease>\let\copy@kernel@robust@command\@undefined
768 <latexrelease>\let\show@kernel@robust@command\@undefined
769 <latexrelease>\let\@show@newcommand@aux\@undefined
770 <latexrelease>\EndIncludeInRelease
771 <*2ekernel>

```

(End of definition for \@if@newcommand and others.)

1.5.5 Showing environments

\ShowEnvironment

```

772 </2ekernel>
773 <latexrelease>\IncludeInRelease{2023-06-01}{\ShowEnvironment}
774 <latexrelease> {Add \ShowEnvironment}%
775 <*2ekernel | latexrelease>

```

\ShowEnvironment is quite similar to \ShowCommand. We will pass the environment `<env>` around as the macro `\env`, because `\robust@command@act` expects a single token.

```

776 \def\ShowEnvironment#1{%
777   \expandafter\@show@environment\csname #1\endcsname}
778 \long\def\@show@environment#1{%
779   \robust@command@act
780   \@showenvironmentlisthook#1%
781   \@show@normalenv#1}

```

This is similar to `\@showcommandlisthook`, but uses the dedicated versions for environments.

\@showenvironmentlisthook

```

782 \def\@showenvironmentlisthook{%
783   {\@if@DeclareRobustCommand \show@DeclareRobustCommand@env}%
784   {\@if@newcommand \show@newcommand@env}}

```

These are similar to the command versions below, except they say “environment” and call `\@show@environment@end` to print the `\end` part.

\show@newcommand@env

\show@DeclareRobustCommand@env

```

785 \long\def\@show@newcommand@env#1{%
786   \@show@environment@begin#1%
787   \expandafter\@show@newcommand@aux
788   \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
789   \expandafter{#1}\@show@typeout
790   \@show@environment@end#1}
791 \long\def\@show@DeclareRobustCommand@env#1{%
792   \@show@environment@begin#1%
793   \begingroup
794   \let\@show@tokens\@show@typeout
795   \let\@show@macro\@show@nonstop
796   \expandafter\show@kernel@robust@command

```

```

797 \csname\@expl@cs@to@str@N#1 \endcsname
798 \endgroup
799 \@show@environment@end#1}
800 \long\def\@show@environment@begin#1{%
801 \typeout{> \string\begin{\@expl@cs@to@str@N#1}=environment:}%
802 \typeout{\@expl@cs@parameter@spec@N#1->%
803 \@expl@cs@replacement@spec@N#1.^^J}}

```

A “normal” environment is straightforward. `\@show@environment@end` needs to check if the `\end` part is defined and show it accordingly, otherwise the output would show gibberish.

`\@show@normalenv`
`\@show@environment@end`

```

804 \long\def\@show@normalenv#1{%
805 \show@environment@begin#1%
806 \show@environment@end#1}
807 \long\def\@show@environment@end#1{%
808 \expandafter\@show@environment@end@aux
809 \csname end\@expl@cs@to@str@N#1\endcsname#1}
810 \long\def\@show@environment@end@aux#1#2{%
811 \@show@tokens{\string\end{\@expl@cs@to@str@N#2}%
812 \ifx\relax#1=undefined%
813 \else:^^J\@expl@cs@parameter@spec@N#1->%
814 \@expl@cs@replacement@spec@N#1%
815 \fi}}

```

And here some auxiliaries:

`\@show@nonstop`
`\@show@typeout`

`\@show@nonstop` same output as `\show`, but doesn’t stop for interaction;

`\@show@typeout` same output as `\showtokens`, but doesn’t stop for interaction.

```

816 \def\@show@nonstop#1{%
817 \typeout{> \string#1=\@expl@cs@prefix@spec@N#1macro:^^J%
818 \@expl@cs@parameter@spec@N#1->\@expl@cs@replacement@spec@N#1.}}
819 \def\@show@typeout#1{\typeout{> #1.^^J}}

```

Now the rollback code.

```

820 </2ekernel | latexrelease>
821 <latexrelease>\EndIncludeInRelease
822 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowEnvironment}
823 <latexrelease> {Undefine \ShowEnvironment}%
824 <latexrelease>\let\ShowEnvironment\@undefined
825 <latexrelease>\EndIncludeInRelease
826 <*2ekernel>

```

(End of definition for \ShowEnvironment and others.)

1.6 Internal defining commands

These commands are used internally to define other L^AT_EX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```

827 </2ekernel>
828 <latexrelease>\IncludeInRelease{2018-04-01}{\@ifundefined}
829 <latexrelease>\Leave commands undefined in \@ifundefined}%
830 <*2ekernel | latexrelease>

```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```

831 \def\@ifundefined#1{%
832   \ifcsname#1\endcsname\@ifundefined@i\else\@ifundefined@ii\fi{#1}}
833 \long\def\@ifundefined@i#1\fi#2{\fi
834   \expandafter\ifx\csname #2\endcsname\relax
835     \@ifundefined@ii
836   \fi
837   \@secondoftwo}
838 \long\def\@ifundefined@ii\fi#1#2#3{\fi #2}

```

Now test of engine.

```
839 \ifx\numexpr\@undefined
```

Classic version (should not be needed as etex is assumed).

```

840 \def\@ifundefined#1{%
841   \expandafter\ifx\csname#1\endcsname\relax
842     \expandafter\@firstoftwo
843   \else
844     \expandafter\@secondoftwo
845   \fi}
846 \else\ifx\directlua\@undefined

```

Use the `\ifcsname` defined above.

```
847 \else
```

Optimised version for LuaTeX, using `\lastnamedcs`

```

848 \def\@ifundefined#1{%
849   \ifcsname#1\endcsname
850     \expandafter\ifx\lastnamedcs\relax\else\@ifundefined@i\fi
851   \fi
852   \@firstoftwo}
853 \long\def\@ifundefined@i#1#2#3#4#5{#1#2#5}
854 \fi
855 \fi
856 </2ekernel | latexrelease>
857 <latexrelease>\EndIncludeInRelease
858 <latexrelease>\IncludeInRelease{0000-00-00}{\@ifundefined}
859 <latexrelease>\Leave commands undefined in \@ifundefined}%
860 <latexrelease>\def\@ifundefined#1{%
861 <latexrelease>   \expandafter\ifx\csname#1\endcsname\relax
862 <latexrelease>     \expandafter\@firstoftwo
863 <latexrelease>   \else
864 <latexrelease>     \expandafter\@secondoftwo
865 <latexrelease>   \fi}
866 <latexrelease>\EndIncludeInRelease
867 <*2ekernel>

```

(End of definition for \@ifundefined.)

`\@qend` The following define `\@qend` and `\@qrelax` to be the strings ‘end’ and ‘relax’ with the characters `\catcoded 12`.

```

868 \edef\@qend{\expandafter\@cdr\string\end\@nil}
869 \edef\@qrelax{\expandafter\@cdr\string\relax\@nil}

```


(End of definition for \@qend and \@qrelax.)

\@ifnextchar \@ifnextchar peeks at the following character and compares it with its first argument. If both are the same it executes its second argument, otherwise its third.

```
870 \long\def\@ifnextchar#1#2#3{%  
871   \let\reserved@d=#1%  
872   \def\reserved@a{#2}%  
873   \def\reserved@b{#3}%  
874   \futurelet\@let@token\@ifnch}
```

(End of definition for \@ifnextchar.)

\kernel@ifnextchar This macro is the kernel version of \@ifnextchar which is used in a couple of places to prevent the AMS variant from being used since in some places this produced chaos. For example, if an `fd` file is loaded in a random place then the optional argument to `\ProvidesFile` could get printed there instead of being written only in the log file. This happened when there was a space or a newline between the mandatory and optional arguments! It should really be fixed in the `amsmath` package one day, but...

Note that there may be other places in the kernel where this version should be used rather than the original, but variable, version.

```
875 \let\kernel@ifnextchar\@ifnextchar
```

(End of definition for \kernel@ifnextchar.)

\@ifnch \@ifnch is a tricky macro to skip any space tokens that may appear before the character in question. If it encounters a space token, it calls `xifnch`.

```
876 \def\@ifnch{%  
877   \ifx\@let@token\@sptoken  
878     \let\reserved@c\@xifnch  
879   \else  
880     \ifx\@let@token\reserved@d  
881       \let\reserved@c\reserved@a  
882     \else  
883       \let\reserved@c\reserved@b  
884     \fi  
885   \fi  
886   \reserved@c}
```

(End of definition for \@ifnch.)

\@sptoken The following code makes \@sptoken a space token. It is important here that the control sequence `\:` consists of a non-letter only, so that the following whitespace is significant. Together with the fact that the equal sign in a `\let` may be followed by only one optional space the desired effect is achieved. NOTE: the following hacking must precede the definition of `\:` as math medium space.

```
887 \def\:{\let\@sptoken= } \: % this makes \@sptoken a space token
```

(End of definition for \@sptoken.)

\@xifnch In the following definition of \@xifnch, `\:` is again used to get a space token as delimiter into the definition.

```
888 \def\:{\@xifnch} \expandafter\def\:{ {\futurelet\@let@token\@ifnch}}
```

(End of definition for \@xifnch.)

`\@ifstar` The new implementation below avoids passing the `<true code>` Through one more `\def` than the `<false code>`, which previously meant that `#` had to be written as `####` in one argument, but `##` in the other. The `*` is gobbled by `\@firstoftwo`.

```
889 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

(End of definition for `\@ifstar`.)

`\@dblarg`

```
\@xdblarg 890 \long\def\@dblarg#1{\kernel@ifnextchar[{\#1}{\@xdblarg{#1}}}
```

```
891 \long\def\@xdblarg#1#2{\#1[\#2]{\#2}}
```

(End of definition for `\@dblarg` and `\@xdblarg`.)

`\@sanitize` The command `\@sanitize` changes the catcode of all special characters except for braces to ‘other’. It can be used for commands like `\index` that want to write their arguments verbatim. Needless to say, this command should only be executed within a group, or chaos will ensue.

```
892 \def\@sanitize{\@makeother\ \@makeother\\\@makeother$\@makeother\&%
```

```
893 \@makeother\#\@makeother\^\@makeother\_ \@makeother%\@makeother\~}
```

(End of definition for `\@sanitize`.)

`\@onelevel@sanitize` This makes the whole “meaning” of `#1` (its one-level expansion) into catcode 12 tokens: it could be used in `\DeclareRobustCommand`.

If it is to be used on default float specifiers, this should be done when they are defined.

```
894 \def \@onelevel@sanitize #1{%
895   \edef #1{\expandafter\strip@prefix
896             \meaning #1}%
897 }
```

(End of definition for `\@onelevel@sanitize`.)

`\string@makeletter` Iterates through a string, turning each alphabetic character into a catcode-11 token (partly undoes a `\detokenize`). Useful for `\ifx`-based string comparisons where `\detokenize`-ing the other string would break too much code.

The macro uses `expl3`’s `\@expl@str@map@function@NN` to iterate on the string (without losing spaces) and applies `\string@makeletter` on each character. The latter checks if character is between `a–z` or `A–Z`, and uses `\@alph` or `\@Alph` to get the corresponding catcode-11 token. Other tokens are passed through unchanged.

```
898 </2ekernel>
899 <latexrelease>\IncludeInRelease{2020/10/01}{\string@makeletter}
900 <latexrelease> {Add \string@makeletter}%
901 <*2ekernel|latexrelease>
902 \def\string@makeletter#1{%
903   \@expl@str@map@function@NN#1\string@makeletter}
904 \def\@string@makeletter#1{%
905   \@char@if@alph{#1}%
906   {\@expl@char@generate@nn{‘#1’}{11}}%
907   {#1}}
908 \def\@char@if@alph#1{%
909   \ifnum0\ifnum‘#1<‘A 1\fi\ifnum‘#1>‘z 1\fi
910   \if\ifnum‘#1>‘Z \@fi\ifnum‘#1<‘a \@fi01\fi>0
```

```

911     \expandafter\@secondoftwo
912     \else
913     \expandafter\@firstoftwo
914     \fi}
915 </2ekernel | latexrelease>
916 <latexrelease>\EndIncludeInRelease
917 %
918 <latexrelease>\IncludeInRelease{0000/00/00}{\string@makeletter}
919 <latexrelease> {Undefine \string@makeletter}%
920 <latexrelease>\let\string@makeletter\@undefined
921 <latexrelease>\let\@string@makeletter\@undefined
922 <latexrelease>\let\char@if@alph\@undefined
923 <latexrelease>\EndIncludeInRelease
924 <*2ekernel>

(End of definition for \string@makeletter, \@string@makeletter, and \char@if@alph.)

```

```

\makeatletter Make internal control sequences accessible or inaccessible.
\makeatother
925 \DeclareRobustCommand\makeatletter{\catcode'\@11\relax}
926 \DeclareRobustCommand\makeatother{\catcode'\@12\relax}

(End of definition for \makeatletter and \makeatother.)

```

2 Discretionary Hyphenation

```

\@dischyph
\@dischyph

```

Moved here to be after the definition of `\DeclareRobustCommand`.

The primitive `\-` command adds a discretionary hyphen using the current font's `\hyphenchar`. Monospace fonts are usually declared with `\hyphenchar` set to `-1` to suppress hyphenation.

$\text{\LaTeX}$, from $\text{\LaTeX}$ 2.09 in 1986 defined `\-` by

```

\def\-\{\discretionary{-}{-}{-}

```

The following comment was added when these commands were first set up, 19 April 1986:

the `\-` command is redefined to allow it to work in the `\ttfamily` type style, where automatic hyphenation is suppressed by setting `\hyphenchar` to `-1`. The original primitive $\text{\TeX}$ definition is saved as `\@@hyph` just in case anyone needs it.

$\text{\LaTeX}$ 2 ϵ , between 1993 and 2017, had a comment at this point saying that the definition “would probably change” because the definition always uses `-`. The definition used below was given in comments at this point during time.

In 2017 we finally enabled this definition by default, with the older $\text{\LaTeX}$ definition accessible via `latexrelease` as usual.

In $\text{\Lua\LaTeX}$ the primitive definition of `\-` is used directly because its use of extended hyphenation parameters means that `\-` works correctly even with `\hyphenchar` set to `-1`. This change makes `\-` under $\text{\Lua\LaTeX}$ compatible with language specific hyphenation characters.

Temporary definition of `\@latex@info`, final definition is later.

```

927 \def\@latex@info#1{}

```

```

928 </2ekernel>
929 <latexrelease>\IncludeInRelease{2020/10/01}{\-}{Use primitive \- in Lua\LaTeX}%
930 <*2ekernel | latexrelease>
931 \ifx\directlua\undefined
932   \DeclareRobustCommand{\-}{%
933     \discretionary{%
934       \char \ifnum\hyphenchar\font<\z@
935         \defaultthyphenchar
936       \else
937         \hyphenchar\font
938       \fi
939     }{}{}}%
940   }
941 \else
942   \let\-\@@hyph
943 \fi
944 </2ekernel | latexrelease>
945 <latexrelease>\EndIncludeInRelease
946 <latexrelease>\IncludeInRelease{2017/04/15}{\-}{Use \hyphenchar in \-}%
947 <latexrelease>\DeclareRobustCommand{\-}{%
948 <latexrelease>   \discretionary{%
949 <latexrelease>     \char \ifnum\hyphenchar\font<\z@
950 <latexrelease>       \defaultthyphenchar
951 <latexrelease>     \else
952 <latexrelease>       \hyphenchar\font
953 <latexrelease>     \fi
954 <latexrelease>     }{}{}}%
955 <latexrelease>}
956 <latexrelease>\EndIncludeInRelease
957 <latexrelease>\IncludeInRelease{0000/00/00}{\-}{Use \hyphenchar in \-}%
958 <latexrelease>\def\-\{\discretionary{\-}{}{}\}
959 <latexrelease>\EndIncludeInRelease

960 <*2ekernel | latexrelease>
961 \let\@dischyph=\-
962 </2ekernel | latexrelease>
963 <*2ekernel>

(End of definition for \- and \@dischyph.)
  Delayed from ltvers.dtx

964 \newif\if@includeinrelease
965 \@includeinreleasefalse

  Delayed from ltplain.dtx

966 </2ekernel>
967 <*2ekernel | latexrelease>
968 <latexrelease>\IncludeInRelease{2019/10/01}%
969 <latexrelease>      {\allowbreak}{Make various commands robust}%
970 \MakeRobust\allowbreak
971 \MakeRobust\bigbreak
972 \MakeRobust\break
973 \MakeRobust\dotfill
974 \MakeRobust\frenchspacing
975 \MakeRobust\goodbreak
976 \MakeRobust\hrulefill

```

```

977 \MakeRobust\medbreak
978 \MakeRobust\nobreak
979 \MakeRobust\nonfrenchspacing
980 \MakeRobust\obeylines
981 \MakeRobust\obeyspaces
982 \MakeRobust\slash
983 \MakeRobust\smallbreak
984 \MakeRobust\strut
985 \MakeRobust\underbar
986 </2ekernel | latexrelease>
987 <latexrelease>\EndIncludeInRelease
988 <latexrelease>\IncludeInRelease{0000/00/00}%
989 <latexrelease>                {\allowbreak}{Make various commands robust}%
990 <latexrelease>
991 <latexrelease>\kernel@make@fragile\allowbreak
992 <latexrelease>\kernel@make@fragile\bigbreak
993 <latexrelease>\kernel@make@fragile\break
994 <latexrelease>\kernel@make@fragile\dotfill
995 <latexrelease>\kernel@make@fragile\frenchspacing
996 <latexrelease>\kernel@make@fragile\goodbreak
997 <latexrelease>\kernel@make@fragile\hrulefill
998 <latexrelease>\kernel@make@fragile\medbreak
999 <latexrelease>\kernel@make@fragile\nobreak
1000 <latexrelease>\kernel@make@fragile\nonfrenchspacing
1001 <latexrelease>\kernel@make@fragile\obeylines
1002 <latexrelease>\kernel@make@fragile\obeyspaces
1003 <latexrelease>\kernel@make@fragile\slash
1004 <latexrelease>\kernel@make@fragile\smallbreak
1005 <latexrelease>\kernel@make@fragile\strut
1006 <latexrelease>\kernel@make@fragile\underbar
1007 <latexrelease>
1008 <latexrelease>\EndIncludeInRelease
1009 <*2ekernel>

```

`\g@addto@macro` Globally add to the end of a macro. This macro is used by the kernel to add to its internal hooks.

```

1010 \long\def\g@addto@macro#1#2{%
1011   \begingroup
1012     \toks@{\expandafter{#1#2}}%
1013     \xdef#1{\the\toks@}%
1014   \endgroup}

```

(End of definition for \g@addto@macro.)

```

1015 </2ekernel>

```

File 07

ltxcmd.dtx

1 Creating document commands

1.1 Overview

Creating document commands and environments using the L^AT_EX3 toolset is based around the idea that a common set of descriptions can be used to cover almost all argument types used in real documents. Thus parsing is reduced to a simple description of which arguments a command takes: this description provides the “glue” between the document syntax and the implementation of the command.

First, we will describe the argument types, then move on to explain how these can be used to create both document commands and environments. Various more specialized features are then described, which allow an even richer application of a simple interface set up.

The details here are intended to help users create document commands in general. More technical detail, suitable for T_EX programmers, is included in `interface3`.

1.2 Describing argument types

In order to allow each argument to be defined independently, the parser does not simply need to know the number of arguments for a function, but also the nature of each one. This is done by constructing an *argument specification*, which defines the number of arguments, the type of each argument and any additional information needed for the parser to read the user input and properly pass it through to internal functions.

The basic form of the argument specifier is a list of letters, where each letter defines a type of argument. As will be described below, some of the types need additional information, such as default values. The argument types can be divided into two, those which define arguments that are mandatory (potentially raising an error if not found) and those which define optional arguments. The mandatory types

- m A standard mandatory argument, which can either be a single token alone or multiple tokens surrounded by curly braces `{}`. Regardless of the input, the argument will be passed to the internal code without the outer braces. This is the type specifier for a normal T_EX argument.
- r Given as `r<token1><token2>`, this denotes a “required” delimited argument, where the delimiters are `<token1>` and `<token2>`. If the opening delimiter `<token1>` is missing, the default marker `\NoValue` will be inserted after a suitable error.
- R Given as `R<token1><token2>{<default>}`, this is a “required” delimited argument as for `r`, but it has a user-definable recovery `<default>` instead of `\NoValue`.
- v Reads an argument “verbatim”, between the following character and its next occurrence, in a way similar to the argument of the L^AT_EX₂ ϵ command `\verb`. Thus a v-type argument is read between two identical characters, which cannot be any of `%`, `\`, `#`, `{`, `}` or `␣`. The verbatim argument can also be enclosed between braces, `{` and `}`. A command with a verbatim argument will produce an error when it appears within an argument of another command.

- b Only suitable in the argument specification of an environment, it denotes the body of the environment, between `\begin{environment}` and `\end{environment}`. See Section 1.11 for details.
- c Only suitable in the argument specification of an environment, it denotes collection of the environment verbatim, between `\begin{environment}` and `\end{environment}`. See Section 1.16 for details.

The types which define optional arguments are:

- o A standard L^AT_EX optional argument, surrounded with square brackets, which will supply the special `\NoValue` marker if not given (as described later).
- d Given as `d{token1}{token2}`, an optional argument which is delimited by `<token1>` and `<token2>`. As with `o`, if no value is given the special marker `\NoValue` is returned.
- O Given as `O{default}`, is like `o`, but returns `<default>` if no value is given.
- D Given as `D{token1}{token2}{default}`, it is as for `d`, but returns `<default>` if no value is given. Internally, the `o`, `d` and `O` types are short-cuts to an appropriated-constructed `D` type argument.
- s An optional star, which will result in a value `\BooleanTrue` if a star is present and `\BooleanFalse` otherwise (as described later).
- t An optional `<token>`, which will result in a value `\BooleanTrue` if `<token>` is present and `\BooleanFalse` otherwise. Given as `t{token}`.
- e Given as `e{tokens}`, a set of optional *embellishments*, each of which requires a *value*. If an embellishment is not present, `\NoValue` is returned. Each embellishment gives one argument, ordered as for the list of `<tokens>` in the argument specification. All `<tokens>` must be distinct.
- E As for `e` but returns one or more `<defaults>` if values are not given: `E{tokens}{defaults}`. See Section 1.7 for more details.

1.3 Modifying argument descriptions

In addition to the argument *types* discussed above, the argument description also gives special meaning to three other characters.

First, `+` is used to make an argument long (to accept paragraph tokens). In contrast to `\newcommand`, this applies on an argument-by-argument basis. So modifying the example to “`s o o +m Odefault`” means that the mandatory argument is now `\long`, whereas the optional arguments are not.

Secondly, `!` is used to control whether spaces are allowed before optional arguments. There are some subtleties to this, as T_EX itself has some restrictions on where spaces can be “detected”: more detail is given in Section 1.6.

Thirdly, `=` is used to declare that the following argument should be interpreted as a series of keyvals. See Section 1.9 for more details.

Finally, the character `>` is used to declare so-called ‘argument processors’, which can be used to modify the contents of an argument before it is passed to the macro definition. The use of argument processors is a somewhat advanced topic, (or at least a less commonly used feature) and is covered in Section 1.10.

1.4 Creating document commands and environments

<code>\NewDocumentCommand</code>	<code>\NewDocumentCommand {<cmd>} {<arg spec>} {<code>}</code>
<code>\RenewDocumentCommand</code>	
<code>\ProvideDocumentCommand</code>	
<code>\DeclareDocumentCommand</code>	

This family of commands are used to create a `<cmd>`. The argument specification for the function is given by `<arg spec>`, and the command uses the `<code>` with `#1`, `#2`, etc. replaced by the arguments found by the parser.

An example:

```
\NewDocumentCommand\chapter{s o m}
{%
  \IfBooleanTF{#1}%
    {\typesetstarchapter{#3}}%
    {\typesetnormalchapter{#2}{#3}}%
}
```

would be a way to define a `\chapter` command which would essentially behave like the current $\text{\LaTeX 2}_{\epsilon}$ command (except that it would accept an optional argument even when a `*` was parsed). The `\typesetnormalchapter` could test its first argument for being `\NoValue` to see if an optional argument was present. (See Section 1.8 for details of `\IfBooleanTF` and testing for `\NoValue`.)

The difference between the `\New...`, `\Renew...`, `\Provide...` and `\Declare...` versions is the behavior if `<cmd>` is already defined.

- `\NewDocumentCommand` will issue an error if `<cmd>` has already been defined.
- `\RenewDocumentCommand` will issue an error if `<cmd>` has not previously been defined.
- `\ProvideDocumentCommand` creates a new definition for `<cmd>` only if one has not already been given.
- `\DeclareDocumentCommand` will always create the new definition, irrespective of any existing `<cmd>` with the same name. This should be used sparingly.

If the `<cmd>` can't be provided as a single token but needs “constructing”, you can use `\ExpandArgs` as explained in Section ?? which also gives an example in which this is needed.

<code>\NewDocumentEnvironment</code>	<code>\NewDocumentEnvironment {<env>} {<arg spec>} {<beg-code>} {<end-code>}</code>
<code>\RenewDocumentEnvironment</code>	
<code>\ProvideDocumentEnvironment</code>	
<code>\DeclareDocumentEnvironment</code>	

These commands work in the same way as `\NewDocumentCommand`, etc., but create environments (`\begin{<env>} ... \end{<env>}`). Both the `<beg-code>` and `<end-code>` may access the arguments as defined by `<arg spec>`. The arguments will be given following `\begin{<env>}`. Any spaces at the start and end of the `{<env>}` are removed before the definition takes place, thus


```
\NewDocumentEnvironment{foo}
```

and

```
\NewDocumentEnvironment{ foo }
```

both create the same “foo” environment.

1.5 Optional arguments

In contrast to commands created using L^AT_EX 2_ε’s `\newcommand`, optional arguments created using `\NewDocumentCommand` may safely be nested. Thus for example, following

```
\NewDocumentCommand\foo{om}{I grabbed ‘#1’ and ‘#2’}  
\NewDocumentCommand\baz{o}{#1-#1}
```

using the command as

```
\foo[\baz[stuff]]{more stuff}
```

will print

```
I grabbed ‘stuff-stuff’ and ‘more stuff’
```

This is particularly useful when placing a command with an optional argument *inside* the optional argument of a second command.

When an optional argument is followed by a mandatory argument with the same delimiter, the parser issues a warning because the optional argument could not be omitted by the user, thus becoming in effect mandatory. This can apply to `o`, `d`, `O`, `D`, `s`, `t`, `e`, and `E` type arguments followed by `r` or `R`-type required arguments.

The default for `O`, `D` and `E` arguments can be the result of grabbing another argument. Thus for example

```
\NewDocumentCommand\foo{O{#2} m}
```

would use the mandatory argument as the default for the leading optional one.

1.6 Spacing and optional arguments

T_EX will find the first argument after a function name irrespective of any intervening spaces. This is true for both mandatory and optional arguments. So `\foo[arg]` and `\foo_ [arg]` are equivalent. Spaces are also ignored when collecting arguments up to the last mandatory argument to be collected (as it must exist). So after

```
\NewDocumentCommand\foo{m o m}{ ... }
```

the user input `\foo{arg1}[arg2]{arg3}` and `\foo{arg1}_ [arg2]_ [arg3]` will both be parsed in the same way.

The behavior of optional arguments *after* any mandatory arguments is selectable. The standard settings will allow spaces here, and thus with

```
\NewDocumentCommand\foobar{m o}{ ... }
```

both `\foobar{arg1}[arg2]` and `\foobar{arg1}_ [arg2]` will find an optional argument. This can be changed by giving the modified `!` in the argument specification:

```
\NewDocumentCommand\foobar{m !o}{ ... }
```

where `\foobar{arg1}_[arg2]` will not find an optional argument.

There is one subtlety here due to the difference in handling by T_EX of ‘control symbols’, where the command name is made up of a single character, such as ‘\’. Spaces are not ignored by T_EX here, and thus it is possible to require an optional argument directly follow such a command. The most common example is the use of \ in `amsmath` environments, which in the terms here would be defined as

```
\NewDocumentCommand\\{!s !o}{ ... }
```

Also notable when using optional arguments in the last position is that T_EX will necessarily look ahead for the argument opening token. This means that the value of `\inputlineno` will be ‘out by one’ if such a trailing optional argument is *not* present and the command ends a line; it will be one greater than the line number containing the last mandatory argument.

1.7 ‘Embellishments’

The E-type argument allows one default value per test token. This is achieved by giving a list of defaults for each entry in the list, for example:

```
E{^_}{{UP}{DOWN}}
```

If the list of default values is *shorter* than the list of test tokens, the special `\NoValue` marker will be returned (as for the e-type argument). Thus for example

```
E{^_}{{UP}}
```

has default UP for the `^` test character, but will return the `\NoValue` marker as a default for `_`. This allows mixing of explicit defaults with testing for missing values.

1.8 Testing special values

Optional arguments make use of dedicated variables to return information about the nature of the argument received.

```
\IfNoValueTF \IfNoValueTF {<arg>} {<true code>} {<false code>}
\IfNoValueT
\IfNoValueF
```

The `\IfNoValue(TF)` tests are used to check if `<argument>` (`#1`, `#2`, *etc.*) is the special `\NoValue` marker. For example

```
\NewDocumentCommand\foo{o m}
{%
  \IfNoValueTF {#1}%
    {\DoSomethingJustWithMandatoryArgument{#2}}%
    {\DoSomethingWithBothArguments{#1}{#2}}%
}
```

will use a different internal function if the optional argument is given than if it is not present.

Note that three tests are available, depending on which outcome branches are required: `\IfNoValueTF`, `\IfNoValueT` and `\IfNoValueF`.

As the `\IfNoValue(TF)` tests are expandable, it is possible to test these values later, for example at the point of typesetting or in an expansion context.

When two optional arguments follow each other (a syntax we typically discourage), it can make sense to allow users of the command to specify only the second argument by providing an empty first argument. If you wish to test if an argument is blank or not, but are not concerned with distinguishing an entirely absent argument from an empty one, use the `0` type specifier with the conditional `\IfBlankTF` (described below).

```

\IfValueTF \IfValueTF {<arg>} {<true code>} {<false code>}
\IfValueT
\IfValueF

```

The reverse form of the `\IfNoValue(TF)` tests are also available as `\IfValue(TF)`. The context will determine which logical form makes the most sense for a given code scenario.

```

\IfBlankTF \IfBlankTF {<arg>} {<true code>} {<false code>}
\IfBlankT
\IfBlankF

```

The `\IfNoValueTF` command chooses the `<true code>` if the optional argument has not been used at all (and it returns the special `\NoValue` marker), but not if it has been given an empty value. In contrast `\IfBlankTF` returns true if its argument is either truly empty or only contains one or more normal blanks. For example

```

\NewDocumentCommand\foo{m!o}{\par #1:
  \IfNoValueTF{#2}
    {No optional}%
    {%
      \IfBlankTF{#2}
        {Blanks in or empty}%
        {Real content in}%
      }%
  \space argument!}
\foo{1}[bar] \foo{2}[ ] \foo{3}[] \foo{4}[\space] \foo{5} [x]

```

results in the following output:

- 1: Real content in argument!
- 2: Blanks in or empty argument!
- 3: Blanks in or empty argument!
- 4: Real content in argument!
- 5: No optional argument! [x]

Note that the `\space` in (4) is considered real content—because it is a command and not a “space” character—even though it results in producing a space. You can also

observe in (5) the effect of the `!` specifier, preventing the last `\foo` from interpreting `[x]` as its optional argument.

```
\BooleanTrue
\BooleanFalse
```

The `true` and `false` flags set when searching for an optional character (using `s` or `t`(*char*)) have names which are accessible outside of code blocks.

```
\IfBooleanTF \IfBooleanTF {<arg>} {<true code>} {<false code>}
\IfBooleanT
\IfBooleanF
```

Used to test if `<argument>` (`#1`, `#2`, *etc.*) is `\BooleanTrue` or `\BooleanFalse`. For example

```
\NewDocumentCommand\foo{sm}
{%
  \IfBooleanTF {#1}%
    {\DoSomethingWithStar{#2}}%
    {\DoSomethingWithoutStar{#2}}%
}
```

checks for a star as the first argument, then chooses the action to take based on this information.

1.9 Auto-converting to key–value format

Some document commands have a long history of accepting a ‘free text’ optional argument, for example `\caption` and the sectioning commands `\section`, *etc.* Introducing more sophisticated (keyval) options to these commands therefore needs a method to interpret the optional argument *either* as free text *or* as a series of keyvals. This needs to take place during argument grabbing as there is a need for careful treatment of braces to obtain the correct result.

The `=` modifier is available to allow `ltxcmd` to correctly implement this process. The modifier guarantees that the argument will be passed to further code as a series of keyvals. To do that, the `=` should be followed by an argument containing the default key name. This is used as the key in a key–value pair *if* the “raw” argument does *not* have the correct form to be interpreted as a set of keyvals.

Taking `\caption` as an example, with the demonstration implementation

```
\DeclareDocumentCommand\caption{s ={short-text} +0{#3} +m}
{%
  \showtokens{Grabbed arguments:^^J(#2)^^Jand^^J(#3)}%
}
```

the default key name is `short-text`. When the command `\caption` is then used, if the optional argument is free text such as

```
\caption[Some short text]{A much longer and more detailed text for
demonstration purposes}
```

then the output will be

```
Grabbed arguments:
(short-text={Some short text})
and
(A much longer and more detailed text for demonstration purposes)
```

On the other hand, if the caption is given with a keyval-form argument

```
\caption[label = cap:demo]%
  {A much longer and more detailed text for demonstration purposes}
```

then this will be respected

```
Grabbed arguments:
(label = cap:demo)
and
(A much longer and more detailed text for demonstration purposes)
```

Interpretation as keyval form is determined by the presence of = characters within the argument. Those in inline math mode (enclosed within $\$...\$$ or $\backslash(...\backslash)$) are ignored. An argument can be forced to be read as keyvals by including an empty entry at the start

```
\caption[=,This is now a keyval]%
% ...
\caption[This is not  $\$=\$$  keyval]%
```

This empty entry is *not* passed to the underlying code, so will not lead to issues with keyval parsers that do not allow an empty key name. Any text-mode = signs will need to be braced to avoid being misinterpreted: this is likely most conveniently handled by bracing the entire argument

```
\caption[{Not = to a keyval!}]%
```

which will be passed correctly as

```
Grabbed arguments:
(short-text = {Not = to a keyval!})
```

If the argument is completely blank, the conversion results in an empty keyval list, not `short-text_`. This reflects the fact that with a move toward keyval processing, an empty argument is best modelled as an empty keyval list. If the user does want an empty classical argument, using `[{}]` will work with both the new processor code and older formats.

1.10 Argument processors

Argument processor are applied to an argument *after* it has been grabbed by the underlying system but before it is passed to `<code>`. An argument processor can therefore be used to regularize input at an early stage, allowing the internal functions to be completely independent of input form. Processors are applied to user input and to default values for optional arguments, but *not* to the special `\NoValue` marker.

Each argument processor is specified by the syntax `>{<processor>}` in the argument specification. Processors are applied from right to left, so that

`>\ProcessorB >\ProcessorA m`

would apply `\ProcessorA` followed by `\ProcessorB` to the tokens grabbed by the `m` argument.

`\SplitArgument \SplitArgument {<number>} {<token(s)>}`

This processor splits the argument given at each occurrence of the `<tokens>` up to a maximum of `<number>` tokens (thus dividing the input into `<number> + 1` parts). An error is given if too many `<tokens>` are present in the input. The processed input is placed inside `<number> + 1` sets of braces for further use. If there are fewer than `{<number>}` of `{<tokens>}` in the argument then `\NoValue` markers are added at the end of the processed argument.

```
\NewDocumentCommand\foo{>\SplitArgument{2}{;}} m}
{\InternalFunctionOfThreeArguments#1}
```

If only a single character `<token>` is used for the split, any category code 13 (active) character matching the `<token>` will be replaced before the split takes place. Spaces are trimmed at each end of each item parsed.

The `E` argument type is somewhat special, because with a single `E` in the command declaration you may end up with several arguments in a command (one formal argument per embellishment token). Therefore, when an argument processor is applied to an `e/E`-type argument, all the arguments pass through that processor before being fed to the `<code>`. For example, this command

```
\NewDocumentCommand\foo{>\TrimSpaces} e{~} }
{ [#1] (#2) }
```

applies `\TrimSpaces` to both arguments.

`\SplitList \SplitList {<token(s)>}`

This processor splits the argument given at each occurrence of the `<token(s)>` where the number of items is not fixed. Each item is then wrapped in braces within `#1`. The result is that the processed argument can be further processed using a mapping function (see below).

```
\NewDocumentCommand\foo{>\SplitList{;}} m}
{\MappingFunction#1}
```

If only a single character `<token>` is used for the split, it will take account of the possibility that the `<token>` has been made active (category code 13) and will split at such tokens. Spaces are trimmed at each end of each item parsed. Exactly one set of braces will be stripped if an entire item is surrounded by them, i.e. the following inputs and outputs result (each separate item as a brace group).

```
a      ==> {a}
{a}    ==> {a}
{a}b   ==> {{a}b}
a,b    ==> {a}{b}
{a},b  ==> {a}{b}
a,{b}  ==> {a}{b}
a,{b}c ==> {a}{{b}c}
```

\ProcessList \ProcessList {<list>} {<token(s)>}

To support `\SplitList`, the function `\ProcessList` is available to apply <tokens> to every entry in a <list>. The <tokens> can be arbitrary contents that should expect one argument after it: the list entry. For example

```
\NewDocumentCommand\foo{>{\SplitList{;}} m}
{\ProcessList{#1}{\SomeDocumentCommand}}
```

or

```
\NewDocumentCommand\foo{>{\SplitList{;}} m}
{\ProcessList{#1}{Abc \SomeDocumentCommand}}
```

\ReverseBoolean \ReverseBoolean

This processor reverses the logic of `\BooleanTrue` and `\BooleanFalse`, so that the example from earlier would become

```
\NewDocumentCommand\foo{>{\ReverseBoolean} s m}
{%
  \IfBooleanTF#1%
    {\DoSomethingWithoutStar{#2}}%
    {\DoSomethingWithStar{#2}}%
}
```

\TrimSpaces \TrimSpaces

Removes any leading and trailing spaces (tokens with character code 32 and category code 10) for the ends of the argument. Thus for example declaring a function

```
\NewDocumentCommand\foo{>{\TrimSpaces} m}
{\showtokens{#1}}
```

and using it in a document as

```
\foo{ hello world }
```

will show ‘hello world’ at the terminal, with the space at each end removed. `\TrimSpaces` will remove multiple spaces from the ends of the input in cases where these have been included such that the standard $\text{\TeX}$ conversion of multiple spaces to a single space does not apply.

1.11 Body of an environment

While environments `\begin{<environment>} ... \end{<environment>}` are typically used in cases where the code implementing the <environment> does not need to access the contents of the environment (its “body”), it is sometimes useful to have the body as a standard argument.

This is achieved by ending the argument specification with `b`, which is a dedicated argument type for this situation. For instance

```

\NewDocumentEnvironment{twice}{0{\ttfamily} +b}
  {#2#1#2} {}
\begin{twice}[\itshape]
  Hello world!
\end{twice}

```

typesets ‘Hello world!’*Hello world!*.

The prefix + is used to allow multiple paragraphs in the environment’s body. Argument processors can also be applied to **b** arguments. By default, spaces are trimmed at both ends of the body: in the example there would otherwise be spaces coming from the ends the lines after `[\itshape]` and `world!`. Putting the prefix `!` before **b** suppresses space-trimming.

When **b** is used in the argument specification, the last argument of the environment declaration (e.g., `\NewDocumentEnvironment`), which consists of an `<end code>` to insert at `\end{<environment>}`, is redundant since one can simply put that code at the end of the `<start code>`. Nevertheless this (empty) `<end code>` must be provided.

Environments that use this feature can be nested.

1.12 Fully-expandable document commands

Document commands created using `\NewDocumentCommand`, etc., are normally created so that they do not expand unexpectedly. This is done using engine features, so is more powerful than L^AT_EX 2_ε’s `\protect` mechanism. There are *very rare* occasion when it may be useful to create functions using a expansion-only grabber. This imposes a number of restrictions on the nature of the arguments accepted by a function, and the code it implements. This facility should only be used when *necessary*.

<code>\NewExpandableDocumentCommand</code>	<code>\NewExpandableDocumentCommand {<cmd>} {<arg spec>} {<code>}</code>
<code>\RenewExpandableDocumentCommand</code>	
<code>\ProvideExpandableDocumentCommand</code>	
<code>\DeclareExpandableDocumentCommand</code>	

This family of commands is used to create a document-level `<cmd>`, which will grab its arguments in a fully-expandable manner. The argument specification for the function is given by `<arg spec>`, and the `<cmd>` will execute `<code>`. In general, `<code>` will also be fully expandable, although it is possible that this will not be the case (for example, a function for use in a table might expand so that `\omit` is the first non-expandable non-space token).

Parsing arguments by pure expansion imposes a number of restrictions on both the type of arguments that can be read and the error checking available:

- The last argument (if any are present) must be one of the mandatory types **m**, **r** or **R**.
- The “verbatim” argument type **v** is not available.
- Argument processors (using `>`) are not available.
- It is not possible to differentiate between, for example `\foo[` and `\foo{[}`: in both cases the `[` will be interpreted as the start of an optional argument. As a result, checking for optional arguments is less robust than in the standard version.

1.13 Commands at the start of tabular cells

Creating commands that are used at the start of tabular cells imposes some restrictions on the underlying implementation. The standard L^AT_EX tabular environments (`tabular`, etc.) use a mechanism which requires that any command wrapping `\multicolumn` or similar must be “expandable”. This is *not* the case for commands created using `\NewDocumentCommand`, etc., which as detailed in Section 1.12 use an engine feature which prevents such “expansion”. Therefore, to create such wrappers for use at the start of tabular cells, you must use `\NewExpandableDocumentCommand`, for example

```
\NewExpandableDocumentCommand\MyMultiCol{m}{\multicolumn{3}{c}{#1}}
\begin{tabular}{lcr}
a & b & c \\
\MyMultiCol{stuff} \\
\end{tabular}
```

1.14 Using the verbatim argument types

As described above, the `v`-type argument may be viewed as similar to `\verb`. Before looking at exactly what that means, it is important to highlight some key differences. Most notably, *grabbing* a verbatim-like argument is separate from *typesetting* it: the latter is covered in the next section.

When grabbing a `v`-type argument, L^AT_EX first uses the kernel command `\dospecials` to turn off the “special” nature of characters. It then makes both spaces and tabs “active”, so that they can be given a custom definition. Any other characters are grabbed as-is: this means that if any characters have been made “special” and are not listed in `\dospecials`, an error will arise (see below).

The characters that are grabbed as the argument are all those between two identical: in contrast to `\verb`, the characters `\`, `{`, `}` and `%` *cannot* be used as the delimiter character. If any of the grabbed tokens have “special” meaning, an error will be issued.

For the `+v`-type argument, which allows line breaks within the argument, new-line characters are converted into `\obeyedline` commands. The standard definition of `\obeyedline` is simple `\par`, thus allowing the grabbed tokens to be used directly in typesetting. A local redefinition of `\obeyedline` can be used to achieve other outputs. For example, to retain blank lines whilst typesetting, one could use

```
\renewcommand*\obeyedline{\mbox{}}\par}
```

More information about using these arguments in typesetting is in the following subsection.

Some additional details that may be useful for those with more T_EX knowledge: do not worry if this does not make sense to you! Spaces and tabs are stored as active characters. In 8-bit engines, non-ASCII characters are “active”, whilst other than the letters a–zA–Z, ASCII characters are “other”. In Unicode engines, non-ASCII codepoints will be either letters or “other”, based on the standard L^AT_EX settings derived from Unicode data. For token-based comparisons, it is likely that the active spaces and tabs should be replaced: this can be done conveniently by expansion.

1.15 Typesetting verbatim-like material

In contrast to `\verb`, the `(+)v`-type argument is only about *grabbing* the argument, not *typesetting* it. As such, features that users often associate with “verbatim” are not automatically activated, e.g., selecting a monospaced font. Material grabbed by the `v`-type argument does not automatically suppress ligatures: with modern $\TeX$ engines, this largely can be done without the token manipulation which `\verb` uses. (In `\verb`, ligatures are suppressed by making characters active and inserting a zero-width kern before the character itself.)

The `\verb` command also selects a monospaced font: this is not intrinsic to verbatim material, so will need to be set up using for example `\ttfamily`. Similarly, the `verbatim` environment sets up the meaning of `\par` suitable for breaking lines.

1.16 Verbatim environments

In some cases, when grabbing the body of an environment you will want the contents to be treated verbatim. This is available using the argument specification `c`. Like the `b` specification, this has to be the last one. Thus for example

```
\NewDocumentEnvironment{MyVerbatim}{!0{\ttfamily} c}
  {\begin{center} #1 #2\end{center}} {}
\begin{MyVerbatim}[\ttfamily\itshape]
  % Some code is shown here
  $y = mx + c$
\end{MyVerbatim}
```

will typeset verbatim the content, thus:

```
%\UUU%\USomeUcodeUisUshownUhere
%UUU$yU=UmxU+Uc$
%U
```

Since grabbing the entire contents verbatim will result in there being no `\par` tokens, newlines are always permitted: there is no need for a `+` modifier here. As for the `v` specification, newlines are stored as `\obeyedline`. In a similar fashion to the `b` specification, by default *newlines* are trimmed at both ends of the body. Putting the prefix `!` before `c` suppresses this trimming.

Collection of the body takes place on a line-by-line basis: content is collected up to the end-of-line in the source, then examined before storage. This means that the line ending the environment (containing in the example above `\end{MyVerbatim}`) cannot have any text *after* the end of the environment. Text *before* the end of environment is treated normally, but note that there is no trailing `\obeyedline` added if there is text here. Other than optional arguments, no text is allowed on the opening line of the environment.

Special handling is applied to a `o`, `0`, `d` or `D` specification argument immediately before an `c` specification. This means that when the optional argument is absent, the first character of the next line will be read with the correctly applied verbatim category code. Issues may arise if *multiple* optional arguments are used before a `c` specification: this will only work reliably where the optional tokens are “other” characters.

For technical reasons, we recommend that spaces are *not* ignored when searching for an optional argument before an `c` specification: this can be achieved by adding the `!` modifier as shown in the example. However, this is left as a choice for the user.

1.17 Performance

For document commands where the argument specification is entirely comprised of `m` or `+m` entries (or is entirely empty), the internal structure created by `\NewDocumentCommand` is essentially as efficient as provided by `\newcommand*`. As such, document commands may replace constructs arising from `\newcommand`, etc., without a need to be concerned about performance. It should be noted that `\newcommand*` produces expandable results, so the direct replacement is `\NewExpandableDocumentCommand`; in most cases, however, it is better to use `\NewDocumentCommand` to give more robust structures.

1.18 Details about argument delimiters

In normal (non-expandable) commands, the delimited types look for the initial delimiter by peeking ahead (using `expl3`'s `\peek_...` functions) looking for the delimiter token. The token has to have the same meaning and “shape” of the token defined as delimiter. There are three possible cases of delimiters: character tokens, control sequence tokens, and active character tokens. For all practical purposes of this description, active character tokens will behave exactly as control sequence tokens.

1.18.1 Character tokens

A character token is characterized by its character code, and its meaning is the category code (`\catcode`). When a command is defined, the meaning of the character token is fixed into the definition of the command and cannot change. A command will correctly see an argument delimiter if the open delimiter has the same character and category codes as at the time of the definition. For example in:

```
\NewDocumentCommand { \foobar } { D<>{default} } {(#1)}  
\foobar <hello> \par  
\char_set_catcode_letter:N <  
\foobar <hello>
```

the output would be:

```
(hello)  
(default)<hello>
```

as the open-delimiter `<` changed in meaning between the two calls to `\foobar`, so the second one doesn't see the `<` as a valid delimiter. Commands assume that if a valid open-delimiter was found, a matching close-delimiter will also be there. If it is not (either by being omitted or by changing in meaning), a low-level `TEX` error is raised and the command call is aborted.

1.18.2 Control sequence tokens

A control sequence (or control character) token is characterized by its name, and its meaning is its definition. A token cannot have two different meanings at the same time. When a control sequence is defined as delimiter in a command, it will be detected as delimiter whenever the control sequence name is found in the document regardless of its current definition. For example in:

```

\cs_set:Npn \x { abc }
\NewDocumentCommand { \foobar } { D\x\y{default} } {(#1)}
\foobar \x hello\y \par
\cs_set:Npn \x { def }
\foobar \x hello\y

```

the output would be:

```

(hello)
(hello)

```

with both calls to the command seeing the delimiter `\x`.

1.19 Creating new argument processors

`\ProcessedArgument`

Argument processors allow manipulation of a grabbed argument before it is passed to the underlying code. New processor implementations may be created as functions which take one trailing argument, and which leave their result in the `\ProcessedArgument` variable. For example, `\ReverseBoolean` is defined as

```

\ExplSyntaxOn
\cs_new_protected:Npn \ReverseBoolean #1
{
  \bool_if:NTF #1
  { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
  { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
}
\ExplSyntaxOff

```

[As an aside: the code is written in `expl3`, so we don't have to worry about spaces creeping into the definition.]

```

1 <@@=cmd>
2 <*2ekernel>
3 \message{document commands,}
4 </2ekernel>

```

`ltxcmd` code contains an `^^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `^^@` to restore later:

```

5 <*2ekernel | latexrelease>
6 <latexrelease> \edef\@latexrelease@catcode@null{\the\catcode'\^^@ }
7 <latexrelease> \catcode'\^^@=12
8 \ExplSyntaxOn
9 <latexrelease> \NewModuleRelease{2020/10/01}{ltxcmd}
10 <latexrelease> {Document~command~parser}%
11 \cs_generate_variant:Nn \tl_if_in:nnT { ee }

```

1.20 Variables and constants

`\l__cmd_arg_spec_tl` Holds the argument specification after normalization of shorthands.

12 `\tl_new:N \l__cmd_arg_spec_tl`

`\l__cmd_args_tl` Token list variable for grabbed arguments.

13 `\tl_new:N \l__cmd_args_tl`

`\l__cmd_args_i_tl` Hold the modified arguments when dealing with default values or processors.

`\l__cmd_args_ii_tl`

14 `\tl_new:N \l__cmd_args_i_tl`

15 `\tl_new:N \l__cmd_args_ii_tl`

`\l__cmd_current_arg_int` The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.

16 `\int_new:N \l__cmd_current_arg_int`

`\l__cmd_total_args_int` The total number of arguments found during normalization: this is required where special action is needed for the penultimate argument.

17 `\int_new:N \l__cmd_total_args_int`

`\l__cmd_last_mandatory_arg_int`

Needed to allow handling of verbatim catcodes at the end of the setup.

18 `\int_new:N \l__cmd_last_mandatory_arg_int`

`\l__cmd_defaults_bool` The boolean indicates whether there are any argument with default value other than `\NoValue`; the token list holds the code to determine these default values in terms of other arguments.

`\l__cmd_defaults_tl`

19 `\bool_new:N \l__cmd_defaults_bool`

20 `\tl_new:N \l__cmd_defaults_tl`

`\l__cmd_environment_bool` Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.

21 `\bool_new:N \l__cmd_environment_bool`

`\l__cmd_environment_str` Name of the environment, used at definition time and at run time.

```
22 \str_new:N \l__cmd_environment_str
```

`\l__cmd_expandable_bool` Used to indicate if an expandable command is begin generated, as this affects both the acceptable argument types and how they are implemented.

```
23 \bool_new:N \l__cmd_expandable_bool
```

`\l__cmd_expandable_aux_name_tl`

Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.

```
24 \tl_new:N \l__cmd_expandable_aux_name_tl
25 \tl_set:Nn \l__cmd_expandable_aux_name_tl
26 {
27   \l__cmd_function_tl \c_space_tl
28   ( arg~ \int_use:N \l__cmd_current_arg_int )
29 }
```

`\g__cmd_grabber_int` Used (in exceptional cases) to get unique names for grabbers used by expandable commands.

```
30 \int_new:N \g__cmd_grabber_int
```

`\l__cmd_fn_tl` For passing the pre-formed name of the auxiliary to be used as the parsing function.

```
31 \tl_new:N \l__cmd_fn_tl
```

`\l__cmd_fn_code_tl` For passing the pre-formed name of the auxiliary that contains the actual code.

```
32 \tl_new:N \l__cmd_fn_code_tl
```

`\l__cmd_function_tl` Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of `\cs_to_str:N`.

```
33 \tl_new:N \l__cmd_function_tl
```

`\l__cmd_grab_expandably_bool`

When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of `xparse` that use protected functions inside `csname` constructions.

```
34 \bool_new:N \l__cmd_grab_expandably_bool
```

`\l__cmd_obey_spaces_bool` For trailing optionals.

```
35 \bool_new:N \l__cmd_obey_spaces_bool
```

<u>\l__cmd_last_delimiters_tl</u>	<p>Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.</p> <pre>36 \tl_new:N \l__cmd_last_delimiters_tl</pre>
<u>\l__cmd_long_bool</u>	<p>Used to indicate that an argument is long, on a per-argument basis.</p> <pre>37 \bool_new:N \l__cmd_long_bool</pre>
<u>\l__cmd_suppress_strip_bool</u>	<p>Used to indicate that an a pair of braces should not be stripped from an optional argument.</p> <pre>38 \bool_new:N \l__cmd_suppress_strip_bool</pre>
<u>\l__cmd_m_args_int</u>	<p>The number of m arguments: if this is the same as the total number of arguments, then a short-cut can be taken in the creation of the grabber code.</p> <pre>39 \int_new:N \l__cmd_m_args_int</pre>
<u>\l__cmd_prefixed_bool</u>	<p>When preparing the signature of non-expandable commands, indicates that the current argument is affected by a processor or by + (namely is long).</p> <pre>40 \bool_new:N \l__cmd_prefixed_bool</pre>
<u>\l__cmd_process_all_tl</u> <u>\l__cmd_process_one_tl</u> <u>\l__cmd_process_some_bool</u>	<p>When preparing the signature, the processors that will be applied to a given argument are collected in \l__cmd_process_one_tl, while \l__cmd_process_all_tl contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).</p> <pre>41 \tl_new:N \l__cmd_process_all_tl 42 \tl_new:N \l__cmd_process_one_tl 43 \bool_new:N \l__cmd_process_some_bool</pre>
<u>\l__cmd_saved_args_tl</u>	<p>Stores \l__cmd_args_tl to deal with space-trimming of b-type arguments.</p> <pre>44 \tl_new:N \l__cmd_saved_args_tl</pre>
<u>\l__cmd_signature_tl</u>	<p>Used when constructing the signature (code for argument grabbing) to hold what will become the implementation of the main function. When arguments are grabbed (at point of use of the command/environment), it also stores the code for grabbing the remaining arguments.</p> <pre>45 \tl_new:N \l__cmd_signature_tl</pre>

```

\l__cmd_some_obey_spaces_bool
\l__cmd_some_long_bool
\l__cmd_some_short_bool

```

These flags are set while normalizing the argument specification. The `obey_spaces` one is used to detect when `!` is used on an argument that is not a trailing optional argument. The other two are used to check whether all short arguments appear before long arguments: this is needed to grab arguments expandably. As soon as the first long argument is seen (other than `t`-type, whose long status is ignored) the `some_long` flag is set. The `some_short` flag is used for expandable commands, to know whether to define a short auxiliary too.

```

46 \bool_new:N \l__cmd_some_obey_spaces_bool
47 \bool_new:N \l__cmd_some_long_bool
48 \bool_new:N \l__cmd_some_short_bool

```

```

\l__cmd_final_verb_bool

```

Needed to establish whether optional arguments should be collected “verbatim safe”.

```

49 \bool_new:N \l__cmd_final_verb_bool

```

```

\q__cmd_recursion_tail

```

```

\__cmd_if_recursion_stop

```

```

\__cmd_use_i_delimit_by_q_recursion_stop:nw

```

Quarks and functions for internal processing.

```

50 \quark_new:N \q__cmd_recursion_tail
51 \quark_new:N \q__cmd_recursion_stop
52 \__kernel_quark_new_test:N \__cmd_if_recursion_tail_stop_do:Nn

```

(End of definition for `\__cmd_if_recursion_tail_stop_do:Nn` and
`\__cmd_use_i_delimit_by_q_recursion_stop:nw`.)

```

\l__cmd_tmp_prop

```

```

\l__cmd_tmptl

```

```

\l__cmd_tmptl

```

Scratch space.

```

53 \prop_new:N \l__cmd_tmp_prop
54 \tl_new:N \l__cmd_tmptl
55 \tl_new:N \l__cmd_tmptl
56 \cs_new_eq:NN \__cmd_tmp:w ?

```

(End of definition for `\__cmd_tmp:w`.)

With `xparse`, information about commands being (re)defined was switched off by default, unless the `log-declarations` package option was used, so here we’ll switch that off as well.

```

57 \msg_redirect_module:nnn { cmd } { info } { none }

```

Also add `cmd` to the LaTeX messages.

```

58 \prop_gput:Nnn \g_msg_module_type_prop { cmd } { LaTeX }

```

1.21 Declaring commands and environments

```

\__cmd_declare_cmd:Nnn
\__cmd_declare_expandable_cmd:Nnn
\__cmd_declare_cmd_aux:Nnn
\__cmd_declare_cmd_internal:Nnn

```

The main functions for creating commands set the appropriate flag then use the same internal code to do the definition.

```

59 \cs_new_protected:Npn \__cmd_declare_cmd:Nnn
60 {

```



```

61   \bool_set_false:N \l__cmd_expandable_bool
62   \__cmd_declare_cmd_aux:Nnn
63 }
64 \cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
65 {
66   \bool_set_true:N \l__cmd_expandable_bool
67   \__cmd_declare_cmd_aux:Nnn
68 }

```

The first stage is to log information, both for the user in the log and for programmatic use in a property list of all declared commands.

```

69 \cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
70 {
71   \cs_if_exist:NTF #1
72   {
73     \msg_info:nnee { cmd } { redefine }
74     { \token_to_str:N #1 } { \tl_to_str:n {#2} }
75   }
76   {
77     \bool_lazy_or:nnT
78     { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
79     { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
80     {
81       \msg_warning:nne { cmd } { unsupported-let }
82       { \token_to_str:N #1 }
83     }
84     \msg_info:nnee { cmd } { define-command }
85     { \token_to_str:N #1 } { \tl_to_str:n {#2} }
86   }
87   \bool_set_false:N \l__cmd_environment_bool
88   \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }
89 }

```

At definition time, the variable `\l__cmd_fn_tl` is only used for error messages. The real business of defining a document command starts with setting up the appropriate name, then normalizing the argument specification to get rid of shorthands.

```

90 \cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
91 {
92   \tl_set:Nx \l__cmd_function_tl { \cs_to_str:N #1 }
93   \__cmd_normalize_arg_spec:n {#2}
94   \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_tl
95   \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
96   #4
97   \__cmd_break_point:n {#2}
98 }

```

(End of definition for `\__cmd_declare_cmd:Nnn` and others.)

`\__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```

99 \cs_new_eq:NN \__cmd_break_point:n \use_none:n

```

(End of definition for `\__cmd_break_point:n`.)

`\__cmd_all_m_check:n` A quick loop to check for all (+)m-type arguments.

```

\__cmd_all_m_check_aux:n 100 \cs_new:Npn \__cmd_all_m_check:n #1

```

```

101 { \tl_map_function:nN {#1} \__cmd_all_m_check_aux:n }
102 \cs_new:Npn \__cmd_all_m_check_aux:n #1
103 {
104   \str_if_eq:nnF {#1} { m }
105   {
106     \str_if_eq:nnF {#1} { + }
107     { X }
108   }
109 }

```

(End of definition for __cmd_all_m_check:n and __cmd_all_m_check_aux:n.)

__cmd_declare_cmd_code:Nnn At this stage we can check for a short-cut possibility: if the argument specification is made up of just (+)m tokens, and if all arguments are either short or long, then we can produce an optimized document command. This only applies to document commands, not creation of environments (which are more complex).

```

\__cmd_declare_cmd_optimized:Nnn
\__cmd_declare_cmd_code_aux:Nnn
\__cmd_declare_cmd_code_expandable:Nnn
\__cmd_start_optimized:
110 \cs_new_protected:Npn \__cmd_declare_cmd_code:Nnn #1#2
111 {
112   \bool_lazy_any:nTF
113   {
114     { \l__cmd_environment_bool }
115     {
116       \bool_lazy_and_p:nn
117       { \l__cmd_some_short_bool }
118       { \l__cmd_some_long_bool }
119     }
120     { ! \tl_if_blank_p:e { \__cmd_all_m_check:n {#2} } }
121   }
122   {
123     \tl_set:Nx \l__cmd_fn_tl
124     { \exp_not:c { \l__cmd_function_tl \c_space_tl } }
125     \bool_if:NTF \l__cmd_grab_expandably_bool
126     { \__cmd_declare_cmd_code_expandable:Nnn }
127     { \__cmd_declare_cmd_code_aux:Nnn }
128   }
129   { \__cmd_declare_cmd_optimized:Nnn }
130   #1 {#2}
131 }

```

The optimized version of commands just has to worry about whether to make them protected or long. The commands start with an expandable marker so that other parts of the kernel know these are set up by ltcmd. We need the two layers of redirection so that the code internal function has the same form as it would for any other document command. Optimization means that there is no \group_align_safe_begin: before grabbing the arguments, so anything involving & tokens will not work. However, this is really only intended for making optional argument processing safe anyway, so in practice should not be an issue.

```

132 \cs_new_protected:Npn \__cmd_declare_cmd_optimized:Nnn #1#2#3
133 {
134   \bool_if:NTF \l__cmd_expandable_bool
135   { \cs_set_nopar:Npe }
136   { \cs_set_protected_nopar:Npe }
137   #1
138   {

```

```

139         \exp_not:N \__cmd_start_optimized:
140         \exp_not:c { \l__cmd_function_tl \c_space_tl code }
141     }
142     \exp_args:Ncc \cs_generate_from_arg_count:NNnn
143     { \l__cmd_function_tl \c_space_tl code }
144     {
145         cs_set
146         \bool_if:NF \l__cmd_expandable_bool { _protected }
147         \bool_if:NF \l__cmd_some_long_bool { _nopar }
148         :Npn
149     }
150     \l__cmd_current_arg_int
151     {#3}
152 }
153 \cs_new:Npn \__cmd_start_optimized: { }

```

Standard functions call `\__cmd_start:nNNnnn`, which receives the argument specification, an auxiliary used for grabbing arguments, an auxiliary containing the code, and then the signature, default arguments, and processors.

```

154 \cs_new_protected:Npn \__cmd_declare_cmd_code_aux:Nnn #1#2#3
155 {
156     \cs_generate_from_arg_count:cNnn
157     { \l__cmd_function_tl \c_space_tl code }
158     \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
159     \cs_set_protected_nopar:Npe #1
160     {
161         \bool_if:NTF \l__cmd_environment_bool
162         {
163             \__cmd_start_env:nnnnn { \exp_not:n {#2} }
164             { \l__cmd_environment_str }
165         }
166         {
167             \__cmd_start:nNNnnn { \exp_not:n {#2} }
168             \exp_not:c { \l__cmd_function_tl \c_space_tl }
169             \exp_not:c { \l__cmd_function_tl \c_space_tl code }
170         }
171         { \exp_not:o \l__cmd_signature_tl }
172         {
173             \bool_if:NT \l__cmd_defaults_bool
174             { \exp_not:o \l__cmd_defaults_tl }
175         }
176         {
177             \bool_if:NT \l__cmd_process_some_bool
178             { \exp_not:o \l__cmd_process_all_tl }
179         }
180     }
181 }

```

Expandable functions and functions whose arguments can be grabbed expandably call `\__cmd_start_expandable:nNNNNn`, which receives the argument specification, four auxiliaries (two for grabbing arguments, one for the code, and one for default arguments), and finally the signature. Non-expandable functions that take this branch should nevertheless be protected, as well as their code function. They will only be expanded in contexts such as constructing a csname. The two grabbers (named after the function

with one or two spaces) are needed when there are both short and long arguments; otherwise the same grabber is included twice in the definition. If all arguments are long or all are short the (only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```

182 \cs_new_protected:Npn \__cmd_declare_cmd_code_expandable:Nnn #1#2#3
183 {
184   \exp_args:Ncc \cs_generate_from_arg_count:NNnn
185   { \l__cmd_function_tl \c_space_tl code }
186   { cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
187   \l__cmd_current_arg_int {#3}
188   \bool_if:NT \l__cmd_defaults_bool
189   {
190     \use:e
191     {
192       \cs_generate_from_arg_count:cNnn
193       { \l__cmd_function_tl \c_space_tl defaults }
194       \cs_set:Npn \l__cmd_current_arg_int
195       { \exp_not:o \l__cmd_defaults_tl }
196     }
197   }
198   \bool_if:NTF \l__cmd_expandable_bool
199   { \cs_set_nopar:Npe } { \cs_set_protected_nopar:Npe } #1
200   {
201     \exp_not:N \__cmd_start_expandable:nNNNNn
202     { \exp_not:n {#2} }
203     \exp_not:c { \l__cmd_function_tl \c_space_tl }
204     \exp_not:c
205     {
206       \l__cmd_function_tl \c_space_tl
207       \bool_if:NT \l__cmd_some_short_bool
208       { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }
209     }
210     \exp_not:c { \l__cmd_function_tl \c_space_tl code }
211     \bool_if:NTF \l__cmd_defaults_bool
212     { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
213     { ? }
214     { \exp_not:o \l__cmd_signature_tl }
215   }
216   \bool_if:NTF \l__cmd_some_long_bool
217   {
218     \bool_if:NT \l__cmd_some_short_bool
219     {
220       \cs_set_nopar:cpe { \l__cmd_function_tl \c_space_tl \c_space_tl }
221       ##1##2 { ##1 {##2} }
222     }
223     \cs_set:cpe
224     {
225       \cs_set_nopar:cpe {
226         { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
227       }
228     }
229   }
230 }

```

(End of definition for __cmd_declare_cmd_code:Nnn and others.)

```

\__cmd_declare_env:nnnn
\__cmd_declare_env:ennn
\__cmd_declare_env_internal:nnnn
\__cmd_set_environment_end:n

```

The lead-off to creating an environment is much the same as that for creating a command: issue the appropriate message, store the argument specification then hand off to an internal function.

```

228 <latexrelease> \IncludeInRelease{2024/11/01}{\__cmd_declare_env:nnnn}%
229 <latexrelease> {Use~space-trimmed~envname~directly}
230 \cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
231 {
232   \str_set:Nn \l__cmd_environment_str {#1}
233   \cs_if_exist:cTF { #1 }
234     { \msg_info:nnnn { cmd } { redefine-env } { #1 } { #2 } }
235     { \msg_info:nnnn { cmd } { define-env } { #1 } { #2 } }
236   \bool_set_false:N \l__cmd_expandable_bool
237   \bool_set_true:N \l__cmd_environment_bool
238   \__cmd_declare_env_internal:nnnn {#1} {#2}
239 }
240 \cs_generate_variant:Nn \__cmd_declare_env:nnnn { e }
241 <latexrelease> \EndIncludeInRelease
242 <latexrelease> \IncludeInRelease{2024/06/01}{\__cmd_declare_env:nnnn}%
243 <latexrelease> {Use~space-trimmed~envname~directly}
244 <latexrelease> \cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
245 <latexrelease> {
246   <latexrelease> \str_set:Nx \l__cmd_environment_str {#1}
247   <latexrelease> \str_set:Nx \l__cmd_environment_str
248     { \tl_trim_spaces:o { \l__cmd_environment_str } }
249   <latexrelease> \cs_if_exist:cTF { \l__cmd_environment_str }
250     {
251       <latexrelease> \msg_info:nnxx { cmd } { redefine-env }
252       <latexrelease> { \l__cmd_environment_str } { \tl_to_str:n {#2} }
253     }
254     {
255       <latexrelease> \msg_info:nnxx { cmd } { define-env }
256       <latexrelease> { \l__cmd_environment_str } { \tl_to_str:n {#2} }
257     }
258   <latexrelease> \bool_set_false:N \l__cmd_expandable_bool
259   <latexrelease> \bool_set_true:N \l__cmd_environment_bool
260   <latexrelease> \exp_args:NV \__cmd_declare_env_internal:nnnn
261   <latexrelease> \l__cmd_environment_str {#2}
262 <latexrelease> }
263 <latexrelease>
264 <latexrelease> \EndIncludeInRelease

```

Creating a document environment requires a few more steps than creating a single command. In order to pass the arguments of the command to the end of the function, it is necessary to store the grabbed arguments. To do that, the function used at the end of the environment has to be redefined to contain the appropriate information. To minimize the amount of expansion at point of use, the code here is expanded now as well as when used. The last argument of `\__cmd_declare_cmd_internal:Nnnn` is only run if the definition succeeded. In package mode this ensures that the original definition of the environment is not changed if the definition fails for any reason. This also avoids an error when defining the `end_au_x` function when the user asks for more than 9 arguments.

```

265 \cs_new_protected:Npn \__cmd_declare_env_internal:nnnn #1#2#3#4
266 {
267   \exp_args:Nc \__cmd_declare_cmd_internal:Nnnn { environment~ #1 } {#2}

```

```

268     {#3}
269     {
270         \cs_set_nopar:cpe { environment~ #1 ~end }
271         { \exp_not:c { environment~ #1 ~end~aux } }
272         \cs_generate_from_arg_count:cNnn
273         { environment~ #1 ~end~aux~ } \cs_set:Npn
274         \l__cmd_current_arg_int {#4}
275         \cs_set_eq:cc {#1} { environment~ #1 }
276         \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
277     }
278 }
279 \cs_new_protected:Npn \__cmd_set_environment_end:n #1
280 {
281     \cs_set_nopar:cpe { environment~ #1 ~end~aux }
282     {
283         \exp_not:c { environment~ #1 ~end~aux~ }
284         \exp_not:o \l__cmd_args_tl
285     }
286 }

```

(End of definition for __cmd_declare_env:nnnn, __cmd_declare_env_internal:nnnn, and __cmd_set_environment_end:n.)

1.22 Structure of xparse commands

__cmd_start_env:nnnnn For error messages that occur during run-time when getting arguments of environments it is necessary to keep track of the environment name. We begin non-expandable commands with a token equal to \scan_stop:, whose name gives a reasonable error message if the command is used inside a csname and protects against f-expansion. This is useless for environments since \begin is already not expandable. Both the command and environment codes start with \group_align_safe_begin:, then __cmd_run_code: (used by both) does \group_align_safe_end:, so that delimited arguments may be grabbed in alignments if they contain an alignment tab token (see latex3/latex3/issues/839).

```

287 \cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
288 {
289     \conditionally@traceoff
290     \group_align_safe_begin:
291     \str_set:Nn \l__cmd_environment_str {#2}
292     \bool_set_true:N \l__cmd_environment_bool
293     \__cmd_start_aux:ccnnnn
294     { environment~ \l__cmd_environment_str \c_space_tl }
295     { environment~ \l__cmd_environment_str \c_space_tl code }
296     {#1}
297 }
298 \cs_new_protected:Npn \__cmd_start:nNNnnn #1#2#3
299 {
300     \exp_not:c { xparse~function~is~not~expandable }
301     \exp_not:N \conditionally@traceoff
302     \exp_not:N \group_align_safe_begin:
303     \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
304     \exp_not:N \__cmd_start_aux:NNnnnn
305     #2 #3 {#1}
306 }

```

(End of definition for _cmd_start_env:nnnnn and _cmd_start:nNnnnn.)

_cmd_start_aux:NNnnnn This sets up a few variables to minimize the boilerplate code included in all xparse-defined commands. It then runs the grabbers #4. Again, the argument specification #1 is only for diagnostics.

_cmd_start_aux:ccnnnn

```

307 \cs_new_protected:Npn \_cmd_start_aux:NNnnnn #1#2#3#4#5#6
308 {
309   \tl_clear:N \l__cmd_args_tl
310   \tl_set:Nn \l__cmd_fn_tl {#1}
311   \tl_set:Nn \l__cmd_fn_code_tl {#2}
312   \tl_set:Nn \l__cmd_defaults_tl {#5}
313   \tl_set:Nn \l__cmd_process_all_tl {#6}
314   #4
315   \__cmd_run_code:
316 }
317 \cs_generate_variant:Nn \_cmd_start_aux:NNnnnn { cc }

```

(End of definition for _cmd_start_aux:NNnnnn.)

_cmd_run_code: After arguments are grabbed, this function is responsible for inserting default values, running processors, and finally doing \group_align_safe_end: as promised, and running the code.

```

318 \cs_new_protected:Npn \_cmd_run_code:
319 {
320   \tl_if_empty:NF \l__cmd_defaults_tl { \_cmd_defaults: }
321   \tl_if_empty:NF \l__cmd_process_all_tl { \_cmd_args_process: }
322   \bool_if:NT \l__cmd_environment_bool
323     { \exp_args:No \_cmd_set_environment_end:n \l__cmd_environment_str }
324   \group_align_safe_end:
325   \conditionally@traceon
326   \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
327 }

```

(End of definition for _cmd_run_code:.)

_cmd_defaults: First construct _cmd_tmp:w (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

_cmd_defaults_def:

_cmd_defaults_def:nn

_cmd_defaults_def:nnn

_cmd_defaults_aux:

_cmd_defaults_error:w

```

328 \cs_new_protected:Npn \_cmd_defaults:
329 {
330   \_cmd_defaults_def:
331   \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
332   \_cmd_defaults_aux: \_cmd_defaults_aux: \_cmd_defaults_aux:
333   \_cmd_defaults_aux: \_cmd_defaults_aux: \_cmd_defaults_aux:
334   \_cmd_defaults_aux: \_cmd_defaults_aux: \_cmd_defaults_aux:
335   \_cmd_defaults_error:w
336   \q_recursion_stop
337   \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
338 }
339 \cs_new_protected:Npn \_cmd_defaults_aux:
340 {
341   \tl_set:Ne \l__cmd_args_ii_tl

```

```

342     { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
343     \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
344     { \use_none_delimit_by_q_recursion_stop:w }
345     \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
346   }
347 \cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
348 {
349   \msg_error:nne { cmd } { default-loop }
350   { \__cmd_environment_or_command: }
351 }

```

To construct `\__cmd_tmp:w`, first go through the arguments found and the corresponding defaults, building a token list with `{#(arg number)}` for arguments found in the input (whose default will not be used) and otherwise `{\exp_not:n{<default>}}` for arguments whose default will be used.

```

352 \cs_new_protected:Npn \__cmd_defaults_def:
353 {
354   \tl_clear:N \l__cmd_tmpa_tl
355   \int_zero:N \l__cmd_current_arg_int
356   \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
357   \__cmd_defaults_def:nn
358   \cs_generate_from_arg_count:NNno \__cmd_tmp:w \cs_set:Npn
359   \l__cmd_current_arg_int \l__cmd_tmpa_tl
360 }
361 \cs_new_protected:Npn \__cmd_defaults_def:nn
362 {
363   \int_incr:N \l__cmd_current_arg_int
364   \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
365 }
366 \cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
367 {
368   \tl_put_right:Ne \l__cmd_tmpa_tl
369   {
370     {
371       \exp_not:N \exp_not:n
372       {
373         \tl_if_novalue:nTF {#2}
374         { \exp_not:o {#3} }
375         { \exp_not:n { ## #1 } }
376       }
377     }
378   }
379 }

```

(End of definition for `\__cmd_defaults:` and others.)

`\__cmd_args_process:` Loop through arguments (stored in `\l__cmd_args_tl`) and the corresponding processors (in `\l__cmd_process_all_tl`) simultaneously, apply all processors for each argument and store the result back into `\l__cmd_args_tl`. To allow processors to depend on other arguments, for every processor define a temporary auxiliary that receives all arguments `\l__cmd_args_tl`.

```

380 \cs_new_protected:Npn \__cmd_args_process:
381 {
382   \tl_clear:N \l__cmd_args_ii_tl

```



```

383 \__cmd_tl_mapthread_function:NNN
384 \l__cmd_args_tl
385 \l__cmd_process_all_tl
386 \__cmd_args_process_loop:nn
387 \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_ii_tl
388 }
389 \cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
390 {
391 \tl_set:Nn \ProcessedArgument {#1}
392 \tl_if_novalue:nF {#1}
393 { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
394 \tl_put_right:No \l__cmd_args_ii_tl
395 { \exp_after:wN { \ProcessedArgument } }
396 }
397 \cs_new_protected:Npn \__cmd_args_process_aux:n #1
398 {
399 \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
400 { \tl_count:N \l__cmd_args_tl } {#1}
401 \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
402 { \ProcessedArgument }
403 }

```

(End of definition for __cmd_args_process:, __cmd_args_process_loop:nn, and __cmd_args_process_aux:n.)

__cmd_start_expandable:nNNNNn This is called for all expandable commands. #6 is the signature, responsible for grabbing arguments. #5 is used to determine default values (or is ? if there are none). #4 is the code to run. #2 and #3 are functions (named after the command) that grab a single argument in the input stream (#3 is short). The argument specification #1 is only used by diagnostic functions. Same as for the non-expandable version, this starts with \group_align_safe_begin:, which expands to nothing, so may be safely used in an expandable context.

```

404 \cs_new:Npn \__cmd_start_expandable:nNNNNn #1#2#3#4#5#6
405 {
406 \group_align_safe_begin:
407 #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
408 }

```

(End of definition for __cmd_start_expandable:nNNNNn.)

__cmd_end_expandable:NNw __cmd_end_expandable_aux:w __cmd_end_expandable_aux:nNNNN __cmd_end_expandable_defaults:nnnNNn __cmd_end_expandable_defaults:nnw __cmd_end_expandable_defaults:nw

Followed by a function #1 to determine default values (or ? if there are no defaults), the code #2, arguments that have been grabbed, then \q__cmd and two generic grabbers. The idea to find default values is similar to the non-expandable case but we cannot define an auxiliary function, so at every step in the loop we need to go through all arguments searching for which ones started out as \NoValue and replacing these by the newly computed values. In fact we need to keep track of three versions of all arguments: the original version, the previous version with default values, and the currently built version (first argument of __cmd_end_expandable_defaults:nnnNNn).

```

409 \cs_new:Npn \__cmd_end_expandable:NNw #1#2
410 { \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing: }
411 \cs_new:Npn \__cmd_end_expandable_aux:w #1#2#3 \q__cmd
412 { \exp_args:No \__cmd_end_expandable_aux:nNNNN {#3} #1 #2 }
413 \cs_new:Npn \__cmd_end_expandable_aux:nNNNN #1#2#3#4#5

```

```

414 {
415   \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
416   \__cmd_end_expandable_defaults:nnnNNn {#1} { } {#1} #2#3
417   { } { } { } { } { } { } { } { } { } { } { } { }
418   {
419     \msg_expandable_error:nnf { cmd } { default-loop }
420     { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
421     \use_iv:nnnn
422   }
423   \q_stop
424 }
425 \cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
426 {
427   #6
428   \str_if_eq:nnTF {#1} {#2}
429   { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
430   {
431     \exp_args:No \__cmd_tl_mapthread_function:nnN
432     { #4 #1 } {#3}
433     \__cmd_end_expandable_defaults:nnw
434     \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
435   }
436 }
437 \cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
438 {
439   \tl_if_novalue:nTF {#2}
440   { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }
441   { \__cmd_end_expandable_defaults:nw {#2} }
442 }
443 \cs_new:Npn \__cmd_end_expandable_defaults:nw
444   #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
445   { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }

```

(End of definition for `\__cmd_end_expandable:NNw` and others.)

1.23 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers `+`, `!` and `=` are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely `G/v` always, and `l/u` after optional arguments (xparse may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.

- Keep track in `\l__cmd_some_long_bool` and `\l__cmd_some_short_bool` of whether the command has some long/short arguments.
- Keep track in `\l__cmd_grab_expandably_bool` of whether all arguments are m/l/u type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that `\l__cmd_last_delimiters_tl` is cleared by every mandatory argument and filled by every optional argument).

`\__cmd_normalize_arg_spec:n` Loop through the argument specification, calling an auxiliary specific to each argument type. If any argument is unknown stop the definition.

`\__cmd_normalize_arg_spec_loop:n`

```

446 \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
447 {
448   \int_zero:N \l__cmd_current_arg_int
449   \int_zero:N \l__cmd_last_mandatory_arg_int
450   \tl_clear:N \l__cmd_last_delimiters_tl
451   \tl_clear:N \l__cmd_arg_spec_tl
452   \bool_set_true:N \l__cmd_grab_expandably_bool
453   \bool_set_false:N \l__cmd_obey_spaces_bool
454   \bool_set_false:N \l__cmd_long_bool
455   \bool_set_false:N \l__cmd_suppress_strip_bool
456   \bool_set_false:N \l__cmd_some_obey_spaces_bool
457   \bool_set_false:N \l__cmd_some_long_bool
458   \bool_set_false:N \l__cmd_some_short_bool
459   \__cmd_normalize_arg_spec_loop:n #1
460   \q_recursion_tail \q_recursion_tail \q_recursion_tail \q_recursion_stop
461   \int_compare:nNnT \l__cmd_current_arg_int > 9
462   {
463     \msg_error:nnee { cmd } { too-many-args }
464     { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
465     \__cmd_bad_def:wn
466   }
467   \bool_if:NT \l__cmd_expandable_bool
468   {
469     \tl_if_empty:NF \l__cmd_last_delimiters_tl
470     {
471       \msg_error:nnee { cmd } { expandable-ending-optional }
472       { \iow_char:N \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
473       \__cmd_bad_def:wn
474     }
475   }
476   \bool_if:NT \l__cmd_expandable_bool
477   { \bool_set_true:N \l__cmd_grab_expandably_bool }
478   \bool_if:NT \l__cmd_environment_bool
479   { \bool_set_false:N \l__cmd_grab_expandably_bool }

```

```

480 }
481 \cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
482 {
483   \quark_if_recursion_tail_stop:n {#1}
484   \int_incr:N \l__cmd_current_arg_int
485   \cs_if_exist_use:cF { __cmd_normalize_type_ \tl_to_str:n {#1} :w }
486   {
487     \bool_lazy_any:nTF
488     {
489       { \str_if_eq_p:nn {#1} { G } }
490       { \str_if_eq_p:nn {#1} { g } }
491       { \str_if_eq_p:nn {#1} { l } }
492       { \str_if_eq_p:nn {#1} { u } }
493     }
494     {
495       \msg_error:nnee { cmd } { xparse-arg-type }
496       { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
497     }
498     {
499       \msg_error:nnee { cmd } { unknown-argument-type }
500       { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
501     }
502     \__cmd_bad_def:wn
503   }
504 }

```

(End of definition for __cmd_normalize_arg_spec:n and __cmd_normalize_arg_spec_loop:n.)

__cmd_normalize_type_d:w These argument types are aliases of more general ones, for example with the default argument \NoValue. For argument types that need additional data, check that the data is present (not \q_recursion_tail) before proceeding.

```

505 \cs_new_protected:Npn \__cmd_normalize_type_d:w #1#2
506 {
507   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
508   \__cmd_normalize_type_D:w {#1} {#2} { \NoValue }
509 }
510 \cs_new_protected:Npn \__cmd_normalize_type_e:w #1
511 {
512   \quark_if_recursion_tail_stop_do:nn {#1} { \__cmd_bad_arg_spec:wn }
513   \__cmd_normalize_type_E:w {#1} { }
514 }
515 \cs_new_protected:Npn \__cmd_normalize_type_o:w
516 { \__cmd_normalize_type_D:w [ ] { \NoValue } }
517 \cs_new_protected:Npn \__cmd_normalize_type_O:w
518 { \__cmd_normalize_type_D:w [ ] }
519 \cs_new_protected:Npn \__cmd_normalize_type_r:w #1#2
520 {
521   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
522   \__cmd_normalize_type_R_aux:w { r #1 #2 } {#1} {#2} { \NoValue }
523 }
524 \cs_new_protected:Npn \__cmd_normalize_type_s:w
525 { \__cmd_normalize_type_t:w * }

```

(End of definition for __cmd_normalize_type_d:w and others.)

`__cmd_normalize_type_>:w` Check that these prefixes have arguments, namely that the next token is not `\q-recursion_tail`, and remember to leave it after the looping macro. Processors are forbidden in expandable commands. If all is good, store the prefix in the cleaned up `\l__cmd_arg_spec_tl`, and decrement the argument number as prefixes do not correspond to arguments.

```

526 \cs_new_protected:cpn { __cmd_normalize_type_>:w } #1#2
527 {
528   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
529   \bool_if:NT \l__cmd_expandable_bool
530   {
531     \msg_error:nnee { cmd } { processor-in-expandable }
532     { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
533     \__cmd_bad_def:wn
534   }
535   \tl_put_right:Ne \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
536   \int_decr:N \l__cmd_current_arg_int
537   \bool_set_false:N \l__cmd_grab_expandably_bool
538   \__cmd_normalize_arg_spec_loop:n {#2}
539 }
540 \cs_new_protected:cpn { __cmd_normalize_type_+:w } #1
541 {
542   \__cmd_normalize_type_aux:NnNn + {#1}
543   \l__cmd_long_bool
544   { \bool_set_true:N \l__cmd_long_bool }
545 }
546 \cs_new_protected:cpn { __cmd_normalize_type_!:w } #1
547 {
548   \__cmd_normalize_type_aux:NnNn ! {#1}
549   \l__cmd_obey_spaces_bool
550   {
551     \bool_set_true:N \l__cmd_obey_spaces_bool
552     \bool_set_true:N \l__cmd_some_obey_spaces_bool
553   }
554 }
555 \cs_new_protected:cpn { __cmd_normalize_type_=:w } #1#2
556 {
557   \__cmd_normalize_type_aux:NnNn = {#2}
558   \l__cmd_suppress_strip_bool
559   {
560     \bool_if:NT \l__cmd_expandable_bool
561     {
562       \msg_error:nnee { cmd } { keyval-in-expandable }
563       { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
564       \__cmd_bad_def:wn
565     }
566     \bool_set_true:N \l__cmd_suppress_strip_bool
567     \bool_set_false:N \l__cmd_grab_expandably_bool
568     \tl_put_right:Ne \l__cmd_arg_spec_tl
569     { = { \tl_trim_spaces:n {#1} } }
570   }
571 }
572 \cs_new_protected:Npn \__cmd_normalize_type_aux:NnNn #1#2#3#4
573 {

```

```

574 \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
575 \bool_if:NT #3
576 {
577   \msg_error:nnee { cmd } { two-markers }
578   { \__cmd_environment_or_command: } { #1 }
579   \__cmd_bad_def:wn
580 }
581 #4
582 \int_decr:N \l__cmd_current_arg_int
583 \__cmd_normalize_arg_spec_loop:n {#2}
584 }

```

(End of definition for __cmd_normalize_type_>:w and others.)

__cmd_normalize_type_D:w
 __cmd_normalize_type_E:w
 __cmd_normalize_type_t:w
 __cmd_normalize_E_unique_check:w

Optional argument types. Check that all required data is present (and consists of single characters if applicable) and check for forbidden types for expandable commands. For E-type require that there is at least one embellishment, that each one is a single character, and that there aren't more optional arguments than embellishments; also remember that each embellishment counts as one argument for \l__cmd_current_arg_int. Then in each case store the data in \l__cmd_arg_spec_tl, and for later checks store in \l__cmd_last_delimiters_tl the tokens whose presence determines whether there is an optional argument (for braces store {}, seen later as an empty delimiter).

```

585 \cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
586 {
587   \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
588   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
589   \__cmd_single_token_check:n {#2}
590   \__cmd_add_arg_spec:n { D #1 #2 {#3} }
591   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
592   \bool_set_false:N \l__cmd_grab_expandably_bool
593   \__cmd_normalize_arg_spec_loop:n
594 }
595 \cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
596 {
597   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
598   \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
599   \tl_map_function:nN {#1} \__cmd_single_token_check:n
600   \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
601   \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop
602   \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
603   { \__cmd_bad_arg_spec:wn }
604   \__cmd_add_arg_spec:n { E {#1} {#2} }
605   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
606   \bool_set_false:N \l__cmd_grab_expandably_bool
607   \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
608   \__cmd_normalize_arg_spec_loop:n
609 }
610 \cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop
611 {
612   \quark_if_nil:NF #1
613   {
614     \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
615     \__cmd_normalize_E_unique_check:w #2 \q_stop
616   }

```

```

617 }
618 \cs_new_protected:Npn \__cmd_normalize_type_t:w #1
619 {
620   \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
621   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
622   \tl_put_right:Ne \l__cmd_arg_spec_tl
623   {
624     \bool_if:NT \l__cmd_obey_spaces_bool { ! }
625     t \exp_not:n {#1}
626   }
627   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
628   \bool_set_false:N \l__cmd_grab_expandably_bool
629   \bool_set_false:N \l__cmd_obey_spaces_bool
630   \bool_set_false:N \l__cmd_long_bool
631   \__cmd_normalize_arg_spec_loop:n
632 }

```

(End of definition for __cmd_normalize_type_D:w and others.)

__cmd_normalize_type_m:w
 __cmd_normalize_type_R:w
 __cmd_normalize_type_R_aux:w
 __cmd_normalize_type_v:w

Mandatory arguments. First check the required data is present, consists of single characters where applicable, and that the argument type is allowed for expandable commands if applicable. For the m and R argument types check that they do not follow some optional argument with that delimiter as otherwise the optional argument could not be omitted. Then save data in \l__cmd_arg_spec_tl, count the mandatory argument, and empty the list of last delimiters.

```

633 \cs_new_protected:Npn \__cmd_normalize_type_m:w
634 {
635   \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
636   \__cmd_add_arg_spec_mandatory:n { m }
637   \__cmd_normalize_arg_spec_loop:n
638 }
639 \cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
640 {
641   \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
642   \__cmd_normalize_type_R_aux:w { R #1 #2 {#3} } {#1} {#2} {#3}
643 }
644 \cs_new_protected:Npn \__cmd_normalize_type_R_aux:w #1#2#3#4
645 {
646   \__cmd_single_token_check:n {#2} \__cmd_allowed_token_check:N #2
647   \__cmd_single_token_check:n {#3}
648   \__cmd_delimiter_check:nnn {#2} { R/r } { \tl_to_str:n {#2} }
649   \bool_set_false:N \l__cmd_grab_expandably_bool
650   \__cmd_add_arg_spec_mandatory:nn { R #2 #3 {#4} } {#1}
651   \__cmd_normalize_arg_spec_loop:n
652 }
653 \cs_new_protected:Npn \__cmd_normalize_type_v:w
654 {
655   \__cmd_normalize_check_gv:N v
656   \__cmd_add_arg_spec_mandatory:n { v }
657   \__cmd_normalize_arg_spec_loop:n
658 }

```

(End of definition for __cmd_normalize_type_m:w and others.)

`\__cmd_normalize_type_b:w` These argument types are not allowed for commands. They are only allowed at the end of the argument specification, hence we check that #1 is the end.

```
\__cmd_normalize_type_c:w
  \__cmd_normalize_type_b_or_c:nn
659 \cs_new_protected:Npn \__cmd_normalize_type_b:w #1
660   { \__cmd_normalize_type_b_or_c:nn {#1} { b } }
661 \cs_new_protected:Npn \__cmd_normalize_type_c:w #1
662   { \__cmd_normalize_type_b_or_c:nn {#1} { c } }
663 \cs_new_protected:Npn \__cmd_normalize_type_b_or_c:nn #1#2
664   {
665     \bool_if:NF \l__cmd_environment_bool
666     {
667       \msg_error:nnee { cmd } { invalid-command-arg }
668       { \__cmd_environment_or_command: } {#2}
669       \__cmd_bad_def:wn
670     }
671     \tl_clear:N \l__cmd_last_delimiters_tl
672     \__cmd_add_arg_spec:n {#2}
673     \quark_if_recursion_tail_stop:n {#1}
674     \msg_error:nneee { cmd } { arg-after-body }
675     {#2}
676     { \__cmd_environment_or_command: }
677     { \tl_to_str:n {#1} }
678     \__cmd_bad_def:wn
679   }
```

(End of definition for `\__cmd_normalize_type_b:w`, `\__cmd_normalize_type_c:w`, and `\__cmd_normalize_type_b_or_c:nn`.)

`\__cmd_single_token_check:n` Checks that the argument is a single (non-space) token (possibly surrounded by spaces), and aborts the definition otherwise.

```
680 \cs_new_protected:Npn \__cmd_single_token_check:n #1
681   {
682     \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
683     {
684       \msg_error:nnee { cmd } { not-single-token }
685       { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
686       \__cmd_bad_def:wn
687     }
688   }
```

(End of definition for `\__cmd_single_token_check:n`.)

`\__cmd_allowed_token_check:N` Some tokens are not allowed as delimiters for some argument types, notably implicit begin/end-group tokens (`\bgroup`/`\egroup`). The major problem with these tokens is that for `\peek_...` functions, a literal `{`₁ is virtually indistinguishable from a `\bgroup` or other token which was `\let` to a `{`₁, and the same goes for `}`₂. All other tokens can be easily distinguished from their implicit counterparts by grabbing them and looking at the string length (see `\__cmd_token_if_cs:NTF`), but for begin/end group tokens that is not possible without the risk of mistakenly grabbing the entire brace group (potentially leading to a ! Runaway argument error) or trying to grab a `}`₂, leading to an ! Argument of `\dots` has an extra `}` error.

```
689 \cs_new_protected:Npn \__cmd_allowed_token_check:N #1
690   {
691     \token_if_eq_meaning:NNTF #1 \c_group_begin_token
692     { \use:n }
```



```

693     {
694         \token_if_eq_meaning:NNTF #1 \c_group_end_token
695         { \use:n }
696         { \use_none:n }
697     }
698     {
699         \msg_error:nneee { cmd } { forbidden-group-token }
700         { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
701         {
702             \token_if_eq_meaning:NNTF #1 \c_group_begin_token
703             { begin } { end }
704         }
705         \__cmd_bad_def:wn
706     }
707 }

```

(End of definition for __cmd_allowed_token_check:N.)

__cmd_normalize_check_gv:N Called for arguments that are always forbidden, or forbidden after an optional argument,
__cmd_normalize_check_lu:N for expandable commands.

```

708 \cs_new_protected:Npn \__cmd_normalize_check_gv:N #1
709 {
710     \bool_if:NT \l__cmd_expandable_bool
711     {
712         \msg_error:nnee { cmd } { invalid-expandable-arg }
713         { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
714         \__cmd_bad_def:wn
715     }
716     \bool_set_false:N \l__cmd_grab_expandably_bool
717 }
718 \cs_new_protected:Npn \__cmd_normalize_check_lu:N #1
719 {
720     \bool_if:NT \l__cmd_expandable_bool
721     {
722         \tl_if_empty:NF \l__cmd_last_delimiters_tl
723         {
724             \msg_error:nnee { cmd } { invalid-after-optional-expandably }
725             { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
726             \__cmd_bad_def:wn
727         }
728     }
729 }

```

(End of definition for __cmd_normalize_check_gv:N and __cmd_normalize_check_lu:N.)

__cmd_delimiter_check:nnn Called for m and R arguments. Checks that the leading token does not coincide with the
token denoting the presence of a previous optional argument. Instead of dealing with
braces for the m-type we use an empty delimiter to denote that case.

```

730 \cs_new_protected:Npn \__cmd_delimiter_check:nnn #1#2#3
731 {
732     \tl_map_inline:Nn \l__cmd_last_delimiters_tl
733     {
734         \tl_if_eq:nnT {##1} {#1}
735     }

```

```

736         \msg_warning:nnee { cmd } { optional-mandatory }
737         {#2} {#3}
738     }
739 }
740 }

```

(End of definition for `\__cmd_delimiter_check:nnn`.)

`\__cmd_bad_arg_spec:wn` If the argument specification is wrong, this provides an escape from the entire definition process.

```

741 \cs_new_protected:Npn \__cmd_bad_arg_spec:wn #1 \__cmd_break_point:n #2
742 {
743     \msg_error:nnee { cmd } { bad-arg-spec }
744     { \__cmd_environment_or_command: } { \tl_to_str:n {#2} }
745 }
746 \cs_new_protected:Npn \__cmd_bad_def:wn #1 \__cmd_break_point:n #2 { }

```

(End of definition for `\__cmd_bad_arg_spec:wn` and `\__cmd_bad_def:wn`.)

`\__cmd_add_arg_spec:n` When adding an argument to the argument specification, set the `some_long` or `some_short` booleans as appropriate and clear the booleans keeping track of +, ! and = markers. `\__cmd_add_arg_spec_mandatory:n` Before that, test for a short argument following some long arguments: this is forbidden for expandable commands and prevents grabbing arguments expandably. `\__cmd_add_arg_spec_mandatory:nn`

For mandatory arguments do some more work, in particular complain if they were preceded by !.

```

747 \cs_new_protected:Npn \__cmd_add_arg_spec:n #1
748 {
749     \bool_lazy_and:nnT
750     { ! \l__cmd_long_bool }
751     { \l__cmd_some_long_bool }
752     {
753         \bool_if:NT \l__cmd_expandable_bool
754         {
755             \msg_error:nne { cmd } { long-short-mix }
756             { \iow_char:N \l__cmd_function_tl }
757             \__cmd_bad_def:wn
758         }
759         \bool_set_false:N \l__cmd_grab_expandably_bool
760     }
761     \bool_if:NTF \l__cmd_long_bool
762     { \bool_set_true:N \l__cmd_some_long_bool }
763     { \bool_set_true:N \l__cmd_some_short_bool }
764     \tl_put_right:Ne \l__cmd_arg_spec_tl
765     {
766         \bool_lazy_and:nnT
767         { \l__cmd_long_bool }
768         { ! \str_if_eq_p:nn {#1} { c } }
769         { + }
770         \bool_if:NT \l__cmd_obey_spaces_bool { ! }
771         \exp_not:n {#1}
772     }
773     \bool_set_false:N \l__cmd_long_bool
774     \bool_set_false:N \l__cmd_obey_spaces_bool
775 }

```

```

776 \cs_new_protected:Npn \__cmd_add_arg_spec_mandatory:n #1
777 { \__cmd_add_arg_spec_mandatory:nn { #1 } { #1 } }
778 \cs_new_protected:Npn \__cmd_add_arg_spec_mandatory:nn #1 #2
779 {
780   \bool_if:NT \l__cmd_some_obey_spaces_bool
781   {
782     \msg_error:nnee { cmd } { invalid-bang }
783     { \__cmd_environment_or_command: }
784     {
785       \bool_if:NTF \l__cmd_obey_spaces_bool
786       { \tl_to_str:n {'#2'} }
787       { an~optional~argument~before~mandatory~ \tl_to_str:n {'#2'} }
788     }
789     \__cmd_bad_def:wn
790   }
791   \tl_clear:N \l__cmd_last_delimiters_tl
792   \int_set_eq:NN \l__cmd_last_mandatory_arg_int \l__cmd_current_arg_int
793   \__cmd_add_arg_spec:n {#1}
794 }

```

(End of definition for __cmd_add_arg_spec:n, __cmd_add_arg_spec_mandatory:n, and
__cmd_add_arg_spec_mandatory:nn.)

1.24 Preparing the signature: general mechanism

```

\__cmd_prepare_signature:n
  \__cmd_prepare_signature_verb_chk:n
\__cmd_prepare_signature:N
  \__cmd_prepare_signature_bypass:N

```

Actually creating the signature uses the same loop approach as normalizing the signature. There are first a number of variables which need to be set to track what is going on. Many of these variables are unused when defining expandable commands.

```

795 \cs_new_protected:Npn \__cmd_prepare_signature:n #1
796 {
797   \int_set_eq:NN \l__cmd_total_args_int \l__cmd_current_arg_int
798   \int_zero:N \l__cmd_current_arg_int
799   \bool_set_false:N \l__cmd_long_bool
800   \bool_set_false:N \l__cmd_obey_spaces_bool
801   \bool_set_false:N \l__cmd_suppress_strip_bool
802   \int_zero:N \l__cmd_m_args_int
803   \bool_set_false:N \l__cmd_defaults_bool
804   \tl_clear:N \l__cmd_defaults_tl
805   \tl_clear:N \l__cmd_process_all_tl
806   \tl_clear:N \l__cmd_process_one_tl
807   \bool_set_false:N \l__cmd_process_some_bool
808   \tl_clear:N \l__cmd_signature_tl
809   \bool_set_false:N \l__cmd_final_verb_bool
810   \__cmd_prepare_signature_verb_chk:n {#1}
811   \__cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
812   \bool_if:NF \l__cmd_expandable_bool { \__cmd_flush_m_args: }
813 }

```

A quick check on the final arg. type, done using a string approach (that retains braces) to avoid an issue with {c} as the last balanced text.

```

814 \cs_new_protected:Npn \__cmd_prepare_signature_verb_chk:n #1
815 {
816   \tl_if_in:eeT
817   { \tl_trim_spaces:e { \tl_to_str:n {#1} } } \exp_not:N \q_stop }

```

```

818     { \tl_to_str:n { c } \exp_not:N \q_stop }
819     { \bool_set_true:N \l__cmd_final_verb_bool }
820 }

```

The main looping function does not take an argument, but carries out the reset on the processor boolean. This is split off from the rest of the process so that when actually setting up processors the flag-reset can be bypassed.

For each known argument type there is an appropriate function to actually do the addition to the signature. These are separate for expandable and standard functions, as the approaches are different.

```

821 \cs_new_protected:Npn \__cmd_prepare_signature:N
822 {
823   \bool_set_false:N \l__cmd_prefixed_bool
824   \__cmd_prepare_signature_bypass:N
825 }
826 \cs_new_protected:Npn \__cmd_prepare_signature_bypass:N #1
827 {
828   \quark_if_recursion_tail_stop:N #1
829   \use:c
830   {
831     __cmd_add
832     \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
833     _type_ \token_to_str:N #1 :w
834   }
835 }

```

(End of definition for __cmd_prepare_signature:n and others.)

1.25 Setting up a standard signature

Each argument-adding function appends to the signature a grabber (and for some types, the delimiters or default value), except the one for `m` arguments. These are collected and added to the signature all at once by `\__cmd_flush_m_args:`, called for every other argument type. All of the functions then call the loop function `\__cmd_prepare_signature:N`. Default values of arguments are collected by `\__cmd_add_default:n` rather than being stored with the argument; this function and `\__cmd_add_default:` are also responsible for keeping track of `\l__cmd_current_arg_int`.

`\__cmd_add_type_+:w` Making the next argument long means setting the flag. The `m` arguments are recorded here as this has to be done for every case where there is then a long argument.

```

836 \cs_new_protected:cpn { __cmd_add_type_+:w }
837 {
838   \__cmd_flush_m_args:
839   \bool_set_true:N \l__cmd_long_bool
840   \bool_set_true:N \l__cmd_prefixed_bool
841   \__cmd_prepare_signature_bypass:N
842 }

```

(End of definition for __cmd_add_type_+:w.)

`\__cmd_add_type_!:w` Much the same for controlling trailing optional arguments.

```

843 \cs_new_protected:cpn { __cmd_add_type_!:w }
844 {
845   \__cmd_flush_m_args:

```

```

846     \bool_set_true:N \l__cmd_obey_spaces_bool
847     \bool_set_true:N \l__cmd_prefixed_bool
848     \__cmd_prepare_signature_bypass:N
849 }

```

(End of definition for __cmd_add_type_!:w.)

__cmd_add_type_>:w When a processor is found, the processor code is stored. It will be used by **__cmd_args_process**: once arguments are all found. Here too the loop calls **__cmd_prepare_signature_bypass:N** rather than **__cmd_prepare_signature:N** so that the flag is not reset.

```

850 \cs_new_protected:cpn { __cmd_add_type_>:w } #1
851 {
852     \__cmd_flush_m_args:
853     \bool_set_true:N \l__cmd_prefixed_bool
854     \bool_set_true:N \l__cmd_process_some_bool
855     \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
856     \__cmd_prepare_signature_bypass:N
857 }

```

(End of definition for __cmd_add_type_>:w.)

__cmd_add_type_=:w A mix of the ideas from above: set a flag and add a processor.

```

858 \cs_new_protected:cpn { __cmd_add_type_=:w } #1
859 {
860     \__cmd_flush_m_args:
861     \bool_set_true:N \l__cmd_prefixed_bool
862     \bool_set_true:N \l__cmd_suppress_strip_bool
863     \bool_set_true:N \l__cmd_process_some_bool
864     \tl_put_left:Nn \l__cmd_process_one_tl
865     { { \__cmd_arg_to_keyvalue:nn {#1} } }
866     \__cmd_prepare_signature_bypass:N
867 }

```

(End of definition for __cmd_add_type_=:w.)

```

\__cmd_add_type_b:w
\__cmd_add_type_c:w
\__cmd_add_type_b_or_c:N
868 \cs_new_protected:Npn \__cmd_add_type_b:w
869 { \__cmd_add_type_b_or_c:N b }
870 \cs_new_protected:Npn \__cmd_add_type_c:w
871 { \__cmd_add_type_b_or_c:N c }
872 \cs_new_protected:Npn \__cmd_add_type_b_or_c:N #1
873 {
874     \__cmd_flush_m_args:
875     \__cmd_add_default:
876     \__cmd_add_grabber:N #1
877     \__cmd_prepare_signature:N
878 }

```

(End of definition for __cmd_add_type_b:w, __cmd_add_type_c:w, and __cmd_add_type_b_or_c:N.)

__cmd_add_type_D:w

```

879 \cs_new_protected:Npn \__cmd_add_type_D:w #1#2#3
880 {
881     \__cmd_flush_m_args:

```

```

882 \__cmd_add_default:n {#3}
883 \__cmd_add_grabber:N D
884 \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
885 \__cmd_prepare_signature:N
886 }

```

(End of definition for __cmd_add_type_D:w.)

`\__cmd_add_type_E:w` The E-type argument needs a special handling of default values. Since each embellishment is a separate argument, it also needs to replicate the argument processors for each embellishment argument so that the numbers of arguments and processors remain in sync.

```

887 \cs_new_protected:Npn \__cmd_add_type_E:w #1#2
888 {
889   \__cmd_flush_m_args:
890   \__cmd_add_default_E:nn {#1} {#2}
891   \use:e
892   {
893     \__cmd_replicate_processor:nn { \tl_count:n {#1} }
894     { \exp_not:o \l__cmd_process_one_tl }
895   }
896   \__cmd_add_grabber:N E
897   \tl_put_right:Nn \l__cmd_signature_tl { {#1} }
898   \__cmd_prepare_signature:N
899 }

```

(End of definition for __cmd_add_type_E:w.)

`\__cmd_replicate_processor:nn` In the command's argument processor signature (the final argument of `\__cmd_start:nNNnnn`) there is one braced item for each formal argument (up to nine), and in each of these items there is one braced item for each processor (as many as there were processors declared for a given argument). Something like this:

```

{ % argument processors
  { % argument 1
    { processor 1 } { processor 2 } ... { processor n }
  } % end argument 1
  { ... } % argument 2
  :
  { ... } % argument n
} % end argument processors

```

The function `\__cmd_add_grabber:N` adds one single grabber for an argument, and adds the braced item for that one argument. However, in an E-type argument each embellishment requires its own formal argument, so we need to break out of one layer of braces in `\l__cmd_process_one_tl`, add copies of the processor as necessary, and then return the removed brace. The function below does just that: it defines `\l__cmd_process_one_tl` starting with a `}_2` and ending with a `{_1`, so that it adds as many processors as needed when x-expanded.

```

900 \cs_new_protected:Npn \__cmd_replicate_processor:nn #1 #2

```

```

901 {
902   \int_compare:nNnF {#1} > { 1 } { \use_none:nnn }
903   \tl_set:Nx \l__cmd_process_one_tl
904     {
905       \exp_not:n { \exp_not:n {#2} \if_false: { \fi: } }
906       \prg_replicate:nn { #1 - 2 }
907       { \exp_not:n { \exp_not:n { {#2} } } }
908       \exp_not:n { { \if_false: } \fi: \exp_not:n {#2} }
909     }
910 }

```

(End of definition for __cmd_replicate_processor:nn.)

`\__cmd_add_type_m:w` The `m` type is special as short arguments which are not post-processed are simply counted at this stage. Thus there is a check to see if either of these cases apply. If so, a one-argument grabber is added to the signature. On the other hand, if a standard short argument is required it is simply counted at this stage, to be added later using `\__cmd_flush_m_args:`.

```

911 \cs_new_protected:Npn \__cmd_add_type_m:w
912 {
913   \__cmd_add_default:
914   \bool_if:NTF \l__cmd_prefixed_bool
915     { \__cmd_add_grabber:N m }
916     { \int_incr:N \l__cmd_m_args_int }
917   \__cmd_prepare_signature:N
918 }

```

(End of definition for __cmd_add_type_m:w.)

`\__cmd_add_type_R:w` The `R`-type argument is very similar to the `D`-type.

```

919 \cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
920 {
921   \__cmd_flush_m_args:
922   \__cmd_add_default:n {#3}
923   \__cmd_add_grabber:N R
924   \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
925   \__cmd_prepare_signature:N
926 }

```

(End of definition for __cmd_add_type_R:w.)

`\__cmd_add_type_t:w` Setting up a `t` argument means collecting one token for the test, and adding it along with the grabber to the signature.

```

927 \cs_new_protected:Npn \__cmd_add_type_t:w #1
928 {
929   \__cmd_flush_m_args:
930   \__cmd_add_default:
931   \__cmd_add_grabber:N t
932   \tl_put_right:Nn \l__cmd_signature_tl {#1}
933   \__cmd_prepare_signature:N
934 }

```

(End of definition for __cmd_add_type_t:w.)

`\__cmd_add_type_v:w` At this stage, the `v` argument is identical to `l` except that since the grabber may fail to read a verbatim argument we need a default value.

```

935 \cs_new_protected:Npn \__cmd_add_type_v:w
936 {
937   \__cmd_flush_m_args:
938   \__cmd_add_default:n { \NoValue }
939   \__cmd_add_grabber:N v
940   \__cmd_prepare_signature:N
941 }

```

(End of definition for __cmd_add_type_v:w.)

`\__cmd_flush_m_args:` As `m` arguments are simply counted, there is a need to add them to the token register in a block. As this function can only be called if something other than `m` turns up, the flag can be switched here.

```

942 \cs_new_protected:Npn \__cmd_flush_m_args:
943 {
944   \int_compare:nNnT \l__cmd_m_args_int > 0
945   {
946     \tl_put_right:Ne \l__cmd_signature_tl
947     { \exp_not:c { __cmd_grab_m_ \int_use:N \l__cmd_m_args_int :w } }
948     \tl_put_right:Ne \l__cmd_process_all_tl
949     { \prg_replicate:nn { \l__cmd_m_args_int } { { } } }
950   }
951   \int_zero:N \l__cmd_m_args_int
952 }

```

(End of definition for __cmd_flush_m_args:.)

`\__cmd_add_grabber:N` To keep the various checks needed in one place, adding the grabber to the signature is done here. The only questions are whether the grabber should be long or not, and whether to obey spaces. The `\l__cmd_obey_spaces_bool` boolean can only be `true` for trailing optional arguments. In that case spaces will not be ignored when looking for that optional argument.

```

953 \cs_new_protected:Npn \__cmd_add_grabber:N #1
954 {
955   \tl_put_right:Ne \l__cmd_signature_tl
956   {
957     \exp_not:c
958     {
959       __cmd_grab_ #1
960       \bool_if:NT \l__cmd_long_bool { _long }
961       \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
962       \bool_lazy_and:nnT
963       { \l__cmd_suppress_strip_bool }
964       { \str_if_eq_p:nn {#1} { D } }
965       { _no_strip }
966       \bool_lazy_all:nT
967       {
968         { \l__cmd_final_verb_bool }
969         { \str_if_eq_p:nn {#1} { D } }
970       }
971       \int_compare_p:nNn \l__cmd_current_arg_int
972       > \l__cmd_last_mandatory_arg_int

```



```

973         }
974     }
975     { _verb_safe }
976     :w
977 }
978 }
979 \bool_set_false:N \l__cmd_long_bool
980 \bool_set_false:N \l__cmd_obey_spaces_bool
981 \bool_set_false:N \l__cmd_suppress_strip_bool
982 \bool_set_false:N \l__cmd_verb_safe_bool
983 \tl_put_right:Ne \l__cmd_process_all_tl
984 {
985     {
986         \if_charcode:w E #1 \use_i:nn \fi:
987         \exp_not:o \l__cmd_process_one_tl
988     }
989 }
990 \tl_clear:N \l__cmd_process_one_tl
991 }

```

(End of definition for __cmd_add_grabber:N.)

__cmd_add_default:n
 __cmd_add_default:
 __cmd_add_default_E:nn

Store the default value of an argument, or rather code that gives that default value (it may involve other arguments). This is (in a brace group) \prg_do_nothing: followed by a default value or \NoValue. For E-type arguments, pad the defaults #2 with some \NoValue until there are as many as embellishments #1. These functions are also used when defining expandable commands. The \prg_do_nothing: here are removed in __cmd_defaults_def:nnn: for a user-supplied default the token is injected to avoid brace stripping, and for \NoValue means we have a common code path later.

```

992 \cs_new_protected:Npn \__cmd_add_default:n #1
993 {
994     \tl_if_novalue:nTF {#1}
995     { \__cmd_add_default: }
996     {
997         \int_incr:N \l__cmd_current_arg_int
998         \bool_set_true:N \l__cmd_defaults_bool
999         \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
1000     }
1001 }
1002 \cs_new_protected:Npn \__cmd_add_default:
1003 {
1004     \int_incr:N \l__cmd_current_arg_int
1005     \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: \NoValue } }
1006 }
1007 \cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
1008 {
1009     \tl_map_function:nN {#2} \__cmd_add_default:n
1010     \prg_replicate:nn
1011     { \tl_count:n {#1} - \tl_count:n {#2} }
1012     { \__cmd_add_default: }
1013 }

```

(End of definition for __cmd_add_default:n, __cmd_add_default:, and __cmd_add_default_E:nn.)

1.26 Setting up expandable types

The approach here is not dissimilar to that for standard types, but fewer types are supported. There is also a need to define the per-function auxiliaries: this is done here, while the general grabbers are dealt with later.

`\_cmd_add_expandable_type_+ :w` We have already checked that short arguments are before long arguments, so `\l\_cmd_long_bool` only changes from `false` to `true` once (and there is no need to reset it after each argument). Continue the loop.

```
1014 \cs_new_protected:cpn { \_cmd_add_expandable_type_+ :w }
1015 {
1016   \bool_set_true:N \l\_cmd_long_bool
1017   \_cmd_prepare_signature:N
1018 }
```

(End of definition for _cmd_add_expandable_type_+ :w.)

`\_cmd_add_expandable_type_D :w`
`\_cmd_add_expandable_type_D_aux :NNNn`
`\_cmd_add_expandable_type_D_aux :NNN`
`\_cmd_add_expandable_type_D_aux :NN` The set up for D-type arguments involves constructing a rather complex auxiliary which is used repeatedly when grabbing. There is an auxiliary here so that the R-type can share code readily: #1 is D or R. The `_aux :NN` auxiliary is needed if the two delimiting tokens are identical: in contrast to the non-expandable route, the grabber here has to act differently for this case.

```
1019 \cs_new_protected:Npn \_cmd_add_expandable_type_D :w
1020 { \_cmd_add_expandable_type_D_aux :NNNn D }
1021 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux :NNNn #1#2#3#4
1022 {
1023   \_cmd_add_default:n {#4}
1024   \tl_if_eq:nnTF {#2} {#3}
1025     { \_cmd_add_expandable_type_D_aux :NN #1 #2 }
1026     { \_cmd_add_expandable_type_D_aux :NNN #1 #2 #3 }
1027   \_cmd_prepare_signature:N
1028 }
1029 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux :NNN #1#2#3
1030 {
1031   \bool_if:NTF \l\_cmd_long_bool
1032     { \cs_set:cpe }
1033     { \cs_set_nopar:cpe }
1034     { \l\_cmd_expandable_aux_name_tl } ##1 ##2 #2 ##3 \q\_cmd ##4 #3
1035     { ##1 {##2} {##3} {##4} }
1036   \_cmd_add_expandable_grabber:nn {#1}
1037   {
1038     \exp_not:c { \l\_cmd_expandable_aux_name_tl }
1039     \exp_not:n { #2 #3 }
1040   }
1041 }
1042 \cs_new_protected:Npn \_cmd_add_expandable_type_D_aux :NN #1#2
1043 {
1044   \bool_if:NTF \l\_cmd_long_bool
1045     { \cs_set:cpe }
1046     { \cs_set_nopar:cpe }
1047     { \l\_cmd_expandable_aux_name_tl } ##1 #2 ##2 #2
1048     { ##1 {##2} }
1049   \_cmd_add_expandable_grabber:nn { #1_alt }
1050   {
```

```

1051         \exp_not:c { \l__cmd_expandable_aux_name_tl }
1052         \exp_not:n {#2}
1053     }
1054 }

```

(End of definition for __cmd_add_expandable_type_D:w and others.)

__cmd_add_expandable_type_E:w For each embellishment, use __cmd_get_grabber:NN to obtain an auxiliary delimited by that token and store a pair constituted of the auxiliary and the token in \l__cmd_tmpb_tl, before appending the whole set of these pairs to the signature, and an equal number of \NoValue markers (regardless of the default values of arguments). Set the current argument appropriately.

```

1055 \cs_new_protected:Npn \__cmd_add_expandable_type_E:w #1#2
1056 {
1057     \__cmd_add_default_E:nn {#1} {#2}
1058     \tl_clear:N \l__cmd_tmpb_tl
1059     \tl_map_function:nN {#1} \__cmd_add_expandable_type_E_aux:n
1060     \__cmd_add_expandable_grabber:nn
1061     { E \bool_if:NT \l__cmd_long_bool { _long } }
1062     {
1063         { \exp_not:o \l__cmd_tmpb_tl }
1064         {
1065             \prg_replicate:nn { \tl_count:n {#1} }
1066             { { \NoValue } }
1067         }
1068     }
1069     \__cmd_prepare_signature:N
1070 }
1071 \cs_new_protected:Npn \__cmd_add_expandable_type_E_aux:n #1
1072 {
1073     \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
1074     \tl_put_right:Ne \l__cmd_tmpb_tl
1075     { \exp_not:o \l__cmd_tmpa_tl \exp_not:N #1 }
1076 }

```

(End of definition for __cmd_add_expandable_type_E:w and __cmd_add_expandable_type_E_aux:n.)

__cmd_add_expandable_type_m:w Unlike the standard case, when working expandably each argument is always grabbed separately.

```

1077 \cs_new_protected:Npn \__cmd_add_expandable_type_m:w
1078 {
1079     \__cmd_add_default:
1080     \__cmd_add_expandable_grabber:nn
1081     { m \bool_if:NT \l__cmd_long_bool { _long } } { }
1082     \__cmd_prepare_signature:N
1083 }

```

(End of definition for __cmd_add_expandable_type_m:w.)

__cmd_add_expandable_type_R:w The R-type is very similar to the D-type argument, and so the same internals are used.

```

1084 \cs_new_protected:Npn \__cmd_add_expandable_type_R:w
1085 { \__cmd_add_expandable_type_D_aux:NNNn R }

```

(End of definition for __cmd_add_expandable_type_R:w.)

_cmd_add_expandable_type_t:w An auxiliary delimited by #1 is built now. It will be used to test for the presence of that token.

```
1086 \cs_new_protected:Npn \_cmd_add_expandable_type_t:w #1
1087 {
1088   \_cmd_add_default:
1089   \_cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
1090   \_cmd_add_expandable_grabber:nn { t }
1091   {
1092     \exp_not:o \l__cmd_tmpa_tl
1093     \exp_not:N #1
1094   }
1095   \_cmd_prepare_signature:N
1096 }
```

(End of definition for _cmd_add_expandable_type_t:w.)

_cmd_add_expandable_grabber:nn This is called for all arguments to place the right grabber in the signature.

```
1097 \cs_new_protected:Npn \_cmd_add_expandable_grabber:nn #1#2
1098 {
1099   \tl_put_right:Ne \l__cmd_signature_tl
1100   { \exp_not:c { __cmd_expandable_grab_ #1 :w } #2 }
1101 }
```

(End of definition for _cmd_add_expandable_grabber:nn.)

_cmd_get_grabber:NN Given a token #1, defines an expandable function delimited by that token and stores it in the token list #2. The function is named after the token, unless that function name is already taken by some other grabber (this can happen in the rare case where delimiters with different category codes are used in the same document): in that case use a global counter to get a unique name. Since the grabbers are not named after xparse commands they should not be used to get material from the input stream.

_cmd_get_grabber_auxi:NN
_cmd_get_grabber_auxii:NN

```
1102 \cs_new_protected:Npn \_cmd_get_grabber:NN #1#2
1103 {
1104   \cs_set:Npn \_cmd_tmp:w ##1 #1 {##1}
1105   \exp_args:Nc \_cmd_get_grabber_auxi:NN
1106   { __cmd_grabber_ \token_to_str:N #1 :w } #2
1107 }
1108 \cs_new_protected:Npn \_cmd_get_grabber_auxi:NN #1#2
1109 {
1110   \cs_if_eq:NNTF \_cmd_tmp:w #1
1111   { \tl_set:Nn #2 {#1} }
1112   {
1113     \cs_if_exist:NTF #1
1114     {
1115       \int_gincr:N \g__cmd_grabber_int
1116       \exp_args:Nc \_cmd_get_grabber_auxi:NN
1117       {
1118         __cmd_grabber_
1119         - \int_use:N \g__cmd_grabber_int :w
1120       }
1121       #2
1122     }
1123     { \_cmd_get_grabber_auxii:NN #1 #2 }
1124   }
```

```

1125 }
1126 \cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
1127 {
1128   \cs_set_eq:NN #1 \__cmd_tmp:w
1129   \tl_set:Nn #2 {#1}
1130 }

```

(End of definition for __cmd_get_grabber:NN, __cmd_get_grabber_auxi:NN, and __cmd_get_grabber_auxii:NN.)

1.26.1 Copying a command and its internal structure

```

1131 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_copy:NN}%
1132 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}

```

Since the 2020-10-01 L^AT_EX 2_ε release, support for copying, and showing the definition of, robust commands has been available, but the specifics of each command are implemented separately. Here we'll add support for copying and showing ltxcmd definitions.

To fully support copying, we need two commands: a conditional to test if a command is in fact a ltxcmd command, and another command to actually copy the command. The conditional is defined later as __kernel_cmd_if_xparse:NTF, so now to the copying:

This macro just branches to the proper copying command by using __cmd_cmd_type_cases:NnnnnnF. The copying command takes the names of the commands to be copied to and from, and the actual commands as its four arguments.

```

\__cmd_copy:NN
\__cmd_set_eq_if_exist:NN
\__cmd_set_eq_if_exist:cc

```

```

1133 \cs_new_protected:Npn \__cmd_copy:NN #1 #2
1134 {
1135   \use:e
1136   {
1137     \int_set:Nn \tex_escapechar:D { 92 }
1138     \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #2
1139     { \__cmd_copy_command:nnNN }
1140     { \__cmd_copy_expandable:nnNN }
1141     { \__cmd_copy_optimized:nnNN }
1142     { \__cmd_copy_environment:nnNN }
1143     { \__cmd_copy_environment_end:nnNN }
1144     { \__cmd_cant_copy:nwn { non-ltxcmd } }
1145     { \cs_to_str:N #1 } { \cs_to_str:N #2 }
1146     \exp_not:N #1 \exp_not:N #2
1147     \exp_not:N \__cmd_break_point:n { \cs_to_str:N #2 }
1148     \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
1149   }
1150 }
1151 \cs_new_protected:Npn \__cmd_set_eq_if_exist:NN #1 #2
1152 { \cs_if_exist:NTF #2 { \cs_set_eq:NN } { \use_none:nn } #1 #2 }
1153 \cs_generate_variant:Nn \__cmd_set_eq_if_exist:NN { cc }

```

An utility macro similar to __cmd_bad_def:wn to abort a command copy. Contrary to __cmd_bad_def:wn though, when this happens the issue is most likely internal, because the command was already (supposedly) correctly defined so it should be copyable. Hopefully this macro will never be used ever, but if it does, apologize and give the reason for the failure so the user can report.

```

\__cmd_cant_copy:nwn

```

```

1154 \cs_new_protected:Npn \__cmd_cant_copy:nwn #1 #2 \__cmd_break_point:n #3
1155 { \msg_error:nnnn { cmd } { copy-bug } {#1} {#3} }

```

```

1156 \msg_new:nnn { cmd } { copy-bug }
1157 {
1158   Error~while~copying~command~\iow_char:N\|#2:\
1159   \str_case:nn {#1}
1160   {
1161     { non-ltcmd } { Command~is~not~a~valid~ltcmd~command. }
1162     { unknown-type } { Found~an~unknown~argument~type. }
1163     { invalid-end }
1164     { Target~command~is~not~named~\iow_char:N \end<name>. }
1165   }
1166 }

```

And, of course, add `\__kernel_cmd_if_xparse:NTF` and `\__cmd_copy:NN` to `\@declarecommandcopylisthook`:

```

1167 \tl_gput_right:Nn \@declarecommandcopylisthook
1168 { { \__kernel_cmd_if_xparse:NTF \__cmd_copy:NN } }

```

(End of definition for `\__cmd_copy:NN`, `\__cmd_set_eq_if_exist:NN`, and `\__cmd_cant_copy:nwn`.)

`\__cmd_copy_command:nnNN`
`\__cmd_copy_command:NnNNnnnn`

A normal (non-expandable) command has a pretty straightforward structure. Its definition is stored in `\<cmd>_code`, its defaults (if any) are stored in `\<cmd>_defaults`, and its top-level definition contains its signature, which can just be copied over. `\__cmd_copy_command:nnNN` copies the command code and defaults, and then defines the top-level command using the auxiliary `\__cmd_copy_command:NnNNnnnn`. This macro takes the signature of the command being copied from its top-level definition, and replaces the named bits with the new name.

```

1169 \cs_new_protected:Npn \__cmd_copy_command:nnNN #1 #2 #3 #4
1170 {
1171   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1172   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1173   \cs_set_protected_nopar:Npe #3
1174   { \exp_after:wN \__cmd_copy_command:NnNNnnnn #4 {#1} }
1175 }
1176 \cs_new:Npn \__cmd_copy_command:NnNNnnnn #1 #2 #3 #4 #5 #6 #7 #8
1177 {
1178   #1 \exp_not:n { {#2} }
1179   \exp_not:c { #8 ~ } \exp_not:c { #8 ~ code }
1180   \exp_not:n { {#5} {#6} {#7} }
1181 }

```

(End of definition for `\__cmd_copy_command:nnNN` and `\__cmd_copy_command:NnNNnnnn`.)

`\__cmd_copy_expandable:nnNN`
`\__cmd_copy_expandable:NnNNNNnnnn`

An expandable command is slightly more complicated. Besides the `\<cmd>_code`, and `\<cmd>_defaults`, it also has an auxiliary `\<cmd>_` for grabbing delimited arguments, and possibly another auxiliary `\<cmd>_`, if the command has both long and short arguments. Then, its signature also has several specific bits that are unique to that command; this is in contrast to non-expandable commands, which use a common set of parsing functions.

We start by copying the basics, then call `\__cmd_copy_expandable_signature:NnNNNNnnnn` to parse the signature of the command and build up the modified copy in a temporary token list, then we call `\__cmd_copy_expandable:NnNNNNnnnn` that will copy the top-level definition of the command, with the proper internal renames.

```

1182 \<latexrelease>\EndIncludeInRelease
1183 \<latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_copy:NN}%

```

```

1184 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1185 <latexrelease> \EndIncludeInRelease

```

There's one variant: a command begins with `\__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```

1186 <latexrelease> \IncludeInRelease{2023/06/01}{\__cmd_copy_expandable:nnNN}%
1187 <latexrelease> {Distinguish~non-expandable~document~commands}
1188 \cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
1189 {
1190   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1191   \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1192   \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1193   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1194   \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1195   \token_if_protected_macro:NTF #4
1196   { \cs_set_protected_nopar:Npe } { \cs_set_nopar:Npe }
1197   #3
1198   { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1199 }
1200 <latexrelease> \EndIncludeInRelease
1201 <latexrelease> \IncludeInRelease{2021/11/15}{\__cmd_copy_expandable:nnNN}%
1202 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1203 <latexrelease> \cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
1204 <latexrelease> {
1205   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1206   \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1207   \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1208   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1209   \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1210   \cs_set_nopar:Npx #3
1211   { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1212 <latexrelease> }
1213 <latexrelease> \EndIncludeInRelease
1214 <latexrelease> \IncludeInRelease{2020/10/01}{\__cmd_copy_expandable:nnNN}%
1215 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1216 <latexrelease> \EndIncludeInRelease

```

Copy the code, simply define the wrapper.

```

1217 \cs_new_protected:Npn \__cmd_copy_optimized:nnNN #1#2#3#4
1218 {
1219   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1220   \token_if_protected_macro:NTF #4
1221   { \cs_set_protected_nopar:Npe }
1222   { \cs_set_nopar:Npe }
1223   #3
1224   {
1225     \exp_not:N \__cmd_start_optimized:
1226     \exp_not:c { #1 ~ code }
1227   }
1228 }
1229 <latexrelease> \IncludeInRelease{2021/11/15}{\__cmd_copy:NN (part 2)}%
1230 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}

```

```

1231 \cs_new:Npn \__cmd_copy_expandable:NnNNNNnnn #1 #2 #3 #4 #5 #6 #7 #8 #9
1232 {
1233   \exp_not:N #1 \exp_not:n { {#2} }
1234   \exp_not:c { #8 ~ }
1235   \exp_not:c
1236   {
1237     #8 ~
1238     \str_if_eq:eeT
1239     { \exp_not:c { #9 ~ \c_space_tl } } { \exp_not:N #4 }
1240     { \c_space_tl }
1241   }
1242   \exp_not:c { #8 ~ code }
1243   \str_if_eq:eeTF { \exp_not:N #6 } { ? }
1244   { ? }
1245   { \exp_not:c { #8 ~ defaults } }
1246   { \exp_not:V \l__cmd_tmpa_tl }
1247 }

```

A signature for an expandable command contains as many `\expandable_grab_⟨type⟩:w` as there are arguments, and what follows this macro depends on the `⟨type⟩`. We'll start a loop through the signature, and at each argument grabber, we'll step the argument count, and look for the `⟨type⟩` with `\__cmd_copy_parse_grabber:w` so that we know which `\__cmd_copy_grabber_⟨type⟩:w` to call next.

```

1248 \cs_new_protected:Npn \__cmd_copy_expandable_signature:NnNNNNnnn
1249   #1 #2 #3 #4 #5 #6 #7 #8 #9
1250 {
1251   \int_zero:N \l__cmd_current_arg_int
1252   \tl_clear:N \l__cmd_tmpa_tl
1253   \__cmd_copy_expandable:nnN {#8} {#9} #7
1254   \q_recursion_tail \q_recursion_stop
1255 }
1256 \cs_new_protected:Npn \__cmd_copy_expandable:nnN #1 #2 #3
1257 {
1258   \quark_if_recursion_tail_stop:n {#3}
1259   \int_incr:N \l__cmd_current_arg_int
1260   \exp_after:wN \__cmd_copy_parse_grabber:w \token_to_str:N #3 {#1} {#2}
1261 }
1262 \use:e
1263 {
1264   \cs_new_protected:Npn \exp_not:N \__cmd_copy_parse_grabber:w #1
1265   \tl_to_str:n { expandable_grab_ } #2 \tl_to_str:n { :w }
1266   {
1267     \tl_put_right:Ne \exp_not:N \l__cmd_tmpa_tl
1268     { \exp_not:N \exp_not:c { __cmd_expandable_grab_#2:w } }
1269     \exp_not:N \cs_if_exist_use:cF { __cmd_copy_grabber_#2:w }
1270     { \__cmd_cant_copy:nwn { unknown-type } }
1271   }
1272 }

```

The most complicated is the Delimited argument: each argument has a dedicated grabbing function named after the command that has to be copied over (of the form `\⟨cmd⟩_⟨arg⟩_⟨num⟩`)).

```

1273 \cs_new_protected:Npn \__cmd_copy_grabber_D:w #1 #2 #3 #4 #5

```

```

\__cmd_copy_grabber_D:w
\__cmd_copy_grabber_D_alt:w
\__cmd_copy_grabber_R:w
\__cmd_copy_grabber_R_alt:w
\__cmd_copy_grabber_E:w
\__cmd_copy_grabber_E_long:w
\__cmd_copy_grabber_t:w
\__cmd_copy_grabber_m:w
\__cmd_copy_grabber_m_long:w

```



```

1274 {
1275   \tl_put_right:Ne \l__cmd_tmpa_tl
1276   {
1277     \exp_not:c { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1278     \exp_not:n { #4 #5 }
1279   }
1280   \cs_set_eq:cc
1281     { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1282     { #2 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1283   \__cmd_copy_expandable:nnN {#1} {#2}
1284 }

```

D_{alt} is just a special case of D that uses a single delimiter (used when both delimiters of the argument are identical):

```

1285 \cs_new_protected:Npn \__cmd_copy_grabber_D_alt:w #1 #2 #3 #4
1286 { \__cmd_copy_grabber_D:w {#1} {#2} {#3} {#4} { } }

```

As far as copying is concerned, R is identical to D:

```

1287 \cs_new_eq:NN \__cmd_copy_grabber_R:w \__cmd_copy_grabber_D:w
1288 \cs_new_eq:NN \__cmd_copy_grabber_R_alt:w \__cmd_copy_grabber_D_alt:w

```

E is straightforward: we just copy the embellishments over, and increase the current argument number `\l__cmd_current_arg_int` by the number of embellishments (minus one because there is a `\int_incr:N` down the line).

```

1289 \cs_new_protected:Npn \__cmd_copy_grabber_E:w #1 #2 #3 #4
1290 {
1291   \tl_put_right:Nn \l__cmd_tmpa_tl { {#3} {#4} }
1292   \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#4} - 1 }
1293   \__cmd_copy_expandable:nnN {#1} {#2}
1294 }
1295 \cs_new_eq:NN \__cmd_copy_grabber_E_long:w \__cmd_copy_grabber_E:w

```

t just needs copying the token to be tested for:

```

1296 \cs_new_protected:Npn \__cmd_copy_grabber_t:w #1 #2 #3 #4
1297 {
1298   \tl_put_right:Nn \l__cmd_tmpa_tl { #3 #4 }
1299   \__cmd_copy_expandable:nnN {#1} {#2}
1300 }

```

And last but not least, m is the simplest; the grabber is just `\__cmd_expandable_grab_m:w`, which is already added to the new command so here we just resume the loop:

```

1301 \cs_new_protected:Npn \__cmd_copy_grabber_m:w { \__cmd_copy_expandable:nnN }
1302 \cs_new_eq:NN \__cmd_copy_grabber_m_long:w \__cmd_copy_grabber_m:w

```

(End of definition for __cmd_copy_expandable:nnNN and others.)

`\__cmd_copy_environment:nnNN` Copying an environment's `\begin` part is pretty much like copying a command, except it has a longer name, and at the end we have to copy `\environment <name>` into `\<name>`.

```

1303 \cs_new_protected:Npn \__cmd_copy_environment:nnNN #1 #2 #3 #4
1304 {
1305   \cs_set_eq:cc { environment~ #1 ~ code } { environment~ #2 ~ code }
1306   \__cmd_set_eq_if_exist:cc
1307     { environment~ #1 ~ defaults } { environment~ #2 ~ defaults }
1308   \cs_set_protected_nopar:cpe { environment~ #1 }
1309   { \exp_after:wN \__cmd_copy_environment:Nnnnnnn #4 {#1} }
1310   \cs_set_eq:cc {#1} { environment~ #1 }

```

```

1311 }
1312 \cs_new:Npn \__cmd_copy_environment:Nnnnnnn #1 #2 #3 #4 #5 #6 #7
1313 { #1 \exp_not:n { {#2} } {#7} \exp_not:n { {#4} {#5} {#6} } }

(End of definition for \__cmd_copy_environment:nnNN and \__cmd_copy_environment:Nnnnnnn.)

```

`\__cmd_copy_environment_end:nnNN`
`\__cmd_copy_environment_end_aux:nnNN`

Copying an environment's `\end` part is a bit trickier. We first have to make sure that both parts are named `\end⟨name⟩` (that's actually not a hard requirement, but an environment `\end` command makes no sense without the `end` in its name), and strip the leading `end` from the strings. After that, copying is straightforward.

```

1314 \cs_new_protected:Npn \__cmd_copy_environment_end:nnNN #1 #2
1315 {
1316   \__cmd_check_end:Nn \l__cmd_tmpa_tl {#1}
1317   \__cmd_check_end:Nn \l__cmd_tmpb_tl {#2}
1318   \exp_args:Noo \__cmd_copy_environment_end_aux:nnNN
1319   { \l__cmd_tmpa_tl } { \l__cmd_tmpb_tl }
1320 }
1321 \cs_new_protected:Npn \__cmd_copy_environment_end_aux:nnNN #1 #2 #3 #4
1322 {
1323   \cs_set_nopar:cpe { environment~ #1 ~end }
1324   { \exp_not:c { environment~ #1 ~end~aux } }
1325   \cs_set_eq:cc
1326   { environment~ #1 ~end~aux~ } { environment~ #2 ~end~aux~ }
1327   \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
1328 }

```

To check whether an `\end` command is valid, we look for the string `end` at the beginning of the command name, and if not found, raise an error:

```

\__cmd_check_end:Nn 1329 \cs_new_protected:Npn \__cmd_check_end:Nn #1 #2
\__cmd_check_end:n 1330 {
\__cmd_check_end:w 1331   \tl_set:Ne #1 { \__cmd_check_end:n {#2} }
1332   \token_if_eq_meaning:NNT #1 \q_nil
1333   { \__cmd_cant_copy:nwn { invalid-end } }
1334 }
1335 \cs_set_protected:Npn \__cmd_tmp:w #1
1336 {
1337   \cs_new:Npn \__cmd_check_end:n ##1
1338   {
1339     \exp_after:wN \__cmd_check_end:w \tl_to_str:n {##1}
1340     #1 \q_mark #1 \q_stop
1341   }
1342   \cs_new:Npn \__cmd_check_end:w ##1 #1 ##2 #1 ##3 \q_stop
1343   { \if_meaning:w ##2 \q_mark \exp_not:N \q_nil \else: ##2 \fi: }
1344 }
1345 \exp_args:No \__cmd_tmp:w { \tl_to_str:n { end } }

```

(End of definition for `\__cmd_copy_environment_end:nnNN` and others.)

Not much to do regarding `latexrelease`: we could remove the entries from `\@declarecommandcopylist` but it doesn't seem worth it.

```

1346 <latexrelease>\EndIncludeInRelease
1347 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_copy:NN (part 2)}%
1348 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1349 <latexrelease>\EndIncludeInRelease

```

1.26.2 Showing the definition of a command

```
1350 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show:N}%
1351 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
```

To show the definition of a command we need more or less the same building blocks as for copying, except that instead of making a copy, we'll just print stuff to the terminal. This macro just branches to the proper showing command by using `\__cmd_cmd_type_cases:NnnnnnF`. The showing command takes the command to be shown as argument.

`\__cmd_show:N`

```
1352 \cs_new_protected:Npn \__cmd_show:N #1
1353 {
1354   \use:e
1355   {
1356     \int_set:Nn \tex_escapechar:D { 92 }
1357     \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #1
1358     { \__cmd_show_command:N }
1359     { \__cmd_show_expandable:N }
1360     { \__cmd_show_optimized:N }
1361     { \__cmd_show_environment:N }
1362     { \__cmd_show_environment_end:N }
1363     { \__cmd_cant_copy:nwn { non-ltxcmd } }
1364     \exp_not:N #1
1365     \exp_not:N \__cmd_break_point:n { \cs_to_str:N #1 }
1366     \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
1367   }
1368 }
```

(End of definition for __cmd_show:N.)

`\__cmd_show_command:N`
`\__cmd_show_command:NnnNNwN`
`\__cmd_show_expandable:N`
`\__cmd_show_expandable:NnnNNNnN`
`\__cmd_show_optimized:N`
`\__cmd_show_command_aux:NnnNn`
`\__cmd_show_environment:N`
`\__cmd_show:e`

These commands just expand the command once to reveal its innards, then pass the type of command, the control sequence, the signature, and the code macro to `\__cmd_show_command_aux:NnnNn`.

```
1369 \cs_new_protected:Npn \__cmd_show_command:N #1
1370 { \exp_after:wN \__cmd_show_command:NnnNNwN #1 \q__cmd #1 }
1371 \cs_new_protected:Npn \__cmd_show_command:NnnNNwN #1 #2 #3 #4 #5 \q__cmd #6
1372 {
1373   \__cmd_show_command_aux:NnnNn \tl_show:e
1374   { document~command } #6 #4 {#2}
1375 }
1376 \cs_new_protected:Npn \__cmd_show_expandable:N #1
1377 { \exp_after:wN \__cmd_show_expandable:NnnNNNNnN #1 #1 }
1378 <latexrelease>\EndIncludeInRelease
1379 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_show:N}%
1380 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
1381 <latexrelease>\EndIncludeInRelease
```

There's one variant: a command begins with `\__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```
1382 <latexrelease>\IncludeInRelease{2023/06/01}{\__cmd_show_expandable:NnnNNNNnN}%
1383 <latexrelease> {Distinguish~non-expandable~document~commands}
1384 \cs_new_protected:Npn \__cmd_show_expandable:NnnNNNNnN #1 #2 #3 #4 #5 #6 #7 #8
1385 {
1386   \exp_args:NNe \__cmd_show_command_aux:NnnNn \tl_show:e
```

```

1387 { \token_if_protected_macro:NF #8 { expandable~ } document~command }
1388 #8 #5 {#2}
1389 }
1390 <latexrelease>\EndIncludeInRelease
1391 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show_expandable:NnNNNNnN}%
1392 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
1393 <latexrelease>\cs_new_protected:Npn \__cmd_show_expandable:NnNNNNnN #1 #2 #3 #4 #5 #6 #7 #8
1394 <latexrelease> {
1395 <latexrelease> \__cmd_show_command_aux:NnNNn \tl_show:x
1396 <latexrelease> { expandable~document~command } #8 #5 {#2}
1397 <latexrelease> }
1398 <latexrelease>\EndIncludeInRelease
1399 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_show_expandable:NnNNNNnN}%
1400 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
1401 <latexrelease>\EndIncludeInRelease
1402 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show:N (part 2)}%
1403 <latexrelease> {Support~\ShowCommand~in~ltxcmd}

```

Now just print everything in the required format. The auxiliary `\__cmd_split_signature:n` stores a ready-to-print token list in `\l__cmd_tmpa_tl`, so we use that here:

```

1404 \cs_new_protected:Npn \__cmd_show_command_aux:NnNNn #1 #2 #3 #4 #5
1405 {
1406   \__cmd_split_signature:n {#5}
1407   #1
1408   {
1409     \token_to_str:N #3 = #2: \iow_newline:
1410     \tl_use:N \l__cmd_tmpa_tl
1411     -> \cs_replacement_spec:N #4
1412   }
1413 }

```

Optimized functions need things done a bit differently as we need to reconstruct the argument spec.

```

1414 \cs_new_protected:Npn \__cmd_show_optimized:N #1
1415 {
1416   \exp_args:Nc \__cmd_show_optimized:NN
1417   { \cs_to_str:N #1 \c_space_tl code }
1418   #1
1419 }
1420 \cs_new_protected:Npn \__cmd_show_optimized:NN #1#2
1421 {
1422   \cs_set:Npe \__cmd_show_optimized_aux:N ##1
1423   {
1424     \c_space_tl \c_space_tl \c_hash_str ##1 :
1425     \bool_lazy_or:nnT
1426     { \token_if_long_macro_p:N #1 }
1427     { \token_if_protected_long_macro_p:N #1 }
1428     { + } m
1429     \iow_newline:
1430   }
1431   \tl_show:e
1432   {
1433     \token_to_str:N #2 =

```

```

1434     \bool_lazy_or:nnF
1435     { \token_if_protected_macro_p:N #1 }
1436     { \token_if_protected_long_macro_p:N #1 }
1437     { expandable ~ } document~command:
1438 \iow_newline:
1439 \int_step_function:nN
1440 {
1441     \int_div_truncate:nn
1442     { \tl_count:e { \cs_parameter_spec:N #1 } }
1443     { 2 }
1444 }
1445 \__cmd_show_optimized_aux:N
1446 ->
1447 \cs_replacement_spec:N #1
1448 }
1449 }

```

We can reuse most of the above to show an environment, except that we need to ensure that the proper `\environment ...` are passed to `\__cmd_show_command_aux:NnNNn`. Additionally, when `\ShowCommand\foo` is used (if `foo` is an environment), we show `\endfoo` as well, and when `\ShowCommand\endfoo` is used, change that to `\ShowCommand\foo` and do the same.

```

1450 \cs_new_protected:Npn \__cmd_show_environment:N #1
1451 {
1452     \exp_after:wN \__cmd_show_environment:Nnnw #1 \q__cmd
1453     \tl_show:e
1454     {
1455         \token_to_str:N \end { \cs_to_str:N #1 } : \iow_newline:
1456         -> \exp_args:Nc \cs_replacement_spec:N
1457         { environment~ \cs_to_str:N #1 ~end~aux~ }
1458     }
1459 }
1460 \cs_new_protected:Npn \__cmd_show_environment:Nnnw #1 #2 #3 #4 \q__cmd
1461 {
1462     \use:e
1463     {
1464         \__cmd_show_command_aux:NnNNn \__cmd_show:e { document~environment }
1465         { \exp_not:N \begin {#3} }
1466         \exp_not:c { environment~ #3 ~ code }
1467         {#2}
1468     }
1469 }
1470 \cs_if_exist:NTF \iow_show:e
1471 {
1472     \cs_new_protected:Npn \__cmd_show:e #1
1473     { \iow_show:e { > ~ #1 . \iow_newline: } }
1474 }
1475 {
1476     \cs_new_protected:Npn \__cmd_show:e #1
1477     { \iow_term:e { > ~ #1 . \iow_newline: } }
1478 }
1479 \cs_new_protected:Npn \__cmd_show_environment_end:N #1
1480 {
1481     \exp_args:NNe \__cmd_check_end:Nn \l__cmd_tmpa_tl { \cs_to_str:N #1 }

```

```

1482 \exp_args:Nc \__cmd_show_environment:N { \l__cmd_tmpa_tl }
1483 }

```

And, of course, add `\__kernel_cmd_if_xparse:NTF` and `\__cmd_show:N` to `\@showcommandlisthook` and to `\@showenvironmentlisthook` (`\__cmd_show:N` takes care of the environment case as well, so both entries are identical):

```

1484 \tl_gput_right:Nn \@showcommandlisthook
1485 { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
1486 \tl_gput_right:Nn \@showenvironmentlisthook
1487 { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }

```

(End of definition for `\__cmd_show_command:N` and others.)

`\__cmd_split_signature:n` Now we'll try a least-effort adventure into splitting the symbolic user-provided signature for a command into individual parameters for pretty-printing. A counter is used to keep track of the current argument number, and two token lists are used: `\l__cmd_tmpa_tl` holds the final token list to be printed, and `\l__cmd_tmpb_tl` holds just the current item, so that we can make changes to an individual item without having to dissect the whole thing (this is used for e- and E-types).

```

1488 \cs_new_protected:Npn \__cmd_split_signature:n #1
1489 {
1490   \int_set:Nn \l__cmd_current_arg_int { 1 }
1491   \tl_clear:N \l__cmd_tmpa_tl
1492   \tl_clear:N \l__cmd_tmpb_tl
1493   \__cmd_split_signature_loop:Nw #1 \q_recursion_tail \q_recursion_stop
1494 }

```

`\__cmd_split_signature_loop:Nw` This is the main chunk of the loop: it starts an item with `\__cmd_split_start_item:` (this adds indentation and the argument number to `\l__cmd_tmpb_tl`), then checks if a special token list `\c__cmd_show_type_⟨type⟩_tl` exists. If it doesn't, the current argument is a "simple" type which needs no extra processing. Otherwise, call a specific function depending on the value of said token list.

```

1495 \cs_new_protected:Npn \__cmd_split_signature_loop:Nw #1
1496 {
1497   \quark_if_recursion_tail_stop:N #1
1498   \tl_if_empty:NT \l__cmd_tmpb_tl { \__cmd_split_start_item: }
1499   \tl_if_exist:cTF { c__cmd_show_type_#1_tl }
1500   {
1501     \use:c
1502     {
1503       __cmd_show_
1504       \if_case:w \tl_use:c { c__cmd_show_type_#1_tl } \exp_stop_f:
1505       delim \or: delims \or: delims_opt \or: opt \or:
1506       e \or: E \or: prefix \or: processor \fi: :Nw
1507     } #1
1508   }
1509   { \__cmd_split_end_item:n {#1} \__cmd_split_signature_loop:Nw }
1510 }

```

The token lists `\c__cmd_show_type_⟨type⟩_tl` exist for nontrivial (for printing) *⟨types⟩* that require special parsing (like delimiters or optional arguments). Values from 0 to 7 are assigned to each type:

```

\c__cmd_show_type_t_tl
\c__cmd_show_type_r_tl
\c__cmd_show_type_d_tl
\c__cmd_show_type_R_tl
\c__cmd_show_type_D_tl
\c__cmd_show_type_0_tl
\c__cmd_show_type_e_tl
\c__cmd_show_type_E_tl
\c__cmd_show_type+_tl
\c__cmd_show_type!_tl
\c__cmd_show_type>_tl

```

1. a single delimiter token;

2. two delimiter tokens;
3. two delimiter tokens plus a default value;
4. a default value;
5. a list of embellishments (exclusive for e-type);
6. embellishments plus defaults (exclusive for E-type);
7. simple prefixes;
8. prefixes with arguments (argument processors);

```

1511 \cs_set_protected:Npn \__cmd_tmp:w #1 #2
1512 {
1513   \quark_if_nil:nF {#1}
1514   { \tl_const:cn { c__cmd_show_type_#1_tl } {#2} \__cmd_tmp:w }
1515 }
1516 \__cmd_tmp:w t0 r1 d1 R2 D2 O3 e4 E5 +6 !6 >7 =7 \q_nil \q_nil

```

Now, based on each type we know how to act. In most cases it is just a matter of feeding in the grabbed arguments and resuming the loop. The embellishments require a bit more attention: the e-type loops through the list of embellishments and adds each to the token list as a separate argument. The E-type does more or less the same, but uses `\__cmd_tl_mapthread_function:nnN` to map over two lists simultaneously, adding each token and default to the token list for printing.

```

1517 \cs_new_protected:Npn \__cmd_show_delim:Nw #1 #2
1518 { \__cmd_split_end_item:n { #1 #2 } \__cmd_split_signature_loop:Nw }
1519 \cs_new_protected:Npn \__cmd_show_delims:Nw #1 #2 #3
1520 { \__cmd_split_end_item:n { #1 #2 #3 } \__cmd_split_signature_loop:Nw }
1521 \cs_new_protected:Npn \__cmd_show_delims_opt:Nw #1 #2 #3 #4
1522 { \__cmd_split_end_item:n { #1 #2 #3 {#4} } \__cmd_split_signature_loop:Nw }
1523 \cs_new_protected:Npn \__cmd_show_opt:Nw #1 #2
1524 { \__cmd_split_end_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
1525 \cs_new_protected:Npn \__cmd_show_e:Nw #1 #2
1526 {
1527   \tl_map_inline:nn {#2}
1528   {
1529     \__cmd_split_start_item:
1530     \__cmd_split_end_item:n { #1 ##1 }
1531   }
1532   \__cmd_split_signature_loop:Nw
1533 }
1534 \cs_set_protected:Npn \__cmd_tmp:w #1
1535 {
1536   \cs_new_protected:Npn \__cmd_show_E:Nw ##1 ##2 ##3
1537   {
1538     \cs_set_protected:Npn \__cmd_tmp:w ####1 ####2
1539     {
1540       \__cmd_split_start_item:
1541       \__cmd_split_end_item:n { ##1 ####1 {####2} }
1542     }
1543     \__cmd_tl_mapthread_function:nnN {##2}
1544     { ##3 {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} } \__cmd_tmp:w

```

```

1545         \_cmd_split_signature_loop:Nw
1546     }
1547 }
1548 \_cmd_tmp:w \NoValue

```

Minor wrinkle with the prefixes: they use `\_cmd_split_add_item:n` instead of `\_cmd_split_end_item:n` (add *vs.* end) because they are followed by an argument, so they can't end the item.

```

1549 \cs_new_protected:Npn \_cmd_show_prefix:Nw #1
1550 { \_cmd_split_add_item:n {#1} \_cmd_split_signature_loop:Nw }
1551 \cs_new_protected:Npn \_cmd_show_processor:Nw #1 #2
1552 { \_cmd_split_add_item:n { #1 {#2} } \_cmd_split_signature_loop:Nw }

```

And now the auxiliaries that store the strings to be printed. `\_cmd_split_start_item:` starts an item from scratch, `\_cmd_split_add_item:n` adds tokens to an item without adding a newline, and `\_cmd_split_end_item:n` adds tokens, terminates the item with a newline, and steps the argument count.

```

1553 \cs_new_protected:Npn \_cmd_split_start_item:
1554 {
1555     \tl_set:Nx \l__cmd_tmpl_tl
1556     { ~ \c_space_tl \c_hash_str \int_use:N \l__cmd_current_arg_int : }
1557 }
1558 \cs_new_protected:Npn \_cmd_split_add_item:n #1
1559 { \tl_put_right:Nx \l__cmd_tmpl_tl { \tl_to_str:n {#1} } }
1560 \cs_new_protected:Npn \_cmd_split_end_item:n #1
1561 {
1562     \tl_put_right:Nx \l__cmd_tmpl_tl
1563     { \l__cmd_tmpl_tl \tl_to_str:n {#1} \iow_newline: }
1564     \tl_clear:N \l__cmd_tmpl_tl
1565     \int_incr:N \l__cmd_current_arg_int
1566 }

```

(End of definition for `\_cmd_split_signature:n` and others.)

Not much to do regarding `latexrelease`: we could remove the entries from `\@showcommandlisthook`, but it doesn't seem worth it.

```

\_cmd_split_start_item: 1567 <latexrelease>\EndIncludeInRelease
\_cmd_split_add_item:n 1568 %
\_cmd_split_end_item:n 1569 <latexrelease>\IncludeInRelease{2020/10/01}{\_cmd_show:N (part 2)}%
1570 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
1571 <latexrelease>\EndIncludeInRelease

```

1.27 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in `\l__cmd_signature_tl` the code to grab further arguments, defines (the function in) `\l__cmd_fn_tl` that will grab the argument, and calls it.

Defining `\l__cmd_fn_tl` means determining whether to use `\cs_set:Npn` or `\cs_set_nopar:Npn`, and for optional arguments whether to skip spaces. Once the argument is found, `\l__cmd_fn_tl` calls `\_cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

`\__cmd_grab_b:w` This uses the well-tested code of D-type arguments, skipping the peeking step because the
`\__cmd_grab_b_long:w` b-type argument is always present, and adding a cleanup stage at the end by hijacking
`\__cmd_grab_b_obey_spaces:w` the signature. The clean-up consists of properly dealing with `\l__cmd_args_tl` and also
`\__cmd_grab_b_long_obey_spaces:w` putting back the `\end` that served as an end-delimiter: this `\end` receives the environment
`\__cmd_grab_b_aux:NNw` name as its argument and is run normally. The D-type code stores the argument found
`\__cmd_grab_b_end:Nw` (body of the environment) as a brace group in `\l__cmd_args_tl` and depending on the
presence of a prefix `!` we trim spaces or not before adding this braced argument into
the saved `\l__cmd_args_tl`. The strange `\begin_` control sequence is there for display
purposes only: it has to look like `\begin` in the terminal but not to delimited arguments.

```

1572 \cs_new_protected:Npn \__cmd_grab_b:w
1573 { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
1574 \cs_new_protected:Npn \__cmd_grab_b_long:w
1575 { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
1576 \cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
1577 { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
1578 \cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
1579 { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
1580 \cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
1581 {
1582   \__cmd_grab_D_aux:NNnNN \begin \end {#3} #1 \use_i:nn
1583   \tl_put_left:Nn \l__cmd_signature_tl { \__cmd_grab_b_end:Nw #2 }
1584   \tl_set_eq:NN \l__cmd_saved_args_tl \l__cmd_args_tl
1585   \tl_clear:N \l__cmd_args_tl
1586   \exp_args:Nc \l__cmd_fn_tl { begin ~ }
1587 }
1588 \cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
1589 {
1590   \tl_set:Ne \l__cmd_args_tl
1591   {
1592     \exp_not:V \l__cmd_saved_args_tl
1593     { \exp_after:wN #1 \l__cmd_args_tl }
1594   }
1595   #2
1596   \__cmd_run_code:
1597   \end
1598 }

```

(End of definition for `\__cmd_grab_b:w` and others.)

`\__cmd_grab_c:w` Collecting an environment body verbatim shares some ideas with the v-type grabber, and
`\__cmd_grab_c_obey_spaces:w` others with the standard `filecontents` environment. The start is to set the end-of-line
`\__cmd_grab_c_start:n` to a predictable value and to deactivate the specials.

```

1599 \cs_new_protected:Npn \__cmd_grab_c:w #1 \__cmd_run_code:
1600 {
1601   \bool_set_false:N \l__cmd_obey_spaces_bool
1602   \__cmd_grab_c_start:n {#1}
1603 }
1604 \cs_new_protected:Npn \__cmd_grab_c_obey_spaces:w #1 \__cmd_run_code:
1605 {
1606   \bool_set_true:N \l__cmd_obey_spaces_bool
1607   \__cmd_grab_c_start:n {#1}
1608 }
1609 \cs_new_protected:Npn \__cmd_grab_c_start:n #1

```

```

1610 {
1611   \tl_set:Nn \l__cmd_signature_tl {#1}
1612   \group_begin:
1613     \tl_clear:N \l__cmd_v_arg_tl
1614     \tex_escapechar:D = 92 \scan_stop:
1615     \tex_endlinechar:D = '\^M \scan_stop:
1616     \cs_set_eq:NN \do \char_set_catcode_other:N
1617     \dospecials
1618     \char_set_catcode_other:n { '\^M }
1619     \__cmd_grab_c_first:w
1620 }

```

Notice here and below that we cannot use `\token_to_str:N \end` as that would have the wrong category codes for the letters.

```

1621 \group_begin:
1622   \char_set_catcode_other:N \^M %
1623   \cs_new_protected:Npn \__cmd_grab_c_first:w #1 ^M %
1624   { %
1625     \tl_if_blank:nTF {#1} %
1626     { %
1627       \__cmd_grab_c_loop:w #1 ^M %
1628     } %
1629     { %
1630       \msg_warning:nnee { cmd } { chars-dropped-first-line } %
1631       { \exp_not:n {#1} } %
1632       { \exp_not:V \@currenvir } %
1633       \__cmd_grab_c_loop:w ^M %
1634     } %
1635   } %
1636   \cs_new_protected:Npe \__cmd_grab_c_loop:w #1 ^M %
1637   { %
1638     \exp_not:N \__cmd_grab_c_auxi:w #1 %
1639     \c_backslash_str end %
1640     \scan_stop: %
1641   } %
1642 \group_end:

```

(End of definition for `\__cmd_grab_c:w` and others.)

```

\__cmd_grab_c_auxi:w We need to see if the current line contains \end followed by the name of the current
\__cmd_grab_c_auxii:w environment. To do that and allow for spaces, we have to work stepwise. First, establish
\__cmd_grab_c_auxiii:N if there is an \end at all: remember that here we are dealing with “other” tokens. Whether
\__cmd_grab_c_auxiv: these is an \end or not, the tokens before it form part of the line.
\__cmd_grab_c_auxv:
1643 \use:e
\__cmd_grab_c_auxvi:N {
1644   {
1645     \cs_new_protected:Npn \exp_not:N \__cmd_grab_c_auxi:w
1646     #1 \c_backslash_str end #2 \scan_stop:
1647   }
1648   {
1649     \tl_put_right:Nn \l__cmd_v_arg_tl {#1}
1650     \tl_if_empty:nTF {#2}
1651     {
1652       \tl_put_right:Nn \l__cmd_v_arg_tl { \obeyedline }
1653       \__cmd_grab_c_loop:w

```

```

1654     }
1655     { \_cmd_grab_c_auxii:w #2 \scan_stop: }
1656   }

```

There is an `\end`, so we now remove the trailing marker we needed to do the test. This is stripped off, then we need to examine the rest of the line one token at a time: see `verbatim.dtx` for the inspiration. Notice that we use `^^M` here as the end marker: this allows looping to look for multiple `\end` entries in the line.

```

1657 \group_begin:
1658   \char_set_catcode_other:N \^^M %
1659   \use:e %
1660   { %
1661     \cs_new_protected:Npe \exp_not:N \_cmd_grab_c_auxii:w %
1662     #1 \c_backslash_str end \scan_stop: %
1663   } %
1664   { %
1665     \tl_set:Nn \exp_not:N \l__cmd_tmpa_tl
1666     { \c_backslash_str end } %
1667     \exp_not:N \_cmd_grab_c_auxiii:N #1 ^^M %
1668   } %

```

Within the line, we need to collect up the tokens: if we do not find the end-of-environment argument, we will need those to reinsert. There are three special cases here: `^^M` (end of line: tidy up and back to the main loop), `\` (possibly skip over) and `{` (start the inner loop). Anything else means we move back to examine the rest of the line for any more `\end` entries.

```

1669   \cs_new_protected:Npn \_cmd_grab_c_auxiii:N #1 %
1670   { %
1671     \token_case_charcode:NnF #1 %
1672     { %
1673       ^^M %
1674       { \_cmd_grab_c_auxiv: } %
1675       \c_space_token %
1676       { %
1677         \tl_put_right:Nn \l__cmd_tmpa_tl {#1} %
1678         \_cmd_grab_c_auxiii:N %
1679       } %
1680       \c_group_begin_token %
1681       { %
1682         \tl_set:Nn \l__cmd_tmpb_tl {#1} %
1683         \_cmd_grab_c_auxvi:N %
1684       } %
1685     } %
1686     { %
1687       \tl_put_right:Nn \l__cmd_tmpa_tl {#1} %
1688       \_cmd_grab_c_auxv: %
1689     } %
1690   } %
1691 \group_end:
1692 \cs_new_protected:Npn \_cmd_grab_c_auxiv:
1693 {
1694   \tl_put_right:Ne \l__cmd_v_arg_tl
1695   {
1696     \exp_not:V \l__cmd_tmpa_tl

```

```

1697         \exp_not:N \obeyedline
1698     }
1699     \__cmd_grab_c_loop:w
1700 }
1701 \cs_new_protected:Npn \__cmd_grab_c_auxv:
1702 {
1703     \tl_put_right:NV \l__cmd_v_arg_tl \l__cmd_tmpa_tl
1704     \__cmd_grab_c_loop:w
1705 }

```

In the inner loop, we again have only a few special cases. First, we could again have `^^M`, in which case we tidy up using a common auxiliary. Second, we check for the escape char: this cannot happen inside the end of an environment and means we loop, re-inserting the token. Finally, we have `}`, where we need to move on to check what has been collected. Otherwise, collect up and loop. Notice here that the inner loop needs to collect tokens separately: this leaves any spaces after `\end` in `\l__cmd_tmpa_tl`, so we can test `\l__cmd_tmpb_tl` directly.

```

1706 \group_begin:
1707     \char_set_catcode_other:N ^^M %
1708     \cs_new_protected:Npe \__cmd_grab_c_auxvii:N #1 %
1709     { %
1710         \exp_not:N \token_case_charcode:NnF #1 %
1711         { %
1712             ^^M %
1713             {
1714                 \exp_not:N \__cmd_grab_c_auxvii: %
1715                 \exp_not:N \__cmd_grab_c_auxiv: %
1716             }%
1717             \c_backslash_str %
1718             { %
1719                 \exp_not:N \__cmd_grab_c_auxvii: %
1720                 \exp_not:N \__cmd_grab_c_auxv: #1
1721             } %
1722             \c_group_end_token %
1723             { %
1724                 \tl_put_right:Nn \exp_not:N \l__cmd_tmpb_tl {#1} %
1725                 \exp_not:N \__cmd_grab_c_auxviii: %
1726             } %
1727         } %
1728         { %
1729             \tl_put_right:Nn \exp_not:N \l__cmd_tmpb_tl {#1} %
1730             \exp_not:N \__cmd_grab_c_auxvi:N %
1731         } %
1732     } %
1733 \group_end: %
1734 \cs_new_protected:Npn \__cmd_grab_c_auxvii:
1735 { \tl_put_right:NV \l__cmd_tmpa_tl \l__cmd_tmpb_tl }
1736 \cs_new_protected:Npn \__cmd_grab_c_auxviii:
1737 {
1738     \str_if_eq:eeTF { \exp_not:N \l__cmd_tmpb_tl } { { \@currenvir } }
1739     { \__cmd_grab_c_end:w }
1740     {
1741         \__cmd_grab_c_auxvii:
1742         \__cmd_grab_c_auxv:

```

```

1743     }
1744 }

```

(End of definition for `\__cmd_grab_c_auxi:w` and others.)

`\__cmd_grab_c_end:w` To end the collection, we clean up the last line: once again we need to find $\sim$ M. Once that is done, we can warn if there is anything left behind.

```

\__cmd_grab_c_end:n
\__cmd_grab_c_end_auxi:w 1745 \group_begin:
\__cmd_grab_c_end_auxii:w 1746 \char_set_catcode_other:N \sim M %
\__cmd_grab_c_end_auxiii:w 1747 \cs_new_protected:Npn \__cmd_grab_c_end:w #1 \sim M %
1748 { %
1749     \tl_if_blank:nF {#1} %
1750     { %
1751         \msg_warning:nnee { cmd } { chars-dropped-last-line } %
1752         { \exp_not:n {#1} } %
1753         { \exp_not:V \@currenvir } %
1754     } %
1755     \exp_args:NNNo \group_end: %
1756     \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl } %
1757     \__cmd_add_arg:e %
1758     { %
1759         \bool_if:NTF \l__cmd_obey_spaces_bool %
1760         { \exp_not:V } %
1761         { \exp_args:NV \__cmd_grab_c_end:n } %
1762         \l__cmd_v_arg_tl %
1763     } %
1764     \exp_args:NV \end \@currenvir %
1765 } %
1766 \group_end: %

```

Look for line markers at each end and tidy up if required.

```

1767 \cs_new:Npn \__cmd_grab_c_end:n #1
1768 {
1769     \__cmd_grab_c_end_auxi:w \q_nil #1 \q_nil
1770     \obeyedline \obeyedline \q_nil \q_stop
1771 }
1772 \cs_new:Npn \__cmd_grab_c_end_auxi:w #1 \q_nil \obeyedline
1773 { \__cmd_grab_c_end_auxii:w #1 \q_nil }
1774 \cs_new:Npn \__cmd_grab_c_end_auxii:w \q_nil #1 \obeyedline \q_nil
1775 { \__cmd_grab_c_end_auxiii:w #1 \q_nil }
1776 \cs_new:Npn \__cmd_grab_c_end_auxiii:w #1 \q_nil #2 \q_stop
1777 { \exp_not:n {#1} }

```

(End of definition for `\__cmd_grab_c_end:w` and others.)

`\__cmd_grab_D:w` The generic delimited argument grabber. The auxiliary function does a peek test before calling `\__cmd_grab_D_call:Nw`, so that the optional nature of the argument works as expected.

```

\__cmd_grab_D_no_strip:w
\__cmd_grab_D_obey_spaces:w 1778 \tl_map_inline:nn { { } { _long } }
\__cmd_grab_D_obey_spaces_no_strip:w 1779 {
\__cmd_grab_D_long:w 1780     \tl_map_inline:nn { { } { _obey_spaces } }
\__cmd_grab_D_long_no_strip:w 1781 {
\__cmd_grab_D_long_obey_spaces:w 1782     \tl_map_inline:nn { { } { _no_strip } }
\__cmd_grab_D_long_obey_spaces_no_strip:w 1783     {
1784         \tl_map_inline:nn { { } { _verb_safe } }

```

```

1785         {
1786         \cs_new_protected:cpe { __cmd_grab_D #1 ##1 #####1 :w }
1787         #####1#####2#####3
1788         \__cmd_run_code:
1789         {
1790         \exp_not:N \__cmd_grab_D_aux:NNnNNNN
1791         #####1 #####2 {#####3}
1792         \str_if_eq:nnTF {#1} { _long }
1793         \cs_set_protected:Npn
1794         \cs_set_protected_nopar:Npn
1795         \str_if_eq:nnTF {##1} { _obey_spaces }
1796         { \exp_not:N \__cmd_peek_meaning_remove:NTF }
1797         { \exp_not:N \__cmd_peek_nonspace_remove:NTF }
1798         \str_if_eq:nnTF {####1} { _no_strip }
1799         { \exp_not:N \use_none:n }
1800         { \exp_not:N \use_ii:nn }
1801         \str_if_eq:nnTF {#####1} { _verb_safe }
1802         { \exp_not:N \use:n }
1803         { \exp_not:N \use_none:n }
1804         }
1805     }
1806 }
1807 }
1808 }

```

This is a bit complicated. The idea is that, in order to check for nested optional argument tokens ([...] and so on) the argument needs to be grabbed without removing any braces at all. If this is not done, then cases like [[]] fail. So after testing for an optional argument, it is collected piece-wise. Inserting a quark prevents loss of braces, and there is then a test to see if there are nested delimiters to handle.

```

1809 \group_begin:
1810 \char_set_catcode_other:N \^M
1811 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNNNN #1#2#3#4#5#6#7
1812 {
1813     \__cmd_grab_D_aux:NNnNN #1#2 {#3} #4 #6
1814     #7
1815     {
1816         \group_begin:
1817         \__cmd_grab_D_verb_safe:NN #1#5
1818     }
1819     #5 #1
1820     {
1821         #7 { \group_end: }
1822         \__cmd_grab_D_call:Nw #1
1823     }
1824     {
1825         #7 { \group_end: }
1826         \__cmd_add_arg:n { \NoValue }
1827     }
1828 }
1829 \group_end:

```

The only awkwardness here is the need to preserve the catcode of the search token: this is done low-level for performance reasons. Only values that are realistic are included. As

this does not cover spaces when they are skipped that has to be covered separately.

```

1830 \cs_new_protected:Npn \__cmd_grab_D_verb_safe:NN #1#2
1831 {
1832   \cs_set_eq:NN \do \char_set_catcode_other:N
1833   \dospecials
1834   \char_set_catcode_other:N \^~M
1835   \token_if_eq_meaning:NNT #2 \__cmd_peek_nonspace_remove:NTF
1836   { \char_set_catcode_space:n { '\ } }
1837   \use:c
1838   {
1839     char_set_catcode_
1840     \if_catcode:w \c_math_toggle_token #1 math_toggle \else:
1841     \if_catcode:w ^ #1 math_superscript \else:
1842     \if_catcode:w \c_math_subscript_token #1 math_subscript \else:
1843     \if_catcode:w A #1 letter \else:
1844     other \fi: \fi: \fi: \fi:
1845     :N
1846   }
1847   #1
1848 }

```

Inside the “standard” grabber, there is a test to see if the grabbed argument is entirely enclosed by braces. There are a couple of extra factors to allow for: the argument might be entirely empty, and spaces at the start and end of the input must be retained around a brace group. Also notice that a *blank* argument might still contain spaces. To allow for suppression of brace stripping, the business end is passed here as #5.

```

1849 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNN #1#2#3#4#5
1850 {
1851   \tl_set:Nn \l__cmd_signature_tl {#3}
1852   \exp_after:wN #4 \l__cmd_fn_tl ##1 #2
1853   {
1854     \tl_if_in:nnTF {##1} {#1}
1855     { \__cmd_grab_D_nested:NNnN #1 #2 {##1} #4 }
1856     {
1857       \tl_if_blank:oTF { \use_none:n ##1 }
1858       { \__cmd_add_arg:o { \use_none:n ##1 } }
1859       {
1860         \str_if_eq:eeTF
1861         { \exp_not:o { \use_none:n ##1 } }
1862         { { \exp_not:o { \use_ii:nnn ##1 \q_nil } } }
1863         { \__cmd_add_arg:o { #5 ##1 } }
1864         { \__cmd_add_arg:o { \use_none:n ##1 } }
1865       }
1866     }
1867   }
1868 }

```

(End of definition for __cmd_grab_D:w and others.)

```

\__cmd_grab_D_nested:NNnN
\__cmd_grab_D_nested:w
\l__cmd_nesting_a_tl
\l__cmd_nesting_b_tl
\q__cmd

```

Catching nested optional arguments means more work. The aim here is to collect up each pair of optional tokens without T_EX helping out, and without counting anything. The code above will already have removed the leading opening token and a closing token, but the wrong one. The aim is then to work through the material grabbed so far and divide it up on each opening token, grabbing a closing token to match (thus working in

pairs). Once there are no opening tokens, then there is a second check to see if there are any opening tokens in the second part of the argument (for things like `[] []`). Once everything has been found, the entire collected material is added to the output as a single argument. The only tricky part here is ensuring that any grabbing function that might run away is named after the function currently being parsed and not after `xparse`. That leads to some rather complex nesting! There is also a need to prevent the loss of any braces, hence the insertion and removal of quarks along the way.

```

1869 \tl_new:N \l__cmd_nesting_a_tl
1870 \tl_new:N \l__cmd_nesting_b_tl
1871 \quark_new:N \q__cmd
1872 \cs_new_protected:Npn \__cmd_grab_D_nested:NNnN #1#2#3#4
1873 {
1874   \tl_clear:N \l__cmd_nesting_a_tl
1875   \tl_clear:N \l__cmd_nesting_b_tl
1876   \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
1877   {
1878     \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
1879     \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn #2 ##3 }
1880     \tl_if_in:nnTF {##2} {#1}
1881     {
1882       \l__cmd_fn_tl
1883       \q_nil ##2 \q__cmd \ERROR
1884     }
1885     {
1886       \tl_put_right:Ne \l__cmd_nesting_a_tl
1887       { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
1888       \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
1889       {
1890         \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
1891         \tl_clear:N \l__cmd_nesting_b_tl
1892         \exp_after:wN \l__cmd_fn_tl \exp_after:wN
1893         \q_nil \l__cmd_tmpa_tl \q_nil \q__cmd \ERROR
1894       }
1895       {
1896         \tl_put_right:No \l__cmd_nesting_a_tl
1897         \l__cmd_nesting_b_tl
1898         \__cmd_add_arg:V \l__cmd_nesting_a_tl
1899       }
1900     }
1901   }
1902   \l__cmd_fn_tl #3 \q_nil \q__cmd \ERROR
1903 }
1904 \cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
1905 { \exp_not:o { \use_none:n #1 } }

```

(End of definition for `\__cmd_grab_D_nested:NNnN` and others.)

`\__cmd_grab_D_call:Nw` For D and R-type arguments, to avoid losing any braces, a token needs to be inserted before the argument to be grabbed. If the argument runs away because the closing token is missing then this inserted token shows up in the terminal. Ideally, `#1` would therefore be used directly, but that is no good as it will mess up the rest of the grabber. Instead, a copy of `#1` with an altered category code is used, as this will look right in the terminal but will not mess up the grabber. The only issue then is that the category code of `#1`

is unknown. So there is a quick test to ensure that the inserted token can never be matched by the grabber. (This assumes that the open and close delimiters are not the same character with different category codes, but that really should not happen in any sensible document-level syntax.) An exception is when #1 is a control sequence token, in which case the character-token treatment is no good because if hit with `\token_to_str:N` it would add sputios tokens to the argument. In this case a different branch is taken. The token inserted is then the same $\langle csname \rangle$ as #1, but with a space appended, so that the grabber don't see it as another of the same delimiter.

```

1906 \cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
1907 {
1908   \token_if_eq_catcode:NNTF + #1
1909   {
1910     \exp_after:wN \exp_after:wN \exp_after:wN
1911     \l__cmd_fn_tl \char_generate:nn { '#1 } { 11 }
1912   }
1913   {
1914     \__cmd_token_if_cs:NNTF #1
1915     {
1916       \exp_after:wN \l__cmd_fn_tl
1917       \cs:w \cs_to_str:N #1 ~ \cs_end:
1918     }
1919     {
1920       \exp_after:wN \l__cmd_fn_tl
1921       \token_to_str:N #1
1922     }
1923   }
1924 }

```

(End of definition for `\__cmd_grab_D_call:Nw`.)

`\__cmd_grab_E:w` Everything here needs to point to a loop.

```

\__cmd_grab_E_long:w
\__cmd_grab_E_obey_spaces:w
\__cmd_grab_E_long_obey_spaces:w
\__cmd_grab_E:nnNN
\__cmd_grab_E_loop:NnN
\__cmd_grab_E_finalize:
1925 \cs_new_protected:Npn \__cmd_grab_E:w #1#2 \__cmd_run_code:
1926 {
1927   \__cmd_grab_E:nnNN {#1} {#2}
1928   \cs_set_protected_nopar:Npn
1929   \__cmd_peek_nonspace_remove:NNTF
1930 }
1931 \cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
1932 {
1933   \__cmd_grab_E:nnNN {#1} {#2}
1934   \cs_set_protected:Npn
1935   \__cmd_peek_nonspace_remove:NNTF
1936 }
1937 \cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
1938 {
1939   \__cmd_grab_E:nnNN {#1} {#2}
1940   \cs_set_protected_nopar:Npn
1941   \__cmd_peek_meaning_remove:NNTF
1942 }
1943 \cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
1944 {
1945   \__cmd_grab_E:nnNN {#1} {#2}
1946   \cs_set_protected:Npn

```

```

1947     \__cmd_peek_meaning_remove:NTF
1948   }

```

A loop is needed here to allow a random ordering of keys. These are searched for one at a time, with any not found needing to be tracked: they can appear later. The grabbed values are held in a property list which is then turned into an ordered list to be passed back to the user.

```

1949 \cs_new_protected:Npn \__cmd_grab_E:nnNN #1#2#3#4
1950 {
1951   \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
1952   {
1953     \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
1954     \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
1955   }
1956   \prop_clear:N \l__cmd_tmp_prop
1957   \tl_set:Nn \l__cmd_signature_tl {#2}
1958   \cs_set_protected:Npn \__cmd_grab_E_finalize:
1959   {
1960     \tl_map_inline:nn {#1}
1961     {
1962       \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmpb_tl
1963       { \tl_set:Nn \l__cmd_tmpb_tl { \NoValue } }
1964       \tl_put_right:Ne \l__cmd_args_tl
1965       { { \exp_not:V \l__cmd_tmpb_tl } }
1966     }
1967     \l__cmd_signature_tl \__cmd_run_code:
1968   }
1969   \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
1970 }
1971 \cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
1972 {
1973   \cs_if_eq:NNTF #3 \q_recursion_tail
1974   { \__cmd_grab_E_finalize: }
1975   {
1976     #1 #3
1977     { \l__cmd_fn_tl #3 {#2#4} }
1978     { \__cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
1979   }
1980 }
1981 \cs_new_protected:Npn \__cmd_grab_E_finalize: { }

```

(End of definition for __cmd_grab_E:w and others.)

```

\__cmd_grab_m:w Collecting a single mandatory argument is quite easy.
\__cmd_grab_m_long:w
1982 \cs_new_protected:Npn \__cmd_grab_m:w #1 \__cmd_run_code:
1983 {
1984   \tl_set:Nn \l__cmd_signature_tl {#1}
1985   \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
1986   { \__cmd_add_arg:n {##1} }
1987   \l__cmd_fn_tl
1988 }
1989 \cs_new_protected:Npn \__cmd_grab_m_long:w #1 \__cmd_run_code:
1990 {
1991   \tl_set:Nn \l__cmd_signature_tl {#1}

```

```

1992     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
1993     { \__cmd_add_arg:n {##1} }
1994     \l__cmd_fn_tl
1995 }

```

(End of definition for __cmd_grab_m:w and __cmd_grab_m_long:w.)

__cmd_grab_m_1:w Grabbing 1–8 mandatory arguments is done by giving 8–1 known arguments to a 9-argument function that stores them in \l__cmd_args_tl. For simplicity, grabbing 9 mandatory arguments is done by grabbing 5 then 4 arguments.

```

1996 \cs_new_protected_nopar:Npn \__cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9
1997 {
1998     \tl_put_right:No \l__cmd_args_tl
1999     { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }
2000     \l__cmd_signature_tl \__cmd_run_code:
2001 }

```

```

\__cmd_grab_m_aux:Nnnnnnnnn
2002 \cs_new_protected:cpn { __cmd_grab_m_1:w } #1 \__cmd_run_code:
2003 {
2004     \tl_set:Nn \l__cmd_signature_tl {#1}
2005     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2006     \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { } { }
2007 }

```

```

2008 \cs_new_protected:cpn { __cmd_grab_m_2:w } #1 \__cmd_run_code:
2009 {
2010     \tl_set:Nn \l__cmd_signature_tl {#1}
2011     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2012     \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { } { }
2013 }

```

```

2014 \cs_new_protected:cpn { __cmd_grab_m_3:w } #1 \__cmd_run_code:
2015 {
2016     \tl_set:Nn \l__cmd_signature_tl {#1}
2017     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2018     \l__cmd_fn_tl \use_none:nnnnn { } { } { } { } { } { } { }
2019 }

```

```

2020 \cs_new_protected:cpn { __cmd_grab_m_4:w } #1 \__cmd_run_code:
2021 {
2022     \tl_set:Nn \l__cmd_signature_tl {#1}
2023     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2024     \l__cmd_fn_tl \use_none:nnnn { } { } { } { } { } { } { }
2025 }

```

```

2026 \cs_new_protected:cpn { __cmd_grab_m_5:w } #1 \__cmd_run_code:
2027 {
2028     \tl_set:Nn \l__cmd_signature_tl {#1}
2029     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2030     \l__cmd_fn_tl \use_none:nnn { } { } { } { } { } { } { }
2031 }

```

```

2032 \cs_new_protected:cpn { __cmd_grab_m_6:w } #1 \__cmd_run_code:
2033 {
2034     \tl_set:Nn \l__cmd_signature_tl {#1}
2035     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnnn
2036     \l__cmd_fn_tl \use_none:nn { } { } { } { } { } { } { }
2037 }

```

```

2038 \cs_new_protected:cpn { __cmd_grab_m_7:w } #1 \__cmd_run_code:
2039 {

```

```

2040     \tl_set:Nn \l__cmd_signature_tl {#1}
2041     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnn
2042     \l__cmd_fn_tl \use_none:n { }
2043   }
2044   \cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
2045   {
2046     \tl_set:Nn \l__cmd_signature_tl {#1}
2047     \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnn
2048     \l__cmd_fn_tl \prg_do_nothing:
2049   }
2050   \cs_new_protected:cpe { __cmd_grab_m_9:w }
2051   {
2052     \exp_not:c { __cmd_grab_m_5:w }
2053     \exp_not:c { __cmd_grab_m_4:w }
2054   }

```

(End of definition for __cmd_grab_m_1:w and others.)

__cmd_grab_R:w The grabber for R-type arguments is basically the same as that for D-type ones, but
 __cmd_grab_R_long:w always skips spaces (as it is mandatory) and has a hard-coded error message.

```

\__cmd_grab_R_aux:NNnN
2055 \cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
2056   { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn }
2057 \cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
2058   { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected:Npn }
2059 \cs_new_protected:Npn \__cmd_grab_R_aux:NNnN #1#2#3#4
2060   {
2061     \__cmd_grab_D_aux:NNnNN #1 #2 {#3} #4 \use_ii:nn
2062     \__cmd_peek_nonspace_remove:NTF #1
2063     { \__cmd_grab_D_call:Nw #1 }
2064     {
2065       \msg_error:nnee { cmd } { missing-required }
2066       { \__cmd_environment_or_command: }
2067       { \token_to_str:N #1 }
2068       \__cmd_add_arg:n { \NoValue }
2069     }
2070   }

```

(End of definition for __cmd_grab_R:w, __cmd_grab_R_long:w, and __cmd_grab_R_aux:NNnN.)

__cmd_grab_t:w Dealing with a token is quite easy. Check the match, remove the token if needed and add
 __cmd_grab_t_obey_spaces:w a flag to the output.

```

\__cmd_grab_t_aux:NNw
2071 \cs_new_protected:Npn \__cmd_grab_t:w
2072   { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
2073 \cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
2074   { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
2075 \cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
2076   {
2077     \tl_set:Nn \l__cmd_signature_tl {#3}
2078     \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
2079     {
2080       #1 #2
2081       { \__cmd_add_arg:n { \BooleanTrue } }
2082       { \__cmd_add_arg:n { \BooleanFalse } }
2083     }

```

```

2084 \l__cmd_fn_tl
2085 }

```

(End of definition for `\__cmd_grab_t:w`, `\__cmd_grab_t_obey_spaces:w`, and `\__cmd_grab_t_aux:NNw`.)

```

\l__cmd_v_arg_tl 2086 \tl_new:N \l__cmd_v_arg_tl

```

```

\__cmd_grab_v:w
\__cmd_grab_v_long:w
\__cmd_grab_v_aux:w
\__cmd_grab_v_group_end:

```

Firstly, it is necessary to change `\tex_endlinechar:D` so that newlines in different catcode regimes (e.g., `\ExplSyntaxOn`) are not misinterpreted as spaces. The opening delimiter is the first non-space token, and is never read verbatim. This is required by consistency with the case where the preceding argument was optional and absent: then `\TeX` has already read and tokenized that token when looking for the optional argument. The first thing is thus to check is that this delimiter is a character, and to distinguish the case of a left brace (in that case, `\group_align_safe_end:` is needed to compensate for the begin-group character that was just seen). Then set verbatim catcodes with `\__cmd_grab_v_aux_catcodes:`.

The group keep catcode changes local, and `\group_align_safe_begin/end:` allow to use a character with category code 4 (normally `&`) as the delimiter (all commands do `\group_align_safe_begin/end:`, so there's no need to do that again here). It is ended by `\__cmd_grab_v_group_end:`, which smuggles the collected argument out of the group.

```

2087 \cs_new_protected:Npn \__cmd_grab_v:w
2088 {
2089   \bool_set_false:N \l__cmd_long_bool
2090   \__cmd_grab_v_aux:w
2091 }
2092 \cs_new_protected:Npn \__cmd_grab_v_long:w
2093 {
2094   \bool_set_true:N \l__cmd_long_bool
2095   \__cmd_grab_v_aux:w
2096 }
2097 \cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
2098 {
2099   \tl_set:Nn \l__cmd_signature_tl {#1}
2100   \group_begin:
2101     \tex_escapechar:D = 92 \scan_stop:
2102     \tex_endlinechar:D = '\^M \scan_stop:
2103     \tl_clear:N \l__cmd_v_arg_tl
2104     \peek_remove_spaces:n
2105     {
2106       \peek_meaning_remove:NTF \c_group_begin_token
2107       {
2108         \group_align_safe_end:
2109         \__cmd_grab_v_bgroup:
2110       }
2111       {
2112         \peek_N_type:TF
2113         { \__cmd_grab_v_aux_test:N }
2114         { \__cmd_grab_v_aux_abort:n { } }
2115       }
2116     }

```

```

2117 }
2118 \cs_new_protected:Npn \__cmd_grab_v_group_end:
2119 {
2120     \exp_args:NNNo
2121     \group_end:
2122     \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl }
2123 }

```

(End of definition for __cmd_grab_v:w and others.)

__cmd_grab_v_aux_test:N Check that the opening delimiter is a character, setup category codes, then start reading tokens one by one, keeping the delimiter as an argument. If the verbatim was not nested, we will be grabbing one character at each step. Unfortunately, it can happen that what follows the verbatim argument is already tokenized. Thus, we check at each step that the next token is indeed a “nice” character, *i.e.*, is not a character with category code 1 (begin-group), 2 (end-group) or 6 (macro parameter), nor the space character, with category code 10 and character code 32, nor a control sequence. The partially built argument is stored in \l__cmd_v_arg_tl. If we ever meet a token which we cannot grab (non-N-type), or which is not a character according to __cmd_grab_v_token_if_char:NTF, then we bail out with __cmd_grab_v_aux_abort:n. Otherwise, we stop at the first character matching the delimiter.

```

2124 \cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
2125 {
2126     \__cmd_grab_v_token_if_char:NTF #1
2127     {
2128         \__cmd_grab_v_aux_put:N #1
2129         \__cmd_grab_v_aux_catcodes:
2130         \__cmd_grab_v_aux_loop:N #1
2131     }
2132     { \__cmd_grab_v_aux_abort:n {#1} #1 }
2133 }
2134 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1
2135 {
2136     \peek_N_type:TF
2137     { \__cmd_grab_v_aux_loop:NN #1 }
2138     { \__cmd_grab_v_aux_abort:n { } }
2139 }
2140 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
2141 {
2142     \__cmd_grab_v_token_if_char:NTF #2
2143     {
2144         \token_if_eq_charcode:NNTF #1 #2
2145         { \__cmd_grab_v_aux_loop_end: }
2146         {
2147             \__cmd_grab_v_aux_put:N #2
2148             \__cmd_grab_v_aux_loop:N #1
2149         }
2150     }
2151     { \__cmd_grab_v_aux_abort:n {#2} #2 }
2152 }
2153 \cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
2154 {
2155     \__cmd_grab_v_group_end:

```

```

2156 \__cmd_add_arg:e { \tl_tail:N \l__cmd_v_arg_tl }
2157 }

```

(End of definition for __cmd_grab_v_aux_test:N and others.)

\l__cmd_v_nesting_int

```

2158 \int_new:N \l__cmd_v_nesting_int

```

__cmd_grab_v_bgroup: If the opening delimiter is a left brace, we keep track of how many left and right braces were encountered so far in \l__cmd_v_nesting_int (the methods used for optional arguments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using \peek_meaning_remove:NTF in the __cmd_grab_v_aux:w function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```

2159 \cs_new_protected:Npe \__cmd_grab_v_bgroup:
2160 {
2161   \exp_not:N \__cmd_grab_v_aux_catcodes:
2162   \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
2163   \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
2164   \exp_not:N \__cmd_grab_v_bgroup_loop:
2165 }
2166 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
2167 {
2168   \peek_N_type:TF
2169   { \__cmd_grab_v_bgroup_loop:N }
2170   { \__cmd_grab_v_aux_abort:n { } }
2171 }
2172 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
2173 {
2174   \__cmd_grab_v_token_if_char:NTF #1
2175   {
2176     \token_if_eq_charcode:NNTF \c_group_end_token #1
2177     {
2178       \int_decr:N \l__cmd_v_nesting_int
2179       \int_compare:nNnTF \l__cmd_v_nesting_int > 0
2180       {
2181         \__cmd_grab_v_aux_put:N #1
2182         \__cmd_grab_v_bgroup_loop:
2183       }
2184       { \__cmd_grab_v_aux_loop_end: }
2185     }
2186     {
2187       \token_if_eq_charcode:NNT \c_group_begin_token #1
2188       { \int_incr:N \l__cmd_v_nesting_int }
2189       \__cmd_grab_v_aux_put:N #1
2190       \__cmd_grab_v_bgroup_loop:
2191     }
2192   }
2193   { \__cmd_grab_v_aux_abort:n {#1} #1 }
2194 }

```

(End of definition for __cmd_grab_v_bgroup:, __cmd_grab_v_bgroup_loop:, and __cmd_grab_v_bgroup_loop:N.)

`\__cmd_grab_v_aux_catcodes:` The approach for short verbatim arguments is to make the end-line character a macro parameter character: this is forbidden by the rest of the code. Then the error branch can check what caused the bail out and give the appropriate error message.

```

2195 <latexrelease> \IncludeInRelease{2025/06/01}{\__cmd_grab_v_aux_catcodes}%
2196 <latexrelease> {Active-spaces-and-tabs}
2197 \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
2198 {
2199   \cs_set_eq:NN \do \char_set_catcode_other:N
2200   \dospecials
2201   \char_set_catcode_active:n { '\ }
2202   \char_set_catcode_active:n { '\^^I }
2203   \bool_if:NTF \l__cmd_long_bool
2204     { \char_set_catcode_other:n { \tex_endlinechar:D } }
2205     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
2206 }
2207 <latexrelease> \EndIncludeInRelease
2208 <latexrelease> \IncludeInRelease{2020/10/01}{\__cmd_grab_v_aux_catcodes}%
2209 <latexrelease> {Active-spaces-and-tabs}
2210 <latexrelease> \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
2211 <latexrelease> {
2212 <latexrelease>   \cs_set_eq:NN \do \char_set_catcode_other:N
2213 <latexrelease>   \dospecials
2214 <latexrelease>   \bool_if:NTF \l__cmd_long_bool
2215 <latexrelease>     { \char_set_catcode_other:n { \tex_endlinechar:D } }
2216 <latexrelease>     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
2217 <latexrelease> }
2218 <latexrelease> \EndIncludeInRelease
2219 \cs_new_protected:Npn \__cmd_grab_v_aux_abort:n #1
2220 {
2221   \__cmd_grab_v_group_end:
2222   \exp_after:wN \exp_after:wN \exp_after:wN
2223     \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
2224   {
2225     \msg_error:nneee { cmd } { verbatim-nl }
2226     { \__cmd_environment_or_command: }
2227     { \tl_to_str:N \l__cmd_v_arg_tl }
2228     { \tl_to_str:n {#1} }
2229     \__cmd_add_arg:n { \NoValue }
2230   }
2231   {
2232     \msg_error:nneee { cmd } { verbatim-tokenized }
2233     { \__cmd_environment_or_command: }
2234     { \tl_to_str:N \l__cmd_v_arg_tl }
2235     { \tl_to_str:n {#1} }
2236     \__cmd_add_arg:n { \NoValue }
2237   }
2238 }

```

(End of definition for `\__cmd_grab_v_aux_catcodes:` and `\__cmd_grab_v_aux_abort:n`.)

`\__cmd_grab_v_aux_put:N` Storing one token in the collected argument: everything as-is except for end-of-lines, with `\exp_not:N` to handle actives.

```

2239 <latexrelease> \IncludeInRelease{2025-06-01}{\__cmd_grab_v_aux_put:N}%
2240 <latexrelease> {Use-more-std-catcodes}

```



```

2241 \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2242 {
2243   \tl_put_right:Ne \l__cmd_v_arg_tl
2244   {
2245     \int_compare:nNnTF {'#1} = \tex_endlinechar:D
2246     { \exp_not:N \obeyedline }
2247     { \exp_not:N #1 }
2248   }
2249 }
2250 <latexrelease> \EndIncludeInRelease
2251 <latexrelease> \IncludeInRelease{2024/06/01}{\__cmd_grab_v_aux_put:N}%
2252 <latexrelease> {Endlines~as~\obeyedline}
2253 <latexrelease> \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2254 <latexrelease> {
2255   \tl_put_right:Nx \l__cmd_v_arg_tl
2256   {
2257     \token_if_active:NTF #1
2258     { \exp_not:N #1 }
2259     {
2260       \int_compare:nNnTF {'#1} = \tex_endlinechar:D
2261       { \exp_not:N \obeyedline }
2262       { \token_to_str:N #1 }
2263     }
2264   }
2265 <latexrelease> }
2266 <latexrelease> \EndIncludeInRelease
2267 <latexrelease> \IncludeInRelease{2020/10/01}{\__cmd_grab_v_aux_put:N}%
2268 <latexrelease> {Endlines~as~\obeyedline}
2269 <latexrelease> \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2270 <latexrelease> {
2271   \tl_put_right:Nx \l__cmd_v_arg_tl
2272   {
2273     \token_if_active:NTF #1
2274     { \exp_not:N #1 } { \token_to_str:N #1 }
2275   }
2276 <latexrelease> }
2277 <latexrelease> \EndIncludeInRelease

```

(End of definition for __cmd_grab_v_aux_put:N.)

__cmd_grab_v_token_if_char:N

This function assumes that the escape character is printable. Then the string representation of control sequences is at least two characters, and \str_tail:n only removes the escape character. Macro parameter characters are doubled by \tl_to_str:n, and will also yield a non-empty result, hence are not considered as characters.

```

2278 \cs_new_protected:Npn \__cmd_grab_v_token_if_char:N #1
2279 { \str_if_eq:eeTF { } { \str_tail:n {#1} } }

```

(End of definition for __cmd_grab_v_token_if_char:N.)

__cmd_add_arg:n
 __cmd_add_arg:V
 __cmd_add_arg:o
 __cmd_add_arg:e

When an argument is found it is stored, then further arguments are grabbed by calling \l__cmd_signature_tl.

```

2280 \cs_new_protected:Npn \__cmd_add_arg:n #1
2281 {
2282   \tl_put_right:Nn \l__cmd_args_tl { {#1} }

```

```

2283     \l__cmd_signature_tl \__cmd_run_code:
2284   }
2285   \cs_generate_variant:Nn \__cmd_add_arg:n { V , o , e }

(End of definition for \__cmd_add_arg:n.)

```

1.28 Grabbing arguments expandably

```

\__cmd_expandable_grab_D:w The first step is to grab the first token or group. The generic grabbers \<function>_
\__cmd_expandable_grab_D:NNNwNNn and \<function>_ are just after \q__cmd, we go and find them (and use the long one).
\__cmd_expandable_grab_D:NNNwNNnnn 2286 \cs_new:Npn \__cmd_expandable_grab_D:w #1 \q__cmd #2#3
\__cmd_expandable_grab_D:Nw 2287 { #2 { \__cmd_expandable_grab_D:NNNwNNn #1 \q__cmd #2 #3 } }
\__cmd_expandable_grab_D:nnNNwNN

```

We then wish to test whether #7, which we just grabbed, is exactly #2. A preliminary test is whether their string representations coincide, then expand the only grabber function we have, #1, once: the two strings below are equal if and only if #7 matches #2 exactly.² The preliminary test is needed as #7 could validly contain \par (because a later mandatory argument could be long) and our grabber may be short. If #7 does not match #2, then the optional argument is missing, we use the default \NoValue, and put back the argument #7 in the input stream.

If it does match, then interesting things need to be done. We will grab the argument piece by piece, with the following pattern:

```

<grabber> {<tokens>}
\q_nil {<piece 1>} <piece 2> \ERROR \q__cmd
\q_nil <input stream>

```

The <grabber> will find an opening delimiter in <piece 2>, take the \q__cmd as a second delimiter, and find more material delimited by the closing delimiter in the <input stream>. We then move the part before the opening delimiter from <piece 2> to <piece 1>, and the material taken from the <input stream> to the <piece 2>. Thus, the argument moves gradually from the <input stream> to the <piece 2>, then to the <piece 1> when we have made sure to find all opening and closing delimiters. This two-step process ensures that nesting works: the number of opening delimiters minus closing delimiters in <piece 1> is always equal to the number of closing delimiters in <piece 2>. We stop grabbing arguments once the <piece 2> contains no opening delimiter any more, hence the balance is reached, and the final argument is <piece 1> <piece 2>.

```

2288 \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNn #1#2#3#4 \q__cmd #5#6#7
2289 {
2290   \str_if_eq:nnTF {#2} {#7}
2291   {
2292     \str_if_eq:onTF
2293     { #1 { } { } #7 #2 \q__cmd #3 }
2294     { { } {#2} { } }
2295   }
2296   { \use_ii:nn }

```

²It is obvious that if #7 matches #2 then the strings are equal. We must check the converse. The right-hand-side of \str_if_eq:onTF does not end with #3, implying that the grabber function took everything as its arguments. The first brace group can only be empty if #7 starts with #2, otherwise the brace group preceding #7 would not vanish. The third brace group is empty, thus the \q__cmd that was used by our grabber #1 must be the one that we inserted (not some token in #7), hence the second brace group contains the end of #7 followed by #2. Since this is #2 on the right-hand-side, and no brace can be lost there, #7 must contain nothing else than its leading #2.

```

2297 {
2298   #1
2299   { \_cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6 }
2300   \q_nil { } #2 \ERROR \q__cmd \ERROR
2301 }
2302 { #4 { \NoValue } \q__cmd #5 #6 {#7} }
2303 }

```

At this stage, #7 is `\q_nil {<piece 1>}` *<more for piece 1>*, and we want to concatenate all that, removing `\q_nil`, and keeping the opening delimiter #2. Simply use `\use_ii:nn`. Also, #8 is *<remainder of piece 2>* `\ERROR`, and #9 is `\ERROR` *<more for piece 2>*. We concatenate those, replacing the two `\ERROR` by the closing delimiter #3.

```

2304 \cs_new:Npn \_cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6#7#8#9
2305 {
2306   \exp_args:Nof \_cmd_expandable_grab_D:nnNNNwNN
2307   { \use_ii:nn #7 #2 }
2308   { \_cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
2309   #1#2#3 #4 \q__cmd #5 #6
2310 }
2311 \cs_new:Npn \_cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }

```

Armed with our two new *<pieces>*, we are ready to loop. However, we must first see if *<piece 2>* (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with `\use_none:nnn`. The test is similar to `\tl_if_in:nnTF`. The token list is empty if and only if #2 does not contain the opening delimiter. In that case, we are done, and put the argument (from which we remove a spurious pair of delimiters coming from how we started the loop). Otherwise, we go back to looping with `\_cmd_expandable_grab_D:NNNwNNnnn`. The code to deal with brace stripping is much the same as for the non-expandable case.

```

2312 \cs_new:Npn \_cmd_expandable_grab_D:nnNNNwNN #1#2#3#4#5#6 \q__cmd #7#8
2313 {
2314   \exp_args:No \tl_if_empty:oTF
2315   { #3 { \use_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
2316   {
2317     \tl_if_blank:oTF { \use_none:nn #1#2 }
2318     { \_cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
2319     {
2320       \str_if_eq:eeTF
2321       { \exp_not:o { \use_none:nn #1#2 } }
2322       { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
2323       { \_cmd_put_arg_expandable:ow { \use_iii:nnn #1#2 } }
2324       { \_cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
2325     }
2326     #6 \q__cmd #7 #8
2327   }
2328   {
2329     #3
2330     { \_cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
2331     \q_nil {#1} #2 \ERROR \q__cmd \ERROR
2332   }
2333 }

```

(End of definition for `\_cmd_expandable_grab_D:w` and others.)

```

\__cmd_expandable_grab_D_alt:w
\__cmd_expandable_grab_D_alt:NNwNNn
\__cmd_expandable_grab_D_alt:Nwn

```

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different but there cannot be any nesting.

```

2334 \cs_new:Npn \__cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
2335 { #2 { \__cmd_expandable_grab_D_alt:NNwNNn #1 \q__cmd #2 #3 } }
2336 \cs_new:Npn \__cmd_expandable_grab_D_alt:NNwNNn #1#2#3 \q__cmd #4#5#6
2337 {
2338   \str_if_eq:nnTF {#6} {#2}
2339   {
2340     \str_if_eq:onTF
2341     { #1 { } #6 #2 #2 }
2342     { { } #2 }
2343   }
2344   { \use_ii:nn }
2345   {
2346     #1
2347     { \__cmd_expandable_grab_D_alt:NNwn #4 #5 #3 \q__cmd }
2348     #6 \ERROR
2349   }
2350   { #3 { \NoValue } \q__cmd #4 #5 {#6} }
2351 }
2352 \cs_new:Npn \__cmd_expandable_grab_D_alt:NNwn #1#2#3 \q__cmd #4
2353 {
2354   \tl_if_blank:oTF { \use_none:n #4 }
2355   { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
2356   {
2357     \str_if_eq:eeTF
2358     { \exp_not:o { \use_none:n #4 } }
2359     { { \exp_not:o { \use_ii:nnn #4 \q_nil } } }
2360     { \__cmd_put_arg_expandable:ow { \use_ii:nn #4 } }
2361     { \__cmd_put_arg_expandable:ow { \use_none:n #4 } }
2362   }
2363   #3 \q__cmd #1 #2
2364 }

```

(End of definition for __cmd_expandable_grab_D_alt:w, __cmd_expandable_grab_D_alt:NNwNNn, and __cmd_expandable_grab_D_alt:Nwn.)

```

\__cmd_expandable_grab_E:w
\__cmd_expandable_grab_E_long:w
\__cmd_expandable_grab_E_aux:w
\__cmd_expandable_grab_E_test:nnw
\__cmd_expandable_grab_E_loop:nnnNNw
\__cmd_expandable_grab_E_find:w
\__cmd_expandable_grab_E_find:nnw
\__cmd_expandable_grab_E_end:nnw

```

We keep track of long/short by placing the appropriate grabber as the third token after \q__cmd; it is eventually removed by the end:nnw auxiliary. The aux:w auxiliary will be called repeatedly with two arguments: the set of pairs *<parser>* *<token>*, and the set of arguments found so far (initially all {\NoValue}). At each step, grab what follows in the input stream then call the loop:nnnNNw auxiliary to compare it with each possible embellishment in turn. This auxiliary's #1 is what was found in the input, #2 collects *<parser>* *<token>* pairs that did not match, #3 collects the corresponding arguments found previously, #4 and #5 is the current pair, #6 is the remaining pairs, #7 is empty or two \q_nil, and #8 is the current argument. If none of the pairs matched (determined by \quark_if_nil:NTF) then call the end auxiliary to stop looking for embellishments, remembering to put what was grabbed in the input back where it belongs, and storing the arguments found just before \q__cmd. If the current argument #8 is not \NoValue or if the input #1 does not match #5 (see t-type arguments below for a similar \str_if_eq:onTF test) then carry on the loop. Otherwise, we found a new embellishment: grab the corresponding argument in the input using the find:w auxiliary. To avoid losing

braces around that auxiliary's argument we include a space, which will be eliminated in the next loop through embellishments.

```

2365 \cs_new:Npn \__cmd_expandable_grab_E:w #1 \q__cmd #2#3
2366   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #3 }
2367 \cs_new:Npn \__cmd_expandable_grab_E_long:w #1 \q__cmd #2#3
2368   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #2 }
2369 \cs_new:Npn \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2#3#4
2370   { #2 { \__cmd_expandable_grab_E_test:nnw #1 \q__cmd #2 #3 #4 } }
2371 \cs_new:Npn \__cmd_expandable_grab_E_test:nnw #1#2#3 \q__cmd #4#5#6#7
2372   {
2373     \__cmd_expandable_grab_E_loop:nnnNNw {#7} { } { }
2374     #1 \q_nil \q_nil \q_nil \q_mark #2 \q_nil
2375     #3 \q__cmd #4 #5 #6
2376   }
2377 \cs_new:Npn \__cmd_expandable_grab_E_loop:nnnNNw
2378   #1#2#3#4#5#6 \q_nil #7 \q_mark #8
2379   {
2380     \quark_if_nil:NTF #4
2381     { \__cmd_expandable_grab_E_end:nnw {#1} {#3} }
2382     {
2383       \tl_if_novalue:NTF {#8}
2384       { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
2385       { \use_ii:nn }
2386       { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
2387       {
2388         \__cmd_expandable_grab_E_loop:nnnNNw
2389         {#1} { #2 #4 #5 } { #3 {#8} }
2390         #6 \q_nil #7 \q_mark
2391       }
2392     }
2393   }
2394 \cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q__cmd #2#3#4
2395   { #4 { \__cmd_expandable_grab_E_find:nnw #1 \q__cmd #2 #3 #4 } }
2396 \cs_new:Npn \__cmd_expandable_grab_E_find:nnw #1#2#3 \q_nil #4 \q__cmd #5#6#7#8
2397   { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q__cmd #5 #6 #7 }
2398 \cs_new:Npn \__cmd_expandable_grab_E_end:nnw #1#2#3 \q__cmd #4#5#6
2399   { #3 #2 \q__cmd #4 #5 {#1} }

```

(End of definition for __cmd_expandable_grab_E:w and others.)

__cmd_expandable_grab_m:w
 __cmd_expandable_grab_m_long:w
 __cmd_expandable_grab_m_aux:wNn

The mandatory case is easy: find the auxiliary after the \q__cmd, and use it directly to grab the argument, then correctly position the argument before \q__cmd.

```

2400 \cs_new:Npn \__cmd_expandable_grab_m:w #1 \q__cmd #2#3
2401   { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
2402 \cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q__cmd #2#3
2403   { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
2404 \cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2#3#4
2405   { #1 {#4} \q__cmd #2 #3 }

```

(End of definition for __cmd_expandable_grab_m:w, __cmd_expandable_grab_m_long:w, and __cmd_expandable_grab_m_aux:wNn.)

__cmd_expandable_grab_R:w
 __cmd_expandable_grab_R_aux:NNwNn

Much the same as for the D-type argument, with only the lead-off function varying.

```

2406 \cs_new:Npn \__cmd_expandable_grab_R:w #1 \q__cmd #2#3

```

```

2407 { #2 { \__cmd_expandable_grab_R_aux:NNwNNn #1 \q__cmd #2#3 } }
2408 \cs_new:Npn \__cmd_expandable_grab_R_aux:NNwNNn #1#2#3#4 \q__cmd #5#6#7
2409 {
2410   \str_if_eq:nnTF {#7} {#2}
2411   {
2412     \str_if_eq:onTF
2413     { #1 { } { } #7 #2 \q__cmd #3 }
2414     { { } {#2} { } }
2415   }
2416   { \use_ii:nn }
2417   {
2418     #1
2419     { \__cmd_expandable_grab_D:NNwNNnnn #1#2#3#4 \q__cmd #5#6 }
2420     \q_nil { } #2 \ERROR \q__cmd \ERROR
2421   }
2422   {
2423     \msg_expandable_error:nnff { cmd } { missing-required }
2424     { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #5 } }
2425     { \tl_to_str:n {#2} }
2426     #4 { \NoValue } \q__cmd #5 #6 {#7}
2427   }
2428 }

```

(End of definition for __cmd_expandable_grab_R:w and __cmd_expandable_grab_R_aux:NNwNNn.)

__cmd_expandable_grab_R_alt:w
 __cmd_expandable_grab_R_alt_aux:NNwNNn

When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different.

```

2429 \cs_new:Npn \__cmd_expandable_grab_R_alt:w #1 \q__cmd #2#3
2430 { #2 { \__cmd_expandable_grab_R_alt_aux:NNwNNn #1 \q__cmd #2#3 } }
2431 \cs_new:Npn \__cmd_expandable_grab_R_alt_aux:NNwNNn #1#2#3 \q__cmd #4#5#6
2432 {
2433   \str_if_eq:nnTF {#6} {#2}
2434   {
2435     \str_if_eq:onTF
2436     { #1 { } #6 #2 #2 }
2437     { { } #2 }
2438   }
2439   { \use_ii:nn }
2440   {
2441     #1
2442     { \__cmd_expandable_grab_D_alt:NNwn #4 #5 #3 \q__cmd }
2443     #6 \ERROR
2444   }
2445   {
2446     \msg_expandable_error:nnff { cmd } { missing-required }
2447     { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
2448     { \tl_to_str:n {#2} }
2449     #3 { \NoValue } \q__cmd #4 #5 {#6}
2450   }
2451 }

```

(End of definition for __cmd_expandable_grab_R_alt:w and
 __cmd_expandable_grab_R_alt_aux:NNwNNn.)

`\__cmd_expandable_grab_t:w` As for a D-type argument, here we compare the grabbed tokens using the only parser we have in order to work out if #2 is exactly equal to the output of the grabber.
`\__cmd_expandable_grab_t_aux:NNwn`

```

2452 \cs_new:Npn \__cmd_expandable_grab_t:w #1 \q__cmd #2#3
2453 { #2 { \__cmd_expandable_grab_t_aux:NNwn #1 \q__cmd #2 #3 } }
2454 \cs_new:Npn \__cmd_expandable_grab_t_aux:NNwn #1#2#3 \q__cmd #4#5#6
2455 {
2456   \str_if_eq:onTF { #1 { } #6 #2 } {#2}
2457   { #3 { \BooleanTrue } \q__cmd #4 #5 }
2458   { #3 { \BooleanFalse } \q__cmd #4 #5 {#6} }
2459 }

```

(End of definition for __cmd_expandable_grab_t:w and __cmd_expandable_grab_t_aux:NNwn.)

`\__cmd_put_arg_expandable:nw` A useful helper, to store arguments when they are ready.
`\__cmd_put_arg_expandable:ow`

```

2460 \cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q__cmd { #2 {#1} \q__cmd }
2461 \cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }

```

(End of definition for __cmd_put_arg_expandable:nw.)

1.29 Argument processors

`\__cmd_bool_reverse:N` A simple reversal.

```

2462 \cs_new_protected:Npn \__cmd_bool_reverse:N #1
2463 {
2464   \bool_if:NTF #1
2465   { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
2466   { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
2467 }

```

(End of definition for __cmd_bool_reverse:N.)

$\backslash l_cmd_split_list_seq$ $\backslash l_cmd_split_list_tl$ $\backslash_cmd_split_list_multi:nn$ $\backslash_cmd_split_list_multi:nV$ $\backslash_cmd_split_list_single:Nn$	Splitting can take place either at a single token or at a longer identifier. To deal with single active tokens, a two-part procedure is needed.	2468 $\backslash seq_new:N \backslash l_cmd_split_list_seq$ 2469 $\backslash tl_new:N \backslash l_cmd_split_list_tl$ 2470 $\backslash cs_new_protected:Npn \backslash_cmd_split_list:nn \#1\#2$ 2471 { 2472 $\backslash tl_if_single:nTF \{ \#1 \}$ 2473 { 2474 $\backslash token_if_cs:NTF \#1$ 2475 { $\backslash_cmd_split_list_multi:nn \{ \#1 \} \{ \#2 \}$ } 2476 { $\backslash_cmd_split_list_single:Nn \#1 \{ \#2 \}$ } 2477 } 2478 { $\backslash_cmd_split_list_multi:nn \{ \#1 \} \{ \#2 \}$ } 2479 } 2480 $\backslash cs_new_protected:Npn \backslash_cmd_split_list_multi:nn \#1\#2$ 2481 { 2482 $\backslash seq_set_split:Nnn \backslash l_cmd_split_list_seq \{ \#1 \} \{ \#2 \}$ 2483 $\backslash tl_clear:N \backslash ProcessedArgument$ 2484 $\backslash seq_map_inline:Nn \backslash l_cmd_split_list_seq$ 2485 { $\backslash tl_put_right:Nn \backslash ProcessedArgument \{ \{ \#1 \} \}$ } 2486 } 2487 $\backslash cs_generate_variant:Nn \backslash_cmd_split_list_multi:nn \{ nV \}$ 2488 $\backslash group_begin:$ 2489 $\backslash char_set_catcode_active:N \wedge\wedge@$ 2490 $\backslash cs_new_protected:Npn \backslash_cmd_split_list_single:Nn \#1\#2$ 2491 { 2492 $\backslash tl_set:Nn \backslash l_cmd_split_list_tl \{ \#2 \}$ 2493 $\backslash group_begin:$ 2494 $\backslash char_set_lccode:nn \{ '\wedge\wedge@ \} \{ '\#1 \}$ 2495 $\backslash tex_lowercase:D$ 2496 { 2497 $\backslash group_end:$ 2498 $\backslash tl_replace_all:Nnn \backslash l_cmd_split_list_tl \{ \wedge\wedge@ \}$ 2499 } { $\#1$ } 2500 $\backslash_cmd_split_list_multi:nV \{ \#1 \} \backslash l_cmd_split_list_tl$ 2501 } 2502 $\backslash group_end:$	<i>(End of definition for $\backslash_cmd_split_list:nn$, $\backslash_cmd_split_list_multi:nn$, and $\backslash_cmd_split_list_single:Nn$.)</i>
$\backslash_cmd_split_argument:nnn$ $\backslash_cmd_split_argument_aux:nnnn$ $\backslash_cmd_split_argument_aux:n$ $\backslash_cmd_split_argument_aux:wn$	Splitting to a known number of items is a special version of splitting a list, in which the limit is hard-coded and where there will always be exactly the correct number of output items. An auxiliary function is used to save on working out the token list length several times.	2503 $\backslash cs_new_protected:Npn \backslash_cmd_split_argument:nnn \#1\#2\#3$ 2504 { 2505 $\backslash_cmd_split_list:nn \{ \#2 \} \{ \#3 \}$ 2506 $\backslash exp_args:Nf \backslash_cmd_split_argument_aux:nnnn$ 2507 { $\backslash tl_count:N \backslash ProcessedArgument$ } 2508 { $\#1$ } { $\#2$ } { $\#3$ } 2509 }	


```

2510 \cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
2511 {
2512   \int_compare:nNnF {#1} = { #2 + 1 }
2513   {
2514     \int_compare:nNnTF {#1} > { #2 + 1 }
2515     {
2516       \tl_set:Ne \ProcessedArgument
2517       {
2518         \exp_last_unbraced:NnNo
2519         \__cmd_split_argument_aux:n
2520         { #2 + 1 }
2521         \use_none_delimit_by_q_stop:w
2522         \ProcessedArgument
2523         \q_stop
2524       }
2525       \msg_error:nneee { cmd } { arg-split }
2526       { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
2527       { \tl_to_str:n {#4} }
2528     }
2529   {
2530     \tl_put_right:Ne \ProcessedArgument
2531     {
2532       \prg_replicate:nn { #2 + 1 - (#1) }
2533       { { \NoValue } }
2534     }
2535   }
2536 }
2537 }

```

Auxiliaries to leave exactly the correct number of arguments in \ProcessedArgument.

```

2538 \cs_new:Npn \__cmd_split_argument_aux:n #1
2539 { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
2540 \cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
2541 {
2542   \exp_not:n { {#2} }
2543   #1
2544   \use_none_delimit_by_q_stop:w
2545 }

```

(End of definition for __cmd_split_argument:nnn and others.)

__cmd_trim_spaces:n This one is almost trivial.

```

2546 \cs_new_protected:Npn \__cmd_trim_spaces:n #1
2547 { \tl_set:Ne \ProcessedArgument { \tl_trim_spaces:n {#1} } }

```

(End of definition for __cmd_trim_spaces:n.)

1.30 Conversion to key–value form

This is implemented as a process but with no public interfaces, hence is treated separately from the others: it's a feature of ltcmd which just happens to use the same mechanism as a processor.

```

\__cmd_arg_to_keyvalue:nn
  \__cmd_arg_to_keyvalue_braces:nnn
  \__cmd_arg_to_keyvalue_auxi:nnn
  \__cmd_arg_to_keyvalue_auxii:Nnnn
  \__cmd_arg_to_keyvalue_auxiii:nnn
  \__cmd_arg_to_keyvalue_auxiv:Nnnn
  \__cmd_arg_to_keyvalue_auxv:nn
  \__cmd_arg_to_keyvalue_loop:w
  \__cmd_arg_to_keyvalue_loop_group:n
  \__cmd_arg_to_keyvalue_loop_space:w
  \__cmd_arg_to_keyvalue_loop_N_type:N
  \__cmd_arg_to_keyvalue_math:w
  \__cmd_arg_to_keyvalue_math_N_type:N
  \__cmd_arg_to_keyvalue_math_group:n
  \__cmd_arg_to_keyvalue_math_space:w
  \__cmd_arg_to_keyvalue_set_default:nn
  \__cmd_arg_to_keyvalue_set_keyvalue:nn
\__cmd_split_N_head_apply:Nn
  \__cmd_split_N_head_apply_aux:NNw

```

If the entire argument is braced, we treat as free text and return as the value for the text key. Alternatively, if the start of the input is =, then it is forced to be key–value. To avoid needing to worry about catcodes for this, and to allow spaces around the =, we use a series of steps rather than a delimited argument. We also deal with the case where the entire argument is blank (or is an entirely empty brace group).

```

2548 \cs_new_protected:Npn \__cmd_arg_to_keyvalue:nn #1#2
2549 {
2550   \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_braces:nnn
2551   {#1} {#2}
2552 }
2553 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_braces:nnn #1#2#3
2554 {
2555   \bool_lazy_and:nnTF
2556   { \tl_if_head_is_group_p:n {#1} }
2557   { \tl_if_blank_p:o { \use_none:n #1 } }
2558   { \tl_set:Nx \ProcessedArgument { #2 = { \exp_not:n #1 } } }
2559   {
2560     \tl_if_blank:nTF {#1}
2561     { \tl_clear:N \ProcessedArgument }
2562     { \__cmd_arg_to_keyvalue_auxi:nnn {#1} {#2} {#3} }
2563   }
2564 }
2565 \cs_new:Npn \__cmd_arg_to_keyvalue_auxi:nnn #1
2566 {
2567   \tl_if_head_is_N_type:nTF {#1}
2568   { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxii:Nnnn {#1} }
2569   { \__cmd_arg_to_keyvalue_auxv:nn }
2570 }
2571 \cs_new:Npn \__cmd_arg_to_keyvalue_auxii:Nnnn #1#2
2572 {
2573   \str_if_eq:eeTF { \exp_not:n {#1} } { = }
2574   { \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_auxiii:nnn }
2575   { \__cmd_arg_to_keyvalue_auxv:nn }
2576 }
2577 \cs_new:Npn \__cmd_arg_to_keyvalue_auxiii:nnn #1
2578 {
2579   \tl_if_head_is_N_type:nTF {#1}
2580   { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxiv:Nnnn {#1} }
2581   { \__cmd_arg_to_keyvalue_auxv:nn }
2582 }
2583 \cs_new:Npn \__cmd_arg_to_keyvalue_auxiv:Nnnn #1#2
2584 {
2585   \str_if_eq:eeTF { \exp_not:n {#1} } { , }
2586   { \tl_set:Nx \ProcessedArgument {#2} \use_none:nn }
2587   { \__cmd_arg_to_keyvalue_auxv:nn }
2588 }

```

The two clear-cut cases have been eliminated, and we therefore have to deal with a search for = signs. We need an “action” loop here so we do not get misled by for example {=}. As the code here is for very much predictable types of input, we hard-code what constitutes math mode opening and closing. At the very beginning, the default key (#1) and the argument as given by the user (#2) are placed right after the `\__cmd_recursion_stop`, so that when the recursion ends, the macros `\__cmd_arg_to_keyvalue_set_default:nn` or `\__cmd_arg_to_keyvalue_set_keyvalue:nn` can be used to grab these two items and

set the \ProcessedArgument accordingly.

```

2589 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_auxv:nn #1#2
2590 {
2591   \__cmd_arg_to_keyvalue_loop:w #2
2592   \q__cmd_recursion_tail \q__cmd_recursion_stop {#1} {#2}
2593 }
2594 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop:w #1 \q__cmd_recursion_stop
2595 {
2596   \tl_if_head_is_N_type:nTF {#1}
2597   { \__cmd_arg_to_keyvalue_loop_N_type:N }
2598   {
2599     \tl_if_head_is_group:nTF {#1}
2600     { \__cmd_arg_to_keyvalue_loop_group:n }
2601     { \__cmd_arg_to_keyvalue_loop_space:w }
2602   }
2603   #1 \q__cmd_recursion_stop
2604 }
2605 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_group:n #1
2606 { \__cmd_arg_to_keyvalue_loop:w }
2607 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_space:w } ~
2608 { \__cmd_arg_to_keyvalue_loop:w }
2609 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_N_type:N #1
2610 {
2611   \__cmd_if_recursion_tail_stop_do:Nn #1
2612   { \__cmd_arg_to_keyvalue_set_default:nn }
2613   \str_if_eq:nnTF {#1} { = }
2614   {
2615     \__cmd_use_i_delimit_by_q_recursion_stop:nw
2616     { \__cmd_arg_to_keyvalue_set_keyvalue:nn }
2617   }
2618   {
2619     \bool_lazy_or:nnTF
2620     { \token_if_math_toggle_p:N #1 }
2621     { \str_if_eq_p:nn {#1} { \ ( } }
2622     { \__cmd_arg_to_keyvalue_math:w }
2623     { \__cmd_arg_to_keyvalue_loop:w }
2624   }
2625 }
2626 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math:w #1 \q__cmd_recursion_stop
2627 {
2628   \tl_if_head_is_N_type:nTF {#1}
2629   { \__cmd_arg_to_keyvalue_math_N_type:N }
2630   {
2631     \tl_if_head_is_group:nTF {#1}
2632     { \__cmd_arg_to_keyvalue_math_group:n }
2633     { \__cmd_arg_to_keyvalue_math_space:w }
2634   }
2635   #1 \q__cmd_recursion_stop
2636 }
2637 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_N_type:N #1
2638 {
2639   \__cmd_if_recursion_tail_stop_do:Nn #1
2640   { \__cmd_arg_to_keyvalue_set_default:nn }
2641   \bool_lazy_or:nnTF

```

```

2642         { \token_if_math_toggle_p:N #1 }
2643         { \str_if_eq_p:nn {#1} { \ } } }
2644     { \__cmd_arg_to_keyvalue_loop:w }
2645     { \__cmd_arg_to_keyvalue_math:w }
2646 }
2647 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_group:n #1
2648 { \__cmd_arg_to_keyvalue_math:w }
2649 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_space:w } ~
2650 { \__cmd_arg_to_keyvalue_math:w }
2651 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_default:nn #1#2
2652 { \tl_set:Nn \ProcessedArgument { #1 = {#2} } }
2653 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_keyvalue:nn #1#2
2654 { \tl_set:Nn \ProcessedArgument {#2} }

```

A utility to allow us to grab the first N-type token without risking brace stripping the rest of the input.

```

2655 \cs_new:Npn \__cmd_split_N_head_apply:Nn #1#2
2656 { \exp:w \if_false: { \fi: \__cmd_split_N_head_apply_aux:NNw #1#2 } }
2657 \cs_new:Npn \__cmd_split_N_head_apply_aux:NNw #1#2
2658 {
2659     \exp_after:wN \exp_end:
2660     \exp_after:wN #1 \exp_after:wN #2 \exp_after:wN { \if_false: } \fi:
2661 }
2662

```

(End of definition for __cmd_arg_to_keyvalue:nn and others.)

1.31 Utilities

```

\__cmd_check_definable:nNT
  \__cmd_check_definable_aux:nN

```

Check that a token list is appropriate as a first argument of `\NewDocumentCommand` and similar functions and otherwise produce an error. First trim whitespace to allow for spaces around the actual command to be defined. If the result has multiple tokens, it is not a valid argument. The single token is a control sequence exactly if its string representation has more than one character (using `\token_to_str:N` rather than `\tl_to_str:n` to avoid problems with macro parameter characters, and setting `\tex_escapechar:D` to prevent it from being non-printable). Finally, check for an active character: this is done by lowercasing the token to fix its character code (arbitrarily to that of `?`) and comparing the result to an active `?`. Both control sequences and active characters are valid arguments, and non-active character tokens are not. In all cases, the group opened to keep assignments local must be closed.

```

2663 \cs_new_protected:Npn \__cmd_check_definable:nNT #1
2664 { \tl_trim_spaces_apply:nN {#1} \__cmd_check_definable_aux:nN }
2665 \group_begin:
2666   \char_set_catcode_active:n { '?' }
2667   \cs_new_protected:Npn \__cmd_check_definable_aux:nN #1#2
2668   {
2669     \group_begin:
2670     \tl_if_single_token:nTF {#1}
2671     {
2672       \int_set:Nn \tex_escapechar:D { 92 }
2673       \exp_args:Ne \tl_if_empty:nTF
2674       { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2675       {

```

```

2676         \exp_args:Ne \char_set_lccode:nn
2677         { ' \str_head:n {#1} } { '?' }
2678         \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
2679         { \group_end: \use_iii:nnn }
2680         { \group_end: \use_i:nnn }
2681     }
2682     { \group_end: \use_iii:nnn }
2683 }
2684 { \group_end: \use_ii:nnn }
2685 {
2686     \msg_error:nnee { cmd } { not-definable }
2687     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2688 }
2689 {
2690     \msg_error:nnee { cmd } { not-one-token }
2691     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2692 }
2693 }
2694 \group_end:

```

(End of definition for `\__cmd_check_definable:nNT` and `\__cmd_check_definable_aux:nN`.)

`\__cmd_token_if_cs:N` Based on the definition of `\__cmd_check_definable_aux:nN` above, but only checks for an actual control sequence (*i.e.*, `\<anything>`). `\tex_escapechar:D` is temporarily changed to a known value and then it checks if `\string#1` contains more than one character: if it does, it's a control sequence. This test differs from `\token_if_cs:N` for example in `\token_if_cs:N` `\c_group_begin_token {T}{F}`, where `\token_if_cs:N` returns false.

```

2695 \cs_new_protected:Npn \__cmd_token_if_cs:NTF #1
2696 {
2697     \group_begin:
2698     \int_set:Nn \tex_escapechar:D { 92 }
2699     \exp_args:Ne \tl_if_empty:nTF
2700     { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2701     { \group_end: \use_ii:nn }
2702     { \group_end: \use_i:nn }
2703 }

```

(End of definition for `\__cmd_token_if_cs:N`.)

`\__cmd_tl_mapthread_function:NNN` Analogue of `\seq_mapthread_function:NNN` for token lists.

```

\__cmd_tl_mapthread_function:nnN
\__cmd_tl_mapthread_loop:w
2704 \cs_new:Npn \__cmd_tl_mapthread_function:NNN #1#2#3
2705 {
2706     \exp_after:wN \exp_after:wN
2707     \exp_after:wN \__cmd_tl_mapthread_loop:w
2708     \exp_after:wN \exp_after:wN
2709     \exp_after:wN #3
2710     \exp_after:wN #1
2711     \exp_after:wN \q_recursion_tail
2712     \exp_after:wN \q_mark
2713     #2
2714     \q_recursion_tail
2715     \q_recursion_stop
2716 }

```

```

2717 \cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
2718 {
2719   \__cmd_tl_mapthread_loop:w #3
2720   #1 \q_recursion_tail \q_mark
2721   #2 \q_recursion_tail \q_recursion_stop
2722 }
2723 \cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
2724 {
2725   \quark_if_recursion_tail_stop:n {#2}
2726   \quark_if_recursion_tail_stop:n {#4}
2727   #1 {#2} {#4}
2728   \__cmd_tl_mapthread_loop:w #1#3 \q_mark
2729 }

```

(End of definition for __cmd_tl_mapthread_function:NNN, __cmd_tl_mapthread_function:nnN, and __cmd_tl_mapthread_loop:w.)

```

\__kernel_cmd_if_xparse:NTF
  \__cmd_cmd_type_cases:NnnnnnF
\__cmd_cmd_if_xparse_aux:N

```

To determine whether the command is an xparse command check that its `arg_spec` is empty (this also excludes non-macros) and that its `replacement_spec` starts with either `\__cmd_start:nNNnnn` (non-expandable command) or `\__cmd_start_expandable:nNNNNn` (expandable command) or `\__cmd_start_optimized:` (optimized command) or `\__cmd_start_env:nnnnn` (environment) or `\environment #1 end aux` (environment end).

This conditional is needed in several kernel modules and is therefore has a kernel-internal name.

```

2730 \cs_new:Npn \__cmd_cmd_type_cases:NnnnnnF #1 #2 #3 #4 #5 #6 #7
2731 {
2732   \exp_args:Ne \str_case_e:nnF
2733   {
2734     \exp_args:Nf \tl_if_empty:nT { \__kernel_cs_parameter_spec:N #1 }
2735     { \exp_not:N \exp_not:n { \exp_not:e { \tl_head:N #1 } } }
2736   }
2737   {
2738     { \exp_not:N \__cmd_start:nNNnnn } {#2}
2739     { \exp_not:N \__cmd_start_expandable:nNNNNn } {#3}
2740     { \exp_not:N \__cmd_start_optimized: } {#4}
2741     { \exp_not:N \__cmd_start_env:nnnnn } {#5}
2742     {
2743       \exp_after:wN \exp_not:N
2744       \cs:w environment~
2745       \exp_last_unbraced:Ne \use_none:nnn
2746       { \cs_to_str:N #1 } ~end~aux \cs_end:
2747     } {#6}
2748   }
2749   {#7}
2750 }
2751 \cs_new:Npn \__kernel_cmd_if_xparse:NTF #1
2752 {
2753   \__cmd_cmd_type_cases:NnnnnnF #1
2754   { } { } { } { } { } { } { } { \use_iii:nnn }
2755   \use_i:nn
2756 }

```

(End of definition for __kernel_cmd_if_xparse:NTF, __cmd_cmd_type_cases:NnnnnnF, and __cmd_cmd_if_xparse_aux:N.)

_cmd_peek_nonspace:NTF Collect spaces in a loop, and put the collected spaces back in the false branch of a call to \peek_meaning:NTF or \peek_meaning_remove:NTF.

```

2757 \cs_new_protected:Npn \_cmd_peek_nonspace:NTF
2758 { \_cmd_peek_nonspace_aux:nNNTF { } \_cmd_peek_meaning:NTF }
2759 \cs_new_protected:Npn \_cmd_peek_nonspace_remove:NTF
2760 { \_cmd_peek_nonspace_aux:nNNTF { } \_cmd_peek_meaning_remove:NTF }
2761 \cs_new_protected:Npn \_cmd_peek_nonspace_aux:nNNTF #1#2#3#4#5
2762 {
2763   \peek_meaning_remove:NTF \c_space_token
2764   { \_cmd_peek_nonspace_aux:nNNTF { #1 ~ } #2 #3 {#4} {#5} }
2765   { #2 #3 { #4 } { #5 #1 } }
2766 }

```

(End of definition for _cmd_peek_nonspace:NTF, _cmd_peek_nonspace_remove:NTF, and _cmd_peek_nonspace_aux:nNNTF.)

_cmd_peek_meaning:NTF Peek ahead for a token with a given meaning. In case the search token is a control sequence, also check that the <csname> is the same as the control sequence peeked at. This extra verification is necessary when the command is delimited by control sequence tokens (as opposed to character tokens), and we want the exact same control sequence to match.

```

2767 \cs_new_protected:Npn \_cmd_peek_meaning:NTF
2768 { \_cmd_peek_meaning_aux:NNTF \c_false_bool }
2769 \cs_new_protected:Npn \_cmd_peek_meaning_remove:NTF
2770 { \_cmd_peek_meaning_aux:NNTF \c_true_bool }
2771 \cs_new_protected:Npn \_cmd_peek_meaning_aux:NNTF #1#2#3#4
2772 {
2773   \tl_set:Nn \l__cmd_tmpa_tl {#3}
2774   \tl_set:Nn \l__cmd_tmpb_tl {#4}
2775   \peek_meaning:NTF #2
2776   {
2777     \token_if_eq_meaning:NNTF #2 \c_group_begin_token
2778     { \_cmd_peek_true_remove:Nw #1 }
2779     {
2780       \_cmd_token_if_cs:NTF #2
2781       { \_cmd_peek_cs_check_equal:NNN #1 #2 }
2782       { \_cmd_peek_true_remove:Nw #1 }
2783     }
2784   }
2785   { \l__cmd_tmpb_tl }
2786 }
2787 \cs_new_protected:Npn \_cmd_peek_cs_check_equal:NNN #1#2#3
2788 {
2789   \str_if_eq:nnTF {#2} {#3}
2790   { \_cmd_peek_true_remove:Nw #1 }
2791   { \l__cmd_tmpb_tl }
2792   #3
2793 }
2794 \cs_new_protected:Npn \_cmd_peek_true_remove:Nw #1
2795 {
2796   \bool_if:NTF #1
2797   {
2798     \tex_afterassignment:D \l__cmd_tmpa_tl
2799     \cs_set_eq:NN \_cmd_tmp:w

```

```

2800     }
2801     { \l__cmd_tmpa_tl }
2802 }

```

(End of definition for `\__cmd_peek_meaning:NTF` and others.)

1.32 Access to the argument specification

`\cmd_arg_spec:N` First we check that the command passed does exist, and is one with the correct form.
`\cmd_arg_spec:c` If it does then there are two cases. For optimised commands, we can reconstruct the
`\__cmd_arg_spec_opt:N` arg. spec. from the T_EX parameter spec, taking care to add + if required. For the non-
optimized cases, the arg. spec. is stored in the top-level macro. We treat this as a `tl` and
extract the appropriate balanced text: the second item.

```

2803 \cs_new:Npn \cmd_arg_spec:N #1
2804 {
2805   \cs_if_exist:NTF #1
2806   {
2807     \__kernel_cmd_if_xparse:NTF #1
2808     {
2809       \exp_args:No \tl_if_head_eq_meaning:nNTF #1
2810       \__cmd_start_optimized:
2811       {
2812         \exp_args:Nc \__cmd_arg_spec_opt:N
2813         { \cs_to_str:N #1 \c_space_tl code }
2814       }
2815       { \tl_item:Nn #1 { 2 } }
2816     }
2817     { X }
2818   }
2819   { X }
2820 }
2821 \cs_generate_variant:Nn \cmd_arg_spec:N { c }
2822 \cs_new:Npn \__cmd_arg_spec_opt:N #1
2823 {
2824   \prg_replicate:nn { \str_count:e { \cs_parameter_spec:N #1 } / 2 }
2825   {
2826     \bool_lazy_or:nnT
2827     { \token_if_long_macro_p:N #1 }
2828     { \token_if_protected_long_macro_p:N #1 }
2829     { + }
2830   m
2831   }
2832 }

```

(End of definition for `\cmd_arg_spec:N` and `\__cmd_arg_spec_opt:N`.)

1.33 Messages

```

2833 \tl_const:Nn \c__cmd_ignore_def_tl
2834 { \\ \\ LaTeX-will-ignore-this-entire-definition. }

```

_cmd_environment_or_command: Two texts used in several messages.

```
2835 \cs_new:Npn \_cmd_environment_or_command:
2836 {
2837   \bool_if:NTF \l__cmd_environment_bool
2838     { environment ~ ' \l__cmd_environment_str ' }
2839     {
2840       command ~
2841         ' \c_backslash_str \tl_to_str:N \l__cmd_function_tl '
2842     }
2843 }
```

(End of definition for _cmd_environment_or_command:.)

Some messages intended as errors when defining commands/environments.

```
2844 \msg_new:nnnn { cmd } { arg-after-body }
2845 { Argument~type~'#1'~must~be~last~in~#2. }
2846 {
2847   The~'#1'~argument~type~must~come~last~but~it~is~followed~
2848   by~'#3'~in~the~argument~specification.~This~is~not~allowed.
2849   \c__cmd_ignore_def_tl
2850 }
2851 \msg_new:nnnn { cmd } { bad-arg-spec }
2852 { Bad~argument~specification~'#2'~for~#1. }
2853 {
2854   The~argument~specification~provided~is~not~valid:~
2855   one~or~more~mandatory~parts~are~missing.
2856   \c__cmd_ignore_def_tl
2857 }
2858 \msg_new:nnnn { cmd } { already-defined }
2859 { Command~'#1'~already~defined. }
2860 {
2861   You~have~used~#2~
2862   with~a~command~that~already~has~a~definition. \ \ \
2863   The~existing~definition~of~'#1'~will~not~be~altered.
2864 }
2865 \msg_new:nnnn { cmd } { undefined }
2866 { Command ~'#1'~undefined. }
2867 {
2868   You~have~used~#2~
2869   with~a~command~that~was~never~defined.
2870   \c__cmd_ignore_def_tl
2871 }
2872 \msg_new:nnnn { cmd } { chars-dropped-first-line }
2873 { Characters~'#1'~dropped~on~first~line~of~#2~environment. }
2874 {
2875   LaTeX~was~collecting~a~verbatim~like~environment,~and~the~characters~
2876   '#1'~were~found~after~'\begin{#2}'~on~the~first~line:~this~is~not~supported.
2877 }
2878 \msg_new:nnnn { cmd } { chars-dropped-last-line }
2879 { Characters~'#1'~dropped~after~end~of~#2~environment. }
2880 {
2881   LaTeX~was~collecting~a~verbatim~like~environment,~and~the~characters~
2882   '#1'~were~found~after~'\end{#2}'~on~the~last~line:~this~is~not~supported.
2883 }
```

```

2884 \msg_new:nnnn { cmd } { env-already-defined }
2885 { Environment~'~#1'~already~defined. }
2886 {
2887   You~have~used~\NewDocumentEnvironment
2888   with~an~environment~that~already~has~a~definition. \\ \\
2889   The~existing~definition~of~'~#1'~will~not~be~altered.
2890 }
2891 \msg_new:nnnn { cmd } { env-end-already-defined }
2892 { End~of~environment~'~#1'~already~defined. }
2893 {
2894   You~have~used~\NewDocumentEnvironment
2895   with~an~environment~that~already~has~a~definition~for~'end#1'. \\ \\
2896   The~existing~definition~of~'~#1'~will~not~be~altered.
2897 }
2898 \msg_new:nnnn { cmd } { env-undefined }
2899 { Environment~'~#1'~undefined. }
2900 {
2901   You~have~used~\RenewDocumentEnvironment
2902   with~an~environment~that~was~never~defined.
2903   \c__cmd_ignore_def_tl
2904 }
2905 \msg_new:nnnn { cmd } { expandable-ending-optional }
2906 { Bad~argument~specification~'~#2'~for~#1. }
2907 {
2908   Expandable~commands~must~have~a~final~mandatory~argument~
2909   (or~no~arguments~at~all).~You~cannot~have~a~terminal~optional~
2910   argument~with~expandable~commands.
2911 }
2912 \msg_new:nnnn { cmd } { long-short-mix }
2913 { Invalid~argument~prefix~'+~'~in~command~'~#1'. }
2914 {
2915   The~arguments~for~an~expandable~command~must~not~involve~short~
2916   arguments~after~long~arguments.~You~have~tried~to~mix~the~two~types~
2917   when~defining~'~#1'.
2918 }
2919 \msg_new:nnnn { cmd } { invalid-command-arg }
2920 { Invalid~argument~type~'~#2'~in~#1. }
2921 {
2922   The~letter~'~#2'~can~only~be~used~in~environment~argument~
2923   specifications,~but~not~for~commands.
2924   \\ \\
2925   LaTeX~will~ignore~the~entire~definition.
2926 }
2927 \msg_new:nnnn { cmd } { invalid-expandable-arg }
2928 { Invalid~argument~type~'~#2'~in~#1. }
2929 {
2930   The~letter~'~#2'~specifies~an~argument~type~which~cannot~be~used~
2931   in~an~expandable~command.
2932   \c__cmd_ignore_def_tl
2933 }
2934 \msg_new:nnnn { cmd } { invalid-after-optional-expandably }
2935 { Argument~'~#2'~invalid~after~optional~arg~in~#1. }
2936 {
2937   The~letter~'~#2'~specifies~an~argument~type~which~cannot~be~used~

```

```

2938     in-an-expandable-command-after-an-optional-argument.
2939     \c__cmd_ignore_def_tl
2940   }
2941   \msg_new:nnnn { cmd } { invalid-bang }
2942   { Invalid-argument-prefix~'!'~in~#1. }
2943   {
2944     The-prefix~'!'~is-only-allowed-for-trailing-optional-arguments.~
2945     You-tried-to-apply-it-to~#2.
2946     \c__cmd_ignore_def_tl
2947   }
2948   \msg_new:nnnn { cmd } { not-definable }
2949   { First-argument-of~'#2'~must-be-a-command. }
2950   {
2951     The-first-argument-of~'#2'~should-be-the-document-command-that-will~
2952     be-defined.~The-provided-argument~'#1'~is-a-character.~Perhaps-a~
2953     backslash-is-missing?
2954     \c__cmd_ignore_def_tl
2955   }
2956   \msg_new:nnnn { cmd } { not-one-token }
2957   { First-argument-of~'#2'~must-be-a-command. }
2958   {
2959     The-first-argument-of~'#2'~should-be-the-document-command-that-will~
2960     be-defined.~The-provided-argument~'#1'~contains-more-than-one~
2961     token.~Perhaps-a-backslash-is-missing?
2962     \c__cmd_ignore_def_tl
2963   }
2964   \msg_new:nnnn { cmd } { not-single-token }
2965   { Argument-delimiter~'#2'~invalid-in~#1. }
2966   {
2967     The-argument-specification-contains~
2968     \tl_if_empty:nTF{#2}{nothing}{'#2'}~
2969     in-a-place~
2970     where-a-single-token-is-required.
2971     \c__cmd_ignore_def_tl
2972   }
2973   \msg_new:nnnn { cmd } { forbidden-group-token }
2974   { Argument-delimiter~'#2'~invalid-in~#1. }
2975   {
2976     The-argument-specification-contains-the-implicit~
2977     #3-group-token~'#2'~which-is-not-allowed-as-an-argument-delimiter.
2978     \c__cmd_ignore_def_tl
2979   }
2980   \msg_new:nnnn { cmd } { processor-in-expandable }
2981   { Invalid-argument-prefix~'>'~in-command~'#1'. }
2982   {
2983     The-argument-specification-for~'#1'~contains-the-processor-function~'>{#2}'.~
2984     This-is-only-supported-for-robust-commands,~but-not-for-expandable-ones.
2985     \c__cmd_ignore_def_tl
2986   }
2987   \msg_new:nnnn { cmd } { keyval-in-expandable }
2988   { Invalid-argument-prefix~'='~in-command~'#1'. }
2989   {
2990     The-argument-specification-for~'#1'~contains-a-key--value-marker~'={#2}'.~
2991     This-is-only-supported-for-robust-commands,~but-not-for-expandable-ones.

```

```

2992     \c__cmd_ignore_def_tl
2993   }
2994   \msg_new:nnnn { cmd } { too-many-args }
2995   { Too-many-arguments-for-#1. }
2996   {
2997     The~argument~specification~'#2'~asks~for~more~than~9~arguments.~
2998     This~cannot~be~implemented.
2999     \c__cmd_ignore_def_tl
3000   }
3001   \msg_new:nnnn { cmd } { two-markers }
3002   { Invalid~argument~prefix~'#2'~in~#1. }
3003   {
3004     The~argument~specification~provided~for~#1~has~two~'#2'~markers~applied~
3005     to~the~same~argument;~one~is~redundant.
3006   }
3007   \msg_new:nnnn { cmd } { unknown-argument-type } % should be unkown-arg-type but dep in xparse
3008   { Invalid~argument~type~'#2'~in~#1. }
3009   {
3010     The~letter~'#2'~does~not~specify~a~known~argument~type.
3011     \c__cmd_ignore_def_tl
3012   }
3013   \msg_new:nnnn { cmd } { xparse-arg-type }
3014   { Invalid~argument~type~'#2'~in~#1~(requires~xparse). }
3015   {
3016     The~letter~'#2'~specifies~a~known~but~deprecated~argument~type.~
3017     If~you~really~need~it~you~have~to~load~the~xparse~package.
3018     \c__cmd_ignore_def_tl
3019   }

```

Errors when using commands/environments. The if-boolean message is always used in expandable errors. The default-loop and missing-required messages can be expandable or not expandable.

```

3020   \msg_new:nnn { cmd } { if-boolean }
3021   { Invalid~argument~{#1}~to~\iow_char:N\\IfBoolean... }
3022   \msg_new:nnnn { cmd } { default-loop }
3023   { Circular~dependency~in~defaults~of~#1. }
3024   {
3025     The~default~values~of~two~or~more~arguments~of~the~#1~
3026     depend~on~each~other~in~a~way~that~cannot~be~resolved.
3027   }
3028   \msg_new:nnnn { cmd } { missing-required }
3029   { Required~argument~missing~for~#1. }
3030   {
3031     The~#1~expects~one~of~its~arguments~to~start~with~'#2'.~
3032     LaTeX~did~not~find~this~argument~and~will~insert~a~default~value~
3033     for~further~processing.
3034   }
3035   \msg_new:nnnn { cmd } { arg-split }
3036   { Too~many~'#1'~separators~in~argument. }
3037   {
3038     LaTeX~was~asked~to~split~the~input~'#3'~
3039     at~each~occurrence~of~the~separator~'#1'~into~#2~parts.~
3040     Too~many~separators~were~found.
3041   }

```

```

3042 \msg_new:nnnn { cmd } { verbatim-nl }
3043 { Verbatim-like~#1-ended-by~end-of~line. }
3044 {
3045   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
3046   but~the~end~
3047   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
3048   closing~delimiter.
3049   \\ \\
3050   LaTeX~will~ignore~'#2'~and~you~may~get~some~additional~
3051   (low-level)~errors.
3052 }
3053 \msg_new:nnnn { cmd } { verbatim-tokenized }
3054 { Verbatim-like~#1-illegal~in~argument. }
3055 {
3056   The~#1~takes~a~verbatim~argument~and~should~therefore~normally~
3057   not~be~used~in~arguments~of~other~commands~or~environments.~
3058   LaTeX~found~an~illegal~token~ \tl_if_empty:nF {#3} { (#3)~ }
3059   after~'#2'~and~will~drop~everything~up~to~this~point.
3060   \\ \\
3061   Expect~further~(low-level)~errors.
3062 }
3063 \msg_new:nnn { cmd } { define-command } % should be just 'define' but dep in xparse
3064 {
3065   Defining~command~#1~
3066   with~sig.~'#2'~\msg_line_context:.
3067 }
3068 \msg_new:nnn { cmd } { define-env }
3069 {
3070   Defining~environment~'#1'~
3071   with~sig.~'#2'~\msg_line_context:.
3072 }
3073 \msg_new:nnn { cmd } { redefine }
3074 {
3075   Redefining~command~#1~
3076   with~sig.~'#2'~\msg_line_context:.
3077 }
3078 \msg_new:nnn { cmd } { redefine-env }
3079 {
3080   Redefining~environment~'#1'~
3081   with~sig.~'#2'~\msg_line_context:.
3082 }
3083 \msg_new:nnn { cmd } { optional-mandatory }
3084 {
3085   Optional~and~mandatory~argument~with~same~delimiter~'#2'.
3086   \\ \\
3087   The~mandatory~argument~specified~with~
3088   '\str_case:nnF{#1}{ {R/r}{r'~or~'R} }{#1}'~has~the~
3089   same~delimiter~'#2'~as~an~earlier~optional~argument.~
3090   It~will~therefore~not~be~possible~to~omit~all~the~earlier~
3091   optional~arguments~when~calling~this~command.
3092   \\ \\
3093   This~may~be~intentional,~but~then~it~might~be~a~mistake.
3094 }

```

```

3095 \msg_new:nnn { cmd } { unsupported-let }
3096 {
3097   The~command~'#1'~was~undefined~but~not~the~associated~commands~
3098   '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
3099   \iow_char:N\let.~This~may~lead~to~an~infinite~loop.
3100 }

```

1.34 User functions

The user functions are more or less just the internal functions renamed.

\BooleanFalse Design-space names for the Boolean values.

\BooleanTrue

```

3101 \cs_new_eq:NN \BooleanFalse \c_false_bool
3102 \cs_new_eq:NN \BooleanTrue \c_true_bool

```

(End of definition for \BooleanFalse and \BooleanTrue. These functions are documented on page 121.)

\NewDocumentCommand The user macros are pretty simple wrappers around the internal ones. There is however
\RenewDocumentCommand a check that the first argument is a single token, possibly surrounded by spaces (hence
\ProvideDocumentCommand the strange \use:nnn), and is definable.
\DeclareDocumentCommand

```

3103 \cs_new_protected:Npn \NewDocumentCommand #1#2#3
3104 {
3105   \__cmd_check_definable:nNT {#1} \NewDocumentCommand
3106   {
3107     \cs_if_exist:NTF #1
3108     {
3109       \msg_error:nnee { cmd } { already-defined }
3110       { \use:nnn \token_to_str:N #1 { } }
3111       { \token_to_str:N \NewDocumentCommand }
3112     }
3113     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3114   }
3115 }
3116 \cs_new_protected:Npn \RenewDocumentCommand #1#2#3
3117 {
3118   \__cmd_check_definable:nNT {#1} \RenewDocumentCommand
3119   {
3120     \cs_if_exist:NTF #1
3121     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3122     {
3123       \msg_error:nnee { cmd } { undefined }
3124       { \use:nnn \token_to_str:N #1 { } }
3125       { \token_to_str:N \RenewDocumentCommand }
3126     }
3127   }
3128 }
3129 \cs_new_protected:Npn \ProvideDocumentCommand #1#2#3
3130 {
3131   \__cmd_check_definable:nNT {#1} \ProvideDocumentCommand
3132   { \cs_if_exist:NF #1 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} } }
3133 }
3134 \cs_new_protected:Npn \DeclareDocumentCommand #1#2#3
3135 {

```

```

3136 \__cmd_check_definable:nNT {#1} \DeclareDocumentCommand
3137 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3138 }

```

(End of definition for `\NewDocumentCommand` and others. These functions are documented on page 117.)

`\NewDocumentEnvironment` Very similar for environments. Trim spaces from user-specified `<envname>`, do existence check then hand off to `\__cmd_declare_env:nnnn`.

```

\RenewDocumentEnvironment
\ProvideDocumentEnvironment
\DeclareDocumentEnvironment
3139 <latexrelease>\IncludeInRelease{2024/11/01}{\NewDocumentEnvironment}%
3140 <latexrelease> {Trim-spaces~from~envname~first}
3141 \cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
3142 {
3143   \__cmd_new_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3144 }
3145 \cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
3146 {
3147   \__cmd_renew_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3148 }
3149 \cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
3150 {
3151   \__cmd_provide_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3152 }
3153 \cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
3154 {
3155   \__cmd_declare_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3156 }

```

Each of `\__cmd_(new|renew|provide)_env:nnnn` is curried.

```

3157 \cs_new_protected:Npn \__cmd_new_env:nnnn #1
3158 {
3159   \cs_if_exist:cTF {#1}
3160   {
3161     \msg_error:nne { cmd } { env-already-defined } {#1}
3162     \use_none:nnn
3163   }
3164   {
3165     \cs_if_exist:cTF { end #1 }
3166     {
3167       \msg_error:nne { cmd } { env-end-already-defined } {#1}
3168       \use_none:nnn
3169     }
3170     { \__cmd_declare_env:nnnn {#1} }
3171   }
3172 }
3173 \cs_new_protected:Npn \__cmd_renew_env:nnnn #1
3174 {
3175   \cs_if_exist:cTF {#1}
3176   { \__cmd_declare_env:nnnn {#1} }
3177   {
3178     \msg_error:nne { cmd } { env-undefined } {#1}
3179     \use_none:nnn
3180   }
3181 }
3182 \cs_new_protected:Npn \__cmd_provide_env:nnnn #1
3183 {

```

```

3184 \cs_if_exist:cTF {#1}
3185 { \use_none:nnn }
3186 { \__cmd_declare_env:nnnn {#1} }
3187 }
3188 \cs_generate_variant:Nn \__cmd_new_env:nnnn { e }
3189 \cs_generate_variant:Nn \__cmd_renew_env:nnnn { e }
3190 \cs_generate_variant:Nn \__cmd_provide_env:nnnn { e }
3191 <latexrelease>\EndIncludeInRelease
3192 <latexrelease>\IncludeInRelease{2024/06/01}{\NewDocumentEnvironment}%
3193 <latexrelease> {Trim~spaces~from~envname~first}
3194 <latexrelease>\cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
3195 <latexrelease> {
3196 <latexrelease> \cs_if_exist:cTF {#1}
3197 <latexrelease> { \msg_error:nnx { cmd } { env-already-defined } {#1} }
3198 <latexrelease> {
3199 <latexrelease> \cs_if_exist:cTF { end #1 }
3200 <latexrelease> { \msg_error:nnx { cmd } { env-end-already-defined } {#1} }
3201 <latexrelease> { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
3202 <latexrelease> }
3203 <latexrelease> }
3204 <latexrelease>\cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
3205 <latexrelease> {
3206 <latexrelease> \cs_if_exist:cTF {#1}
3207 <latexrelease> { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
3208 <latexrelease> { \msg_error:nnx { cmd } { env-undefined } {#1} }
3209 <latexrelease> }
3210 <latexrelease>\cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
3211 <latexrelease> { \cs_if_exist:cF {#1} { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} } }
3212 <latexrelease>\cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
3213 <latexrelease> { \__cmd_declare_env:nnnn {#1} {#2} {#3} {#4} }
3214 <latexrelease>\cs_undefine:N \__cmd_new_env:nnnn
3215 <latexrelease>\cs_undefine:N \__cmd_new_env:ennn
3216 <latexrelease>\cs_undefine:N \__cmd_renew_env:nnnn
3217 <latexrelease>\cs_undefine:N \__cmd_renew_env:ennn
3218 <latexrelease>\cs_undefine:N \__cmd_provide_env:nnnn
3219 <latexrelease>\cs_undefine:N \__cmd_provide_env:ennn
3220 <latexrelease>
3221 <latexrelease>\EndIncludeInRelease

```

(End of definition for \NewDocumentEnvironment and others. These functions are documented on page 117.)

\NewExpandableDocumentCommand
\RenewExpandableDocumentCommand
\ProvideExpandableDocumentCommand
\DeclareExpandableDocumentCommand

The expandable versions are essentially the same as the basic functions. The strange **\use:nnn** is there in case #1 is surrounded with spaces, as can happen with usual document catcodes in **\RenewExpandableDocumentCommand { \! } ...**

```

3222 \cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
3223 {
3224 \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand
3225 {
3226 \cs_if_exist:NTF #1
3227 {
3228 \msg_error:nnee { cmd } { already-defined }
3229 { \use:nnn \token_to_str:N #1 { } }
3230 { \token_to_str:N \NewExpandableDocumentCommand }

```



```

3231     }
3232     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3233   }
3234 }
3235 \cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
3236 {
3237   \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
3238   {
3239     \cs_if_exist:NTF #1
3240     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3241     {
3242       \msg_error:nnee { cmd } { undefined }
3243       { \use:nnn \token_to_str:N #1 { } }
3244       { \token_to_str:N \RenewExpandableDocumentCommand }
3245     }
3246   }
3247 }
3248 \cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
3249 {
3250   \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
3251   {
3252     \cs_if_exist:NF #1
3253     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3254   }
3255 }
3256 \cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
3257 {
3258   \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
3259   { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3260 }

```

(End of definition for \NewExpandableDocumentCommand and others. These functions are documented on page 125.)

\IfBooleanT The logical `<true>` and `<false>` statements are just the normal `\c_true_bool` and `\c_false_bool` so `\bool_if:N` is almost enough. However, this code-level function blows up badly when passed invalid input. We want `\IfBooleanTF` to accept a single (non-space) token equal to `\c_true_bool` or `\c_false_bool`, possibly surrounded by spaces. If the input is blank or multiple items, jump to the error and pick the false branch. If the input, ignoring spaces (we do this by omitting braces in the `\tl_if_single_token:n` test), is not a single token then jump to the error as well. It is then safe to compare the token to the two booleans, picking the appropriate branch. If neither matches, we jump to the error as well.

```

3261 \cs_new:Npn \IfBooleanTF #1
3262 {
3263   \tl_if_single:nF {#1}
3264   { \prg_break:n { \use:n } }
3265   \tl_if_single_token:nF #1
3266   { \prg_break:n { \use:n } }
3267   \token_if_eq_meaning:NNT #1 \c_true_bool
3268   { \prg_break:n { \use_ii:nnn } }
3269   \token_if_eq_meaning:NNT #1 \c_false_bool
3270   { \prg_break:n { \use_iii:nnn } }
3271   \prg_break:n { \use:n }

```

```

3272 \prg_break_point:
3273 {
3274 \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
3275 \use_i:nn
3276 }
3277 }
3278 \cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
3279 \cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }

```

(End of definition for `\IfBooleanT`, `\IfBooleanF`, and `\IfBooleanTF`. These functions are documented on page 121.)

We adjust here as currently there is still an older version of `\NoValue` in `l3tl`.

`\NoValue` A special marker; done using `gset` as there is also a version in `expl3`, and that has to be in line with what we do here (once everything is synced up, this one will be removed).

```

3280 \cs_gset_protected:Npn \NoValue { -NoValue- }

```

(End of definition for `\NoValue`.)

`\c_novalue_tl`

```

3281 \cs_undefine:N \c_novalue_tl
3282 \tl_const:Nn \c_novalue_tl { \NoValue }

```

`\tl_if_novalue_p:n` The first argument of `\__cmd_if_novalue_aux:w` is empty if and only if `#1` starts with `\NoValue`, while the second argument is empty if `#1` is exactly `\NoValue` or if it has a question mark just following the marker. In this second case, however, the material after the first `?!` remains and makes the emptiness test return `false`.

```

3283 <@@=tl>
3284 \prg_gset_conditional:Npnn \tl_if_novalue:n #1
3285 { p , T , F , TF }
3286 {
3287 \__tl_if_empty_if:o
3288 { \__tl_if_novalue_aux:w { } #1 { } ? ! \NoValue ? ? ! }
3289 \prg_return_true:
3290 \else:
3291 \prg_return_false:
3292 \fi:
3293 }
3294 \cs_gset:Npn \__tl_if_novalue_aux:w #1 \NoValue #2 ? #3 ? ! { #1 #2 }
3295 <@@=cmd>

```

(End of definition for `\tl_if_novalue:nTF` and `\__cmd_if_novalue_aux:w`.)

`\IfNoValueT` Simple re-naming.

```

\IfNoValueF 3296 \cs_new_eq:NN \IfNoValueF \tl_if_novalue:nF
\IfNoValueTF 3297 \cs_new_eq:NN \IfNoValueT \tl_if_novalue:nT
3298 \cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF

```

(End of definition for `\IfNoValueT`, `\IfNoValueF`, and `\IfNoValueTF`. These functions are documented on page 119.)

`\IfValueT` Inverted logic.

```

\IfValueF 3299 \cs_new:Npn \IfValueF { \tl_if_novalue:nT }
\IfValueTF 3300 \cs_new:Npn \IfValueT { \tl_if_novalue:nF }
3301 \cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }

```

(End of definition for \IfValueT, \IfValueF, and \IfValueTF. These functions are documented on page 120.)

\IfBlankT Another simple re-naming.
\IfBlankF 3302 <latexrelease>\IncludeInRelease{2022/06/01}%
\IfBlankTF 3303 <latexrelease> { \IfBlankTF } { Testing~for~empty~or~blank } %
3304 \cs_new_eq:NN \IfBlankF \tl_if_blank:nF
3305 \cs_new_eq:NN \IfBlankT \tl_if_blank:nT
3306 \cs_new_eq:NN \IfBlankTF \tl_if_blank:nTF
3307 <latexrelease>\EndIncludeInRelease
3308 <latexrelease>\IncludeInRelease{2021/11/15}%
3309 <latexrelease> { \IfBlankTF } { Testing~for~empty~or~blank } %
3310 <latexrelease>\cs_undefine:N \IfBlankF
3311 <latexrelease>\cs_undefine:N \IfBlankT
3312 <latexrelease>\cs_undefine:N \IfBlankTF
3313 <latexrelease>
3314 <latexrelease>\EndIncludeInRelease

(End of definition for \IfBlankT, \IfBlankF, and \IfBlankTF. These functions are documented on page 120.)

\ProcessedArgument Processed arguments are returned using this name, which is reserved here although the definition will change.

3315 \tl_new:N \ProcessedArgument

(End of definition for \ProcessedArgument. This function is documented on page 129.)

\ReverseBoolean Simple copies.

\SplitArgument 3316 \cs_new_eq:NN \ReverseBoolean __cmd_bool_reverse:N
\SplitList 3317 \cs_new_eq:NN \SplitArgument __cmd_split_argument:nnn
\TrimSpaces 3318 \cs_new_eq:NN \SplitList __cmd_split_list:nn
3319 \cs_new_eq:NN \TrimSpaces __cmd_trim_spaces:n

(End of definition for \ReverseBoolean and others. These functions are documented on page 124.)

\ProcessList To support \SplitList.

3320 \cs_new_eq:NN \ProcessList \tl_map_tokens:nn

(End of definition for \ProcessList. This function is documented on page 124.)

Finally as promised, restore __kernel_chk_if_free_cs:N:

3321 <latexrelease>\cs_gset_eq:NN __kernel_chk_if_free_cs:N __cmd_chk_if_free_cs:N
3322 <latexrelease>\cs_undefine:N __cmd_chk_if_free_cs:N
3323 <latexrelease>
3324 <latexrelease>\IncludeInRelease{0000/00/00}{ltxcmd}%
3325 <latexrelease> { Document~command~parser } %
3326 <latexrelease>
3327 <latexrelease>\EndModuleRelease
3328 \ExplSyntaxOff

Now in latexrelease mode, redefine \NewDocumentCommand to not complain on commands already defined.

3329 <latexrelease>\@ifundefined{ExplSyntaxOff}{\{\}\latexrelease@postltxcmd}
3330 <latexrelease>\catcode'\^~@=\@latexrelease@catcode@null\relax
3331 </2ekernel | latexrelease>

We need to stop DocStrip treating @@ in a special way at this point.

3332 <@@= >

File 08

lthooks.dtx

1 Introduction

Hooks are points in the code of commands or environments where it is possible to add processing code into existing commands. This can be done by different packages that do not know about each other, and to allow for hopefully safe processing it is necessary to sort different chunks of code added by different packages into a suitable processing order.

This is done by the packages adding chunks of code (via `\AddToHook`) and labeling their code with some label by default using the package name as a label.

At `\begin{document}` all code for a hook is then sorted according to some rules (given by `\DeclareHookRule`) for fast execution without processing overhead. If the hook code is modified afterwards (or the rules are changed), a new version for fast processing is generated.

Some hooks are used already in the preamble of the document. If that happens then the hook is prepared for execution (and sorted) already at that point.

2 Package writer interface

The hook management system is offered as a set of CamelCase commands for traditional $\text{\LaTeX 2}_\epsilon$ packages (and for use in the document preamble if needed) as well as `expl3` commands for modern packages, that use the L3 programming layer of $\text{\LaTeX}$. Behind the scenes, a single set of data structures is accessed so that packages from both worlds can coexist and access hooks in other packages.

2.1 $\text{\LaTeX 2}_\epsilon$ interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands and environments (executed at `\begin` and `\end`), and the generic hooks run when loading files (see section 3.1).

<code>\NewHook</code>	<code>\NewHook {<hook>}</code>
-----------------------	--------------------------------------

Creates a new `<hook>`. If this hook is declared within a package it is suggested that its name is always structured as follows: `<package-name>/<hook-name>`. If necessary you can further subdivide the name by adding more / parts. If a hook name is already taken, an error is raised and the hook is not created.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The string `??` can't be used as a hook name because it has a special significance as a placeholder in hook rules.

<code>\NewReversedHook</code>	<code>\NewReversedHook {⟨hook⟩}</code>
-------------------------------	--

Like `\NewHook` declares a new `⟨hook⟩`. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\NewMirroredHookPair</code>	<code>\NewMirroredHookPair {⟨hook-1⟩} {⟨hook-2⟩}</code>
-----------------------------------	---

A shorthand for `\NewHook{⟨hook-1⟩}\NewReversedHook{⟨hook-2⟩}`.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\NewHookWithArguments</code>	<code>\NewHookWithArguments {⟨hook⟩} {⟨number⟩}</code>
------------------------------------	--

Creates a new `⟨hook⟩` whose code takes `⟨number⟩` arguments, and otherwise works exactly like `\NewHook`. Section 2.7 explains hooks with arguments.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\NewReversedHookWithArguments</code>	<code>\NewReversedHookWithArguments {⟨hook⟩} {⟨number⟩}</code>
--	--

Like `\NewReversedHook`, but creates a hook whose code takes `⟨number⟩` arguments. Section 2.7 explains hooks with arguments.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\NewMirroredHookPairWithArguments</code>	<code>\NewMirroredHookPairWithArguments {⟨hook-1⟩} {⟨hook-2⟩} {⟨number⟩}</code>
--	---

A shorthand for `\NewHookWithArguments{⟨hook-1⟩}{⟨number⟩}`

`\NewReversedHookWithArguments{⟨hook-2⟩}{⟨number⟩}`. Section 2.7 explains hooks with arguments.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.2 Special declarations for generic hooks

The declarations here should normally not be used. They are available to provide support for special use cases mainly involving generic command hooks.

<code>\DisableGenericHook</code>	<code>\DisableGenericHook {⟨hook⟩}</code>
----------------------------------	---

After this declaration³ the `⟨hook⟩` is no longer usable: Any further attempt to add code to it will result in an error and any use, e.g., via `\UseHook`, will simply do nothing.

This is intended to be used with generic command hooks (see `lthcmdhooks-doc`) as depending on the definition of the command such generic hooks may be unusable. If that is known, a package developer can disable such hooks up front.

The `⟨hook⟩` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

³In the 2020/06 release this command was called `\DisableHook`, but that name was misleading as it shouldn't be used to disable non-generic hooks.

\ActivateGenericHook \ActivateGenericHook {<hook>}

This declaration activates a generic hook provided by a package/class (e.g., one used in code with \UseHook or \UseOneTimeHook) without it being explicitly declared with \NewHook). If the hook is already activated, this command does nothing.

Note that this command does not undo the effect of \DisableGenericHook. See section 2.6 for a discussion of when this declaration is appropriate.

2.1.3 Using hooks in code

Using a hook that is executing the code that has been associated with it is only allowed if the hook has been previously declared with \NewHook. For performance reason there are no runtime checks for this and it is the responsibility of the programmer of a package to ensure that all hooks that are used in a package (with one of the commands in this section) are declared first.

\UseHook \UseHook {<hook>}

Execute the code stored in the <hook>.

Before \begin{document} the fast execution code for a hook is not set up, so in order to use a hook there it is explicitly initialized first. As that involves assignments using a hook at those times is not 100% the same as using it after \begin{document}.

The <hook> *cannot* be specified using the dot-syntax. A leading . is treated literally.

\UseHookWithArguments \UseHookWithArguments {<hook>} {<number>} {<arg₁>} ... {<arg_n>}

Execute the code stored in the <hook> and pass the arguments {<arg₁>} through {<arg_n>} to the <hook>. Otherwise, it works exactly like \UseHook. The <number> should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove <number> items from the input. Section 2.7 explains hooks with arguments.

The <hook> *cannot* be specified using the dot-syntax. A leading . is treated literally.

\UseOneTimeHook \UseOneTimeHook {<hook>}

Some hooks are only used (and can be only used) in one place, for example, those in \begin{document} or \end{document}. From that point onwards, adding to the hook through a defined \<addto-cmd> command (e.g., \AddToHook or \AtBeginDocument, etc.) would have no effect (as would the use of such a command inside the hook code itself). It is therefore customary to redefine \<addto-cmd> to simply process its argument, i.e., essentially make it behave like \@firstofone.

\UseOneTimeHook does that: it records that the hook has been consumed and any further attempt to add to it will result in executing the code to be added immediately.

Using \UseOneTimeHook several times with the same {<hook>} means that it only executes the first time it is used. For example, if it is used in a command that can be called several times then the hook executes during only the *first* invocation of that command; this allows its use as an “initialization hook”.

Mixing \UseHook and \UseOneTimeHook for the same {<hook>} should be avoided, but if this is done then neither will execute after the first \UseOneTimeHook.

The <hook> *cannot* be specified using the dot-syntax. A leading . is treated literally. See section 2.1.5 for details.

<code>\UseOneTimeHookWithArguments</code>	<code>\UseOneTimeHookWithArguments {<hook>} {<number>} {<arg₁>} ... {<arg_n>}</code>
---	---

Works exactly like `\UseOneTimeHook`, but passes arguments `{<arg1>}` through `{<argn>}` to the `<hook>`. The `<number>` should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove `<number>` items from the input.

It should be noted that after a one-time hook is used, it is no longer possible to use `\AddToHookWithArguments` or similar with that hook. `\AddToHook` continues to work as normal. Section 2.7 explains hooks with arguments.

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

2.1.4 Updating code for hooks

In contrast to the commands from the previous section, declarations such as `\AddToHook` or `\DeclareHookRule` can be used even when the hook is not yet declared. The rationale is that the hook declaration may be in some package that is loaded later, or perhaps not loaded at all.

A side effect of this design is that misspellings do not raise an error but are simply regarded as declarations for hooks with a different name.

<code>\AddToHook</code>	<code>\AddToHook {<hook>} [<label>] {<code>}</code>
-------------------------	---

Adds `<code>` to the `<hook>` labeled by `<label>`. When the optional argument `<label>` is not provided, the `<default label>` is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the `<default label>` is the package/class name, otherwise it is `top-level` (the `top-level` label is treated differently: see section 2.1.6).

If there already exists code under the `<label>` then the new `<code>` is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the `<label>`, first apply `\RemoveFromHook`.

The hook doesn't have to exist for code to be added to it. However, if it is not declared, then obviously the added `<code>` will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\AddToHookWithArguments</code>	<code>\AddToHookWithArguments {<hook>} [<label>] {<code>}</code>
--------------------------------------	--

Works exactly like `\AddToHook`, except that the `<code>` can access the arguments passed to the hook using `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). If the `<code>` should contain *parameter tokens* (`#`) that are not supposed to be understood as the arguments of the hook, such tokens should be doubled. For example, with `\AddToHook` one can write:

```
\AddToHook{myhook}{\def\foo#1{Hello, #1!}}
```

but to achieve the same with `\AddToHookWithArguments`, one should write:

```
\AddToHookWithArguments{myhook}{\def\foo##1{Hello, ##1!}}
```

because in the latter case, `#1` refers to the first argument of the hook `myhook`. Section 2.7 explains hooks with arguments.

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\RemoveFromHook</code>	<code>\RemoveFromHook {<hook>} [<label>]</code>
------------------------------	---

Removes any code labeled by `<label>` from the `<hook>`. When the optional argument `<label>` is not provided, the `<default label>` is used (see section 2.1.5).

If there is no code under the `<label>` in the `<hook>`, or if the `<hook>` does not exist, a warning is issued when you attempt to `\RemoveFromHook`, and the command is ignored. `\RemoveFromHook` should be used only when you know exactly what labels are in a hook. Typically this will be when some code gets added to a hook by a package, then later this code is removed by that same package. If you want to prevent the execution of code from another package, use the `voids` rule instead (see section 2.1.7).

If the optional `<label>` argument is `*`, then all code chunks are removed. This is rather dangerous as it may well drop code from other packages (that one may not know about); it should therefore not be used in packages but only in document preambles!

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

In contrast to the `voids` relationship between two labels in a `\DeclareHookRule` this is a destructive operation as the labeled code is removed from the hook data structure, whereas the relationship setting can be undone by providing a different relationship later.

A useful application for this declaration inside the document body is when one wants to temporarily add code to hooks and later remove it again, e.g.,

```
\AddToHook{env/quote/begin}{\small}
\begin{quote}
  A quote set in a smaller typeface
\end{quote}
...
\RemoveFromHook{env/quote/begin}
... now back to normal for further quotes
```

Note that you can't cancel the setting with

```
\AddToHook{env/quote/begin}{}
```

Important:
The `\RemoveFromHook` command should be only used if one has full control over the code chunk to be removed. In particular it should not be used to remove code chunks from other packages! For this the `voids` relation is provided.

because that only “adds” a further empty chunk of code to the hook. Adding `\normalsize` would work but that means the hook then contained `\small\normalsize` which means two font size changes for no good reason.

The above is only needed if one wants to typeset several quotes in a smaller typeface. If the hook is only needed once then `\AddToHookNext` is simpler, because it resets itself after one use.

`\AddToHookNext` `\AddToHookNext {<hook>} {<code>}`

Adds `<code>` to the next invocation of the `<hook>`. The code is executed after the normal hook code has finished and it is executed only once, i.e. it is deleted after it was used.

Using this declaration is a global operation, i.e., the code is not lost even if the declaration is used inside a group and the next invocation of the hook happens after the end of that group. If the declaration is used several times before the hook is executed then all code is executed in the order in which it was declared.⁴

If this declaration is used with a one-time hook then the code is only ever used if the declaration comes before the hook’s invocation. This is because, in contrast to `\AddToHook`, the code in this declaration is not executed immediately in the case when the invocation of the hook has already happened—in other words, this code will truly execute only on the next invocation of the hook (and in the case of a one-time hook there is no such “next invocation”). This gives you a choice: should my code execute always, or should it execute only at the point where the one-time hook is used (and not at all if this is impossible)? For both of these possibilities there are use cases.

It is possible to nest this declaration using the same hook (or different hooks): e.g.,

`\AddToHookNext{<hook>}{<code-1>\AddToHookNext{<hook>}{<code-2>}}`

will execute `<code-1>` next time the `<hook>` is used and at that point puts `<code-2>` into the `<hook>` so that it gets executed on following time the hook is run.

A hook doesn’t have to exist for code to be added to it. This allows for hooks to work regardless of package loading order. See section 2.1.8.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookNextWithArguments` `\AddToHookNextWithArguments {<hook>} {<code>}`

Works exactly like `\AddToHookNext`, but the `<code>` can contain references to the arguments of the `<hook>` as described for `\AddToHookWithArguments` above. Section 2.7 explains hooks with arguments.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\ClearHookNext` `\ClearHookNext {<hook>}`

Normally `\AddToHookNext` is only used when you know precisely where it will apply and why you want some extra code at that point. However, there are a few use cases in which such a declaration needs to be canceled, for example, when discarding a page with `\DiscardShipoutBox` (but even then not always), and in such situations `\ClearHookNext` can be used.

⁴There is no mechanism to reorder such code chunks (or delete them).

2.1.5 Hook names and default labels

It is best practice to use `\AddToHook` in packages or classes *without specifying a `<label>`* because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the `<label>`.

Using an explicit `<label>` is only necessary in very specific situations, e.g., if you want to add several chunks of code into a single hook and have them placed in different parts of the hook (by providing some rules).

The other case is when you develop a larger package with several sub-packages. In that case you may want to use the same `<label>` throughout the sub-packages in order to avoid that the labels change if you internally reorganize your code.

Except for `\UseHook`, `\UseOneTimeHook` and `\IfHookEmptyTF` (and their `expl3` interfaces `\hook_use:n`, `\hook_use_once:n` and `\hook_if_empty:nTF`), all `<hook>` and `<label>` arguments are processed in the same way: first, spaces are trimmed around the argument, then it is fully expanded until only character tokens remain. If the full expansion of the `<hook>` or `<label>` contains a non-expandable non-character token, a low-level `TeX` error is raised (namely, the `<hook>` is expanded using `TeX`'s `\csname...\endcsname`, as such, Unicode characters are allowed in `<hook>` and `<label>` arguments). The arguments of `\UseHook`, `\UseOneTimeHook`, and `\IfHookEmptyTF` are processed much in the same way except that spaces are not trimmed around the argument, for better performance.

It is not enforced, but highly recommended that the hooks defined by a package, and the `<labels>` used to add code to other hooks contain the package name to easily identify the source of the code chunk and to prevent clashes. This should be the standard practice, so this hook management code provides a shortcut to refer to the current package in the name of a `<hook>` and in a `<label>`. If the `<hook>` name or the `<label>` consist just of a single dot (`.`), or starts with a dot followed by a slash (`./`) then the dot denotes the `<default label>` (usually the current package or class name—see `\SetDefaultHookLabel`). A `“.”` or `“./”` anywhere else in a `<hook>` or in `<label>` is treated literally and is not replaced.

For example, inside the package `mypackage.sty`, the default label is `mypackage`, so the instructions:

```
\NewHook    {./hook}
\AddToHook {./hook}[.]{code}      % Same as \AddToHook{./hook}{code}
\AddToHook {./hook}[./sub]{code}
\DeclareHookRule{begindocument}{.}{before}{babel}
\AddToHook {file/foo.tex/after}{code}
```

are equivalent to:

```
\NewHook    {mypackage/hook}
\AddToHook {mypackage/hook}[mypackage]{code}
\AddToHook {mypackage/hook}[mypackage/sub]{code}
\DeclareHookRule{begindocument}{mypackage}{before}{babel}
\AddToHook {file/foo.tex/after}{code} % unchanged
```

The `<default label>` is automatically set equal to the name of the current package or class at the time the package is loaded. If the hook command is used outside of a package, or the current file wasn't loaded with `\usepackage` or `\documentclass`, then the `top-level` is used as the `<default label>`. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all $\langle label \rangle$ arguments and most $\langle hook \rangle$ arguments, both in the $\text{\LaTeX} 2_{\epsilon}$ interface, and the $\text{\LaTeX} 3$ interface described in section 2.2.

Important:

*The dot-syntax is **not** available with `\UseHook` and some other commands that are typically used within code!*

Note, however, that the replacement of `.` by the $\langle default label \rangle$ takes place when the hook command is executed, so actions that are somehow executed after the package ends will have the wrong $\langle default label \rangle$ if the dot-syntax is used. For that reason, this syntax is not available in `\UseHook` (and `\hook_use:n`) because the hook is most of the time used outside of the package file in which it was defined. This syntax is also not available in the hook conditionals `\IfHookEmptyTF` (and `\hook_if_empty:nTF`), because these conditionals are used in some performance-critical parts of the hook management code, and because they are usually used to refer to other package's hooks, so the dot-syntax doesn't make much sense.

In some cases, for example in large packages, one may want to separate the code in logical parts, but still use the main package name as the $\langle label \rangle$, then the $\langle default label \rangle$ can be set using `\PushDefaultHookLabel{...}\PopDefaultHookLabel` or `\SetDefaultHookLabel{...}`.

<code>\PushDefaultHookLabel</code>	<code>\PushDefaultHookLabel {$\langle default label \rangle$}</code>
<code>\PopDefaultHookLabel</code>	<code>$\langle code \rangle$</code>

`\PopDefaultHookLabel`

`\PushDefaultHookLabel` sets the current $\langle default label \rangle$ to be used in $\langle label \rangle$ arguments, or when replacing a leading “.” (see above). `\PopDefaultHookLabel` reverts the $\langle default label \rangle$ to its previous value.

Inside a package or class, the $\langle default label \rangle$ is equal to the package or class name, unless explicitly changed. Everywhere else, the $\langle default label \rangle$ is `top-level` (see section 2.1.6) unless explicitly changed.

The effect of `\PushDefaultHookLabel` holds until the next `\PopDefaultHookLabel`. `\usepackage` (and `\RequirePackage` and `\documentclass`) internally use

```
\PushDefaultHookLabel{ $\langle package name \rangle$ }
 $\langle package code \rangle$ 
\PopDefaultHookLabel
```

to set the $\langle default label \rangle$ for the package or class file. Inside the $\langle package code \rangle$ the $\langle default label \rangle$ can also be changed with `\SetDefaultHookLabel`. `\input` and other file input-related commands from the $\text{\LaTeX}$ kernel do not use `\PushDefaultHookLabel`, so code within files loaded by these commands does *not* get a dedicated $\langle label \rangle$! (that is, the $\langle default label \rangle$ is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (TikZ's `\usetikzlibrary`, for example) can use `\PushDefaultHookLabel` and `\PopDefaultHookLabel` to set dedicated labels and to emulate `\usepackage`-like hook behavior within those contexts.

The `top-level` label is treated differently, and is reserved to the user document, so it is not allowed to change the $\langle default label \rangle$ to `top-level`.

<code>\SetDefaultHookLabel</code>	<code>\SetDefaultHookLabel {⟨default label⟩}</code>
-----------------------------------	---

Similarly to `\PushDefaultHookLabel`, sets the current `⟨default label⟩` to be used in `⟨label⟩` arguments, or when replacing a leading “.”. The effect holds until the label is changed again or until the next `\PopDefaultHookLabel`. The difference between `\PushDefaultHookLabel` and `\SetDefaultHookLabel` is that the latter does not save the current `⟨default label⟩`.

This command is useful when a large package is composed of several smaller packages, but all should have the same `⟨label⟩`, so `\SetDefaultHookLabel` can be used at the beginning of each package file to set the correct label.

`\SetDefaultHookLabel` is not allowed in the main document, where the `⟨default label⟩` is `top-level` and there is no `\PopDefaultHookLabel` to end its effect. It is also not allowed to change the `⟨default label⟩` to `top-level`.

2.1.6 The `top-level` label

The `top-level` label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually `\AtBeginDocument`) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the `top-level` is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see `\NewReversedHook`), the `top-level` chunk is executed at the very beginning instead.

Rules regarding `top-level` have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The `top-level` label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precautions to determine whether some other package had also been loaded (before or after) and then to find some ways to alter its behavior accordingly. In addition it was often the user’s responsibility to load packages in the right order so that alterations made by packages were done in that same order; and in some cases even altering the loading order wouldn’t resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and to specify explicitly the order in which they should be processed.

The rules can be declared for hooks before the hook has been declared with `\NewHook` and they are allowed to refer to code labels that do not yet exist, e.g., because a package defining the code chunk with that label has not yet been loaded. When the hook code is finally sorted for fast execution, all rules that apply are acted on and the others are ignored.

This offers the flexibility needed to handle complicated relationships between code from different packages and to set this up beforehand in a way that is independent of whether or not the packages are actually loaded in a specific document. The downside

of this is that misspellings of hook names or code labels will not raise any error, instead the rule will simply never apply!

\DeclareHookRule `\DeclareHookRule {<hook>} {<label1>} {<relation>} {<label2>}`

Defines a relation between `<label1>` and `<label2>` for a given `<hook>`. If `<hook>` is `??` this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with `<label1>` and `<label2>`.

Currently, the supported relations are the following:

before or **<** Code for `<label1>` comes before code for `<label2>`.

after or **>** Code for `<label1>` comes after code for `<label2>`.

incompatible-warning Only code for either `<label1>` or `<label2>` can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

incompatible-error Like **incompatible-warning** but instead of a warning a L^AT_EX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

voids Code for `<label1>` overwrites code for `<label2>`. More precisely, code for `<label2>` is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

unrelated The order of code for `<label1>` and `<label2>` is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookRule` overwrites any previous declaration. In all cases rules specific to a given hook take precedence over default rules that use `??` as the `<hook>`.

If a default rule is applied, it is done before reversing the label order in a reversed hook, e.g., **before** in a default rule effectively becomes **after** in such a hook. In contrast, a rule for a specific hook is always applied to the state after any reversal (i.e., the state you see when using `\ShowHook` on that hook).

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

\ClearHookRule `\ClearHookRule {<hook>} {<label1>} {<label2>}`

Syntactic sugar for saying that `<label1>` and `<label2>` are unrelated for the given `<hook>`.

`\DeclareDefaultHookRule` `\DeclareDefaultHookRule {<label1>} {<relation>} {<label2>}`

This sets up a relation between `<label1>` and `<label2>` for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is `incompatible` or always needs a special ordering `before` or `after`. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., `after` becomes `before`) because those rules are applied first and then the order is reversed. The rationale is that in hook pairs (in which the ordering in one is reversed) default rules have to be reversed too in nearly all scenarios. If this is not the case, a default rule can't be used or has to be overwritten with an explicit `\DeclareHookRule` for that specific hook.

Declaring default rules is only supported in the document preamble.⁵

The `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;
- exist and be non-empty; and
- not exist (in which case emptiness doesn't apply);

Hooks are a bit more complicated: a hook may exist or not, and independently it may or may not be empty. This means that even a hook that doesn't exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package *A* defines hook `A/foo`, and package *B* adds some code to that hook. However, a document may load package *B* before package *A*, or may not load package *A* at all. In both cases some code is added to hook `A/foo` without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn't exist. Therefore, querying the existence of a hook doesn't imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn't physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

⁵Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

<code>\IfHookEmptyTF</code>	★	<code>\IfHookEmptyTF {<hook>} {<true code>} {<false code>}</code>
<code>\IfHookEmptyT</code>	★	
<code>\IfHookEmptyF</code>	★	Tests if the <code><hook></code> is empty (<i>i.e.</i> , no code was added to it using either <code>\AddToHook</code> or <code>\AddToHookNext</code>) or such code was removed again (via <code>\RemoveFromHook</code>), and branches to either <code><true code></code> or <code><false code></code> depending on the result.

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

<code>\ShowHook</code>	<code>\ShowHook {<hook>}</code>
<code>\LogHook</code>	<code>\LogHook {<hook>}</code>

Displays information about the `<hook>` such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\LogHook` prints the information to the `.log` file, and `\ShowHook` prints them to the terminal/command window and starts `TEX`'s prompt (only in `\errorstopmode`) to wait for user action.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

Suppose a hook `example-hook` whose output of `\ShowHook{example-hook}` is:

```

1  -> The hook 'example-hook':
2  > Code chunks:
3  >   foo -> [code from package 'foo']
4  >   bar -> [from package 'bar']
5  >   baz -> [package 'baz' is here]
6  > Document-level (top-level) code (executed last):
7  >   -> [code from 'top-level']
8  > Extra code for next invocation:
9  >   -> [one-time code]
10 > Rules:
11 >   foo|baz with relation >
12 >   baz|bar with default relation <
13 > Execution order (after applying rules):
14 >   baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

`<label> -> <code>`

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

```
Document-level (top-level) code (executed  $\langle first/last \rangle$ ):
->  $\langle top-level\ code \rangle$ 
```

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

```
->  $\langle next-code \rangle$ 
```

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{<label>}[example-hook]`.

Lines 11 and 12 show the rules declared that affect this hook in the format

```
 $\langle label-1 \rangle | \langle label-2 \rangle$  with  $\langle default? \rangle$  relation  $\langle relation \rangle$ 
```

which means that the $\langle relation \rangle$ applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that this rule applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$ in *all* hooks, (unless overridden by a non-default relation).

Finally, line 14 lists the labels in the hook after sorting; that is, in the order they will be executed when the hook is used.

2.1.10 Debugging hook code

```
\DebugHooksOn \DebugHooksOn ... \DebugHooksOff
\DebugHooksOff
```

Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use, but it can be helpful in case hooks do not work as expected. See also [2.1.9](#) for commands to inspect individual hooks.

2.2 L3 programming layer (expl3) interfaces

This is a quick summary of the L^AT_EX3 programming interfaces for use with packages written in `expl3`. In contrast to the L^AT_EX_{2 ϵ} interfaces they always use mandatory arguments only, e.g., you always have to specify the $\langle label \rangle$ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

```
\hook_new:n \hook_new:n {<hook>}
\hook_new_reversed:n \hook_new_reversed:n {<hook>}
\hook_new_pair:nn \hook_new_pair:nn {<hook-1>} {<hook-2>}
```

Creates a new $\langle hook \rangle$ with normal or reverse ordering of code chunks. `\hook_new_pair:nn` creates a pair of such hooks with $\{<hook-2>\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

<code>\hook_new_with_args:nn</code>	<code>\hook_new_with_args:nn {\hook} {\number}</code>
<code>\hook_new_reversed_with_args:nn</code>	<code>\hook_new_reversed_with_args:nn {\hook} {\number}</code>
<code>\hook_new_pair_with_args:nnn</code>	<code>\hook_new_pair_with_args:nnn {\hook-1} {\hook-2} {\number}</code>

Creates a new $\langle\textit{hook}\rangle$ with normal or reverse ordering of code chunks, that takes $\langle\textit{number}\rangle$ arguments from the input stream when it is used. `\hook_new_pair_with_args:nn` creates a pair of such hooks with $\{\langle\textit{hook-2}\rangle\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle\textit{hook}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_disable_generic:n</code>	<code>\hook_disable_generic:n {\hook}</code>
--------------------------------------	--

Marks $\{\langle\textit{hook}\rangle\}$ as disabled. Any further attempt to add code to it or declare it, will result in an error and any call to `\hook_use:n` will simply do nothing.

This declaration is intended for use with generic hooks that are known not to work (see `lthcmdhooks-doc`) if they receive code.

The $\langle\textit{hook}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_activate_generic:n</code>	<code>\hook_activate_generic:n {\hook}</code>
---------------------------------------	---

This is like `\hook_new:n` but it does nothing if the hook was previously declared with `\hook_new:n`. This declaration should be used only in special situations, e.g., when a command from another package needs to be altered and it is not clear whether a generic `cmd` hook (for that command) has been previously explicitly declared.

Normally `\hook_new:n` should be used instead of this.

<code>\hook_use:n</code>	<code>\hook_use:n {\hook}</code>
<code>\hook_use:nnw</code>	<code>\hook_use:nnw {\hook} {\number} {\arg_1} \dots {\arg_n}</code>

Executes the $\{\langle\textit{hook}\rangle\}$ code followed (if set up) by the code for next invocation only, then empties that next invocation code. `\hook_use:nnw` should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The $\langle\textit{number}\rangle$ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove $\langle\textit{number}\rangle$ items from the input.

The $\langle\textit{hook}\rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

<code>\hook_use_once:n</code>	<code>\hook_use_once:n {\hook}</code>
<code>\hook_use_once:nnw</code>	<code>\hook_use_once:nnw {\hook} {\number} {\arg_1} \dots {\arg_n}</code>

Changes the $\{\langle\textit{hook}\rangle\}$ status so that from now on any addition to the hook code is executed immediately. Then execute any $\{\langle\textit{hook}\rangle\}$ code already set up. `\hook_use_once:nnw` should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The $\langle\textit{number}\rangle$ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove $\langle\textit{number}\rangle$ items from the input.

The $\langle\textit{hook}\rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

<code>\hook_gput_code:nnn</code>	<code>\hook_gput_code:nnn</code>	<code>{\hook} {\label} {\code}</code>
<code>\hook_gput_code_with_args:nnn</code>	<code>\hook_gput_code_with_args:nnn</code>	<code>{\hook} {\label} {\code}</code>

Adds a chunk of `\code` to the `\hook` labeled `\label`. If the label already exists the `\code` is appended to the already existing code.

If `\hook_gput_code_with_args:nnn` is used, the `\code` can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

If code is added to an external `\hook` (of the kernel or another package) then the convention is to use the package name as the `\label` not some internal module name or some other arbitrary string.

The `\hook` and `\label` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_gput_next_code:nn</code>	<code>\hook_gput_next_code:nn</code>	<code>{\hook} {\code}</code>
<code>\hook_gput_next_code_with_args:nn</code>	<code>\hook_gput_next_code_with_args:nn</code>	<code>{\hook} {\code}</code>

Adds a chunk of `\code` for use only in the next invocation of the `\hook`. Once used it is gone.

If `\hook_gput_next_code_with_args:nn` is used, the `\code` can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

This is simpler than `\hook_gput_code:nnn`, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed. Thus if one needs to undo what the standard does one has to do that as part of `\code`.

The `\hook` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_gclear_next_code:n</code>	<code>\hook_gclear_next_code:n</code>	<code>{\hook}</code>
---------------------------------------	---------------------------------------	----------------------

Undo any earlier `\hook_gput_next_code:nn`.

<code>\hook_gremove_code:nn</code>	<code>\hook_gremove_code:nn</code>	<code>{\hook} {\label}</code>
------------------------------------	------------------------------------	-------------------------------

Removes any code for `\hook` labeled `\label`.

If there is no code under the `\label` in the `\hook`, or if the `\hook` does not exist, a warning is issued when you attempt to use `\hook_gremove_code:nn`, and the command is ignored.

If the second argument is `*`, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about, so think twice before using that!

The `\hook` and `\label` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<hr/> <hr/>	<code>\hook_gset_rule:nnnn</code>	<code>\hook_gset_rule:nnnn {<hook>} {<label1>} {<relation>} {<label2>}</code>
		Relate <code><label1></code> with <code><label2></code> when used in <code><hook></code> . See <code>\DeclareHookRule</code> for the allowed <code><relation></code> s. If <code><hook></code> is ?? a default rule is specified. The <code><hook></code> and <code><label></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The dot-syntax is parsed in both <code><label></code> arguments, but it usually makes sense to be used in only one of them.
<hr/> <hr/>	<code>\hook_if_empty_p:n *</code> <code>\hook_if_empty:nTF *</code>	<code>\hook_if_empty:nTF {<hook>} {<true code>} {<false code>}</code> Tests if the <code><hook></code> is empty (<i>i.e.</i> , no code was added to it using either <code>\AddToHook</code> or <code>\AddToHookNext</code>), and branches to either <code><true code></code> or <code><false code></code> depending on the result. The <code><hook></code> <i>cannot</i> be specified using the dot-syntax. A leading . is treated literally.
<hr/> <hr/>	<code>\hook_show:n</code> <code>\hook_log:n</code>	<code>\hook_show:n {<hook>}</code> <code>\hook_log:n {<hook>}</code> Displays information about the <code><hook></code> such as <ul style="list-style-type: none"> • the code chunks (and their labels) added to it, • any rules set up to order them, • the computed order in which the chunks are executed, • any code executed on the next invocation only. <code>\hook_log:n</code> prints the information to the .log file, and <code>\hook_show:n</code> prints them to the terminal/command window and starts TeX's prompt (only if <code>\errorstopmode</code>) to wait for user action. The <code><hook></code> can be specified using the dot-syntax to denote the current package name. See section 2.1.5.
<hr/> <hr/>	<code>\hook_debug_on:</code> <code>\hook_debug_off:</code>	<code>\hook_debug_on:</code> Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a `<hook>` under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}[packageA]{\typeout{A}}
\AddToHook{myhook}[packageB]{\typeout{B}}
\AddToHook{myhook}[packageC]{\typeout{C}}
```

then executing the hook with `\UseHook` will produce the typeout A B C in that order. In other words, the execution order is computed to be `packageA`, `packageB`, `packageC` which you can verify with `\ShowHook{myhook}`:

```

-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order:
>   packageA, packageB, packageC.

```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for `packageA`, e.g.,

```

\RemoveFromHook{myhook}[packageA]
\AddToHook{myhook}[packageA]{\typeout{A alt}}

```

then your order becomes `packageB`, `packageC`, `packageA` because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```

\DeclareHookRule{myhook}{packageA}{before}{packageB}

```

instead of the previous lines we get

```

-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   packageB|packageA with relation >
> Execution order (after applying rules):
>   packageA, packageC, packageB.

```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```

\DeclareHookRule{myhook}{packageB}{before}{packageC}

```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of “reversed” hooks

You may have wondered why you can declare a “reversed” hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example⁶, suppose there is a package adding the following:

```
\AddToHook{env/quote/before}[package-1]{\begin{itshape}}
\AddToHook{env/quote/after} [package-1]{\end{itshape}}
```

As a result, all quotes will be in italics. Now suppose further that another `package-too` makes the quotes also in blue and therefore adds:

```
\usepackage{color}
\AddToHook{env/quote/before}[package-too]{\begin{color}{blue}}
\AddToHook{env/quote/after} [package-too]{\end{color}}
```

Now if the `env/quote/after` hook would be a normal hook we would get the same execution order in both hooks, namely:

```
package-1, package-too
```

(or vice versa) and as a result, would get:

```
\begin{itshape}\begin{color}{blue} ...
\end{itshape}\end{color}
```

and an error message saying that `\begin{color}` was ended by `\end{itshape}`. With `env/quote/after` declared as a reversed hook the execution order is reversed and so all environments are closed in the correct sequence and `\ShowHook` would give us the following output:

```
-> The hook 'env/quote/after':
> Code chunks:
>   package-1 -> \end {itshape}
>   package-too -> \end {color}
> Document-level (top-level) code (executed first):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order (after reversal):
>   package-too, package-1.
```

If there is a matching default rule (done with `\DeclareDefaultHookRule` or with `??` for the hook name) then this default rule is applied before the reversal so that the order in the reversed hook mirrors the one in the normal hook. However, all rules specific to a hook happen always after the reversal of the execution order, so if you alter the order you will probably have to alter it in both hooks, not just in one, but that depends on the use case.

⁶There are simpler ways to achieve the same effect.

2.5 Difference between “normal” and “one-time” hooks

When executing a hook a developer has the choice of using either `\UseHook` or `\UseOneTimeHook` (or their expl3 equivalents `\hook_use:n` and `\hook_use_once:n`). This choice affects how `\AddToHook` is handled after the hook has been executed for the first time.

With normal hooks adding code via `\AddToHook` means that the code chunk is added to the hook data structure and then used each time `\UseHook` is called.

With one-time hooks it this is handled slightly differently: After `\UseOneTimeHook` has been called, any further attempts to add code to the hook via `\AddToHook` will simply execute the `<code>` immediately.

This has some consequences one needs to be aware of:

- If `<code>` is added to a normal hook after the hook was executed and it is never executed again for one or the other reason, then this new `<code>` will never be executed.
- In contrast if that happens with a one-time hook the `<code>` is executed immediately.

In particular this means that construct such as

```
\AddToHook{myhook}  
{ <code-1> \AddToHook{myhook}{<code-2>} <code-3> }
```

works for one-time hooks⁷ (all three code chunks are executed one after another), but it makes little sense with a normal hook, because with a normal hook the first time `\UseHook{myhook}` is executed it would

- execute `<code-1>`,
- then execute `\AddToHook{myhook}{code-2}` which adds the code chunk `<code-2>` to the hook for use on the next invocation,
- and finally execute `<code-3>`.

The second time `\UseHook` is called it would execute the above and in addition `<code-2>` as that was added as a code chunk to the hook in the meantime. So each time the hook is used another copy of `<code-2>` is added and so that code chunk is executed `<# of invocations> - 1` times.

2.6 Generic hooks provided by packages

The hook management system also implements a category of hooks that are called “Generic Hooks”. Normally a hook has to be explicitly declared before it can be used in code. This ensures that different packages are not using the same hook name for unrelated purposes—something that would result in absolute chaos. However, there are a number of “standard” hooks where it is unreasonable to declare them beforehand, e.g., each and every command has (in theory) an associated **before** and **after** hook. In such cases, i.e., for command, environment or file hooks, they can be used simply by adding code to them with `\AddToHook`. For more specialized generic hooks, e.g., those provided

⁷This is sometimes used with `\AtBeginDocument` which is why it is supported.

by `babel`, you have to additionally enable them with `\ActivateGenericHook` as explained below.

The generic hooks provided by $\text{\LaTeX}$ are those for `cmd`, `env`, `file`, `include`, `package`, and `class`, and all these are available out of the box: you only have to use `\AddToHook` to add code to them, but you don't have to add `\UseHook` or `\UseOneTimeHook` to your code, because this is already done for you (or, in the case of `cmd` hooks, the command's code is patched at `\begin{document}`, if necessary).

However, if you want to provide further generic hooks in your own code, the situation is slightly different. To do this you should use `\UseHook` or `\UseOneTimeHook`, but *without declaring the hook* with `\NewHook`. As mentioned earlier, a call to `\UseHook` with an undeclared hook name does nothing. So as an additional setup step, you need to explicitly activate your generic hook. Note that a generic hook produced in this way is always a normal hook.

For a truly generic hook, with a variable part in the hook name, such upfront activation would be difficult or impossible, because you typically do not know what kind of variable parts may come up in real documents.

For example, `babel` provides hooks such as `babel/<language>/afterextras`. However, language support in `babel` is often done through external language packages. Thus doing the activation for all languages inside the core `babel` code is not a viable approach. Instead it needs to be done by each language package (or by the user who wants to use a particular hook).

Because the hooks are not declared with `\NewHook` their names should be carefully chosen to ensure that they are (likely to be) unique. Best practice is to include the package or command name, as was done in the `babel` example above.

Generic hooks defined in this way are always normal hooks (i.e., you can't implement reversed hooks this way). This is a deliberate limitation, because it speeds up the processing considerably.

2.7 Hooks with arguments

Sometimes it is necessary to pass contextual information to a hook, and, for one reason or another, it is not feasible to store such information in macros. To serve this purpose, hooks can be declared with arguments, so that the programmer can pass along the data necessary for the code in the hook to function properly.

A hook with arguments works mostly like a regular hook, and most commands that work for regular hooks, also work for hooks that take arguments. The differences are when the hook is declared (`\NewHookWithArguments` is used instead of `\NewHook`), then code can be added with both `\AddToHook` and `\AddToHookWithArguments`, and when the hook is used (`\UseHookWithArguments` instead of `\UseHook`).

A hook with arguments must be declared as such (before it is first used, as all regular hooks) using `\NewHookWithArguments{<hook>}{<number>}`. All code added to that hook can then use `#1` to access the first argument, `#2` to access the second, and so forth up to the number of arguments declared. However, it is still possible to add code with references to the arguments of a hook that was not yet declared (we will discuss that later). At their core, hooks are macros, so $\text{\TeX}$'s limit of 9 arguments applies, and a low-level $\text{\TeX}$ error is raised if you try to reference an argument number that doesn't exist.

To use a hook with arguments, just write `\UseHookWithArguments{<hook>}{<number>}` followed by a braced list of the arguments. For example, if the hook `test` takes three arguments, write:

```
\UseHookWithArguments{test}{3}{arg-1}{arg-2}{arg-3}
```

then, in the `<code>` of the hook, all instances of `#1` will be replaced by `arg-1`, `#2` by `arg-2` and so on. If, at the point of usage, the programmer provides more arguments than the hook is declared to take, the excess arguments are simply ignored by the hook. Behavior is unpredictable⁸ if too few arguments are provided. If the hook isn't declared, `<number>` arguments are removed from the input stream.

Adding code to a hook with arguments can be done with `\AddToHookWithArguments` as well as with the regular `\AddToHook`, to achieve different outcomes. The main difference when it comes to adding code to a hook, in this case, is firstly the possibility of accessing a hook's arguments, of course, and second, how parameter tokens (`#6`) are treated.

Using `\AddToHook` in a hook that takes arguments will work as it does for all other hooks. This allows a package developer to add arguments to a hook that otherwise had none without having to worry about compatibility. This means that, for example:

```
\AddToHook{test}{\def\foo#1{Hello, #1!}}
```

will define the same macro `\foo` regardless if the hook `test` takes arguments or not.

Using `\AddToHookWithArguments` allows the `<code>` added to access the arguments of the hook with `#1`, `#2`, and so forth, up to the number of the arguments declared in the hook. This means that if one wants to add a `#6` to the `<code>` that token must be doubled in the input. The same definition from above, using `\AddToHookWithArguments`, needs to be rewritten:

```
\AddToHookWithArguments{test}{\def\foo##1{Hello, ##1!}}
```

Extending the above example to use the hook arguments, we could rewrite something like (now from declaration to usage, to get the whole picture):

```
\NewHookWithArguments{test}{1}
\AddToHookWithArguments{test}{%
  \typeout{Defining foo with "#1"}
  \def\foo##1{Hello, ##1! Some text after: #1}%
}
\UseHook{test}{Howdy!}
\ShowCommand\foo
```

Running the code above prints in the terminal:

```
Defining foo with "Howdy!"
> \foo=macro:
#1->Hello, #1! Some text after: Howdy!.
```

⁸The hook *will* take the declared number of arguments, and what will happen depends on what was grabbed, and what the hook code does with its arguments.

Note how `##1` in the call to `\AddToHookWithArguments` became `#1`, and the `#1` was replaced by the argument passed to the hook. Should the hook be used again, with a different argument, the definition would naturally change.

It is possible to add code referencing a hook’s arguments before such hook is declared and the number of hooks is fixed. However, if some code is added to the hook, that references more arguments than will be declared for the hook, there will be a low-level `TeX` error about an “Illegal parameter number” at the time the hook is declared, which will be hard to track down because at that point `TeX` can’t know whence the offending code came from. Thus it is important that package writers explicitly document how many arguments (if any) each hook can take, so users of those packages know how many arguments can be referenced, and equally important, what each argument means.

2.8 Private `LaTeX` kernel hooks

There are a few places where it is absolutely essential for `LaTeX` to function correctly that code is executed in a precisely defined order. Even that could have been implemented with the hook management (by adding various rules to ensure the appropriate ordering with respect to other code added by packages). However, this makes every document unnecessary slow, because there has to be sorting even though the result is predetermined. Furthermore it forces package writers to unnecessarily add such rules if they add further code to the hook (or break `LaTeX`).

For that reason such code is not using the hook management, but instead private kernel commands directly before or after a public hook with the following naming convention: `\@kernel@before@hook` or `\@kernel@after@hook`. For example, in `\enddocument` you find

```
\UseHook{enddocument}%
\@kernel@after@enddocument
```

which means first the user/package-accessible `enddocument` hook is executed and then the internal kernel hook. As their name indicates these kernel commands should not be altered by third-party packages, so please refrain from that in the interest of stability and instead use the public hook next to it.⁹

2.9 Legacy `LaTeX 2ε` interfaces

`LaTeX 2ε` offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management, several additional hooks have been added to `LaTeX` and more will follow. See the next section for what is already available.

⁹As with everything in `TeX` there is not enforcement of this rule, and by looking at the code it is easy to find out how the kernel adds to them. The main reason of this section is therefore to say “please don’t do that, this is unconfigurable code!”

<code>\AtBeginDocument</code>	<code>\AtBeginDocument [<i><label></i>] {<i><code></i>}</code>
-------------------------------	--

If used without the optional argument *<label>*, it works essentially like before, i.e., it is adding *<code>* to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add code to the hook using `\AddToHook` or `\AtBeginDocument` using a different label and explicitly order the code chunks as necessary, e.g., run some code before or after another package's code. When using the optional argument the call is equivalent to running `\AddToHook {begindocument} [<label>] {<code>}`.

`\AtBeginDocument` is a wrapper around the `begindocument` hook (see section 3.2), which is a one-time hook. As such, after the `begindocument` hook is executed at `\begin{document}` any attempt to add *<code>* to this hook with `\AtBeginDocument` or with `\AddToHook` will cause that *<code>* to execute immediately instead. See section 2.5 for more on one-time hooks.

For important packages with known order requirement we may over time add rules to the kernel (or to those packages) so that they work regardless of the loading-order in the document.

<code>\AtEndDocument</code>	<code>\AtEndDocument [<i><label></i>] {<i><code></i>}</code>
-----------------------------	--

Like `\AtBeginDocument` but for the `enddocument` hook.

The few hooks that existed previously in L^AT_EX 2_ε used internally commands such as `@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of L^AT_EX we will turn this legacy support off, as it does unnecessary slow down the processing.

3 L^AT_EX 2_ε commands and environments augmented by hooks

In this section we describe the standard hooks that are now offered by L^AT_EX, or give pointers to other documents in which they are described. This section will grow over time (and perhaps eventually move to `usrguide3`).

3.1 Generic hooks

As stated earlier, with the exception of generic hooks, all hooks must be declared with `\NewHook` before they can be used. All generic hooks have names of the form “*<type>/<name>/<position>*”, where *<type>* is from the predefined list shown below, and *<name>* is the variable part whose meaning will depend on the *<type>*. The last component, *<position>*, has more complex possibilities: it can always be `before` or `after`; for `env` hooks, it can also be `begin` or `end`; and for `include` hooks it can also be `end`. Each specific hook is documented below, or in `lthooks-doc.pdf` or `ltfilehook-doc.pdf`.

The generic hooks provided by L^AT_EX belong to one of the six types:

- env** Hooks executed before and after environments – $\langle name \rangle$ is the name of the environment, and available values for $\langle position \rangle$ are **before**, **begin**, **end**, and **after**;
- cmd** Hooks added to and executed before and after commands – $\langle name \rangle$ is the name of the command, and available values for $\langle position \rangle$ are **before** and **after**;
- file** Hooks executed before and after reading a file – $\langle name \rangle$ is the name of the file (with extension), and available values for $\langle position \rangle$ are **before** and **after**;
- package** Hooks executed before and after loading packages – $\langle name \rangle$ is the name of the package, and available values for $\langle position \rangle$ are **before** and **after**;
- class** Hooks executed before and after loading classes – $\langle name \rangle$ is the name of the class, and available values for $\langle position \rangle$ are **before** and **after**;
- include** Hooks executed before and after `\included` files – $\langle name \rangle$ is the name of the included file (without the `.tex` extension), and available values for $\langle position \rangle$ are **before**, **end**, and **after**.

Each of the hooks above are detailed in the following sections and in linked documentation.

3.1.1 Generic hooks for all environments

Every environment $\langle env \rangle$ has now four associated hooks coming with it:

- env/ $\langle env \rangle$ /before** This hook is executed as part of `\begin` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.
- env/ $\langle env \rangle$ /begin** This hook is executed as part of `\begin` directly in front of the code specific to the environment start (e.g., the third argument of `\NewDocumentEnvironment` and the second argument of `\newenvironment`). Its scope is the environment body.
- env/ $\langle env \rangle$ /end** This hook is executed as part of `\end` directly in front of the code specific to the end of the environment (e.g., the forth argument of `\NewDocumentEnvironment` and the third argument of `\newenvironment`).
- env/ $\langle env \rangle$ /after** This hook is executed as part of `\end` after the code specific to the environment end and after the environment group has ended. Its scope is therefore not restricted by the environment.

The hook is implemented as a reversed hook so if two packages add code to **env/ $\langle env \rangle$ /before** and to **env/ $\langle env \rangle$ /after** they can add surrounding environments and the order of closing them happens in the right sequence.

Given that these generic hook names involve `/` as part of their name they would not work if one tries to define an environment using a name that involves a `/`.¹⁰

Generic environment hooks are never one-time hooks even with environments that are supposed to appear only once in a document.¹¹ In contrast to other hooks there is also no need to declare them using `\NewHook`.

¹⁰Officially, L^AT_EX names for environments should only consist of a sequence of letters, numbers, and the character `*`, i.e., this is not a new restriction.

¹¹Thus if one adds code to such hooks after the environment has been processed, it will only be executed if the environment appears again and if that doesn't happen the code will never get executed.

The hooks are only executed if `\begin{env}` and `\end{env}` is used. If the environment code is executed via low-level calls to `\env` and `\endenv` (e.g., to avoid the environment grouping) they are not available. If you want them available in code using this method, you would need to add them yourself, i.e., write something like

```
\UseHook{env/quote/before}\quote
...
\endquote\UseHook{env/quote/after}
```

to add the outer hooks, etc.

Largely for compatibility with existing packages, the following four commands are also available to set the environment hooks; but for new packages we recommend directly using the hook names and `\AddToHook`.

<code>\BeforeBeginEnvironment</code>	<code>\BeforeBeginEnvironment</code> [<i>label</i>] { <i>env</i> } { <i>code</i> }
--------------------------------------	--

This declaration adds to the `env/⟨env⟩/before` hook using the *label*. If *label* is not given, the *default label* is used (see section 2.1.5).

<code>\AtBeginEnvironment</code>	<code>\AtBeginEnvironment</code> [<i>label</i>] { <i>env</i> } { <i>code</i> }
----------------------------------	--

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/begin` hook.

<code>\AtEndEnvironment</code>	<code>\AtEndEnvironment</code> [<i>label</i>] { <i>env</i> } { <i>code</i> }
--------------------------------	--

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/end` hook.

<code>\AfterEndEnvironment</code>	<code>\AfterEndEnvironment</code> [<i>label</i>] { <i>env</i> } { <i>code</i> }
-----------------------------------	---

This is like `\BeforeBeginEnvironment` but it adds to the `env/⟨env⟩/after` hook.

3.1.2 Generic hooks for commands

Similar to environments there are now (at least in theory) two generic hooks available for any $\text{\LaTeX}$ command. These are

cmd/⟨name⟩/before This hook is executed at the very start of the command execution.

cmd/⟨name⟩/after This hook is executed at the very end of the command body. It is implemented as a reversed hook.

In practice there are restrictions and especially the **after** hook works only with a subset of commands. Details about these restrictions are documented in `ltxcmdhooks-doc.pdf` or with code in `ltxcmdhooks-code.pdf`.

3.1.3 Generic hooks provided by file loading operations

There are several hooks added to $\text{\LaTeX}$'s process of loading file via its high-level interfaces such as `\input`, `\include`, `\usepackage`, `\RequirePackage`, etc. These are documented in `ltfilehook-doc.pdf` or with code in `ltfilehook-code.pdf`.

3.2 Hooks provided by `\begin{document}`

Until 2020 `\begin{document}` offered exactly one hook that one could add to using `\AtBeginDocument`. Experiences over the years have shown that this single hook in one place was not enough and as part of adding the general hook management system a number of additional hooks have been added at this point. The places for these hooks have been chosen to provide the same support as offered by external packages, such as `etoolbox` and others that augmented `\document` to gain better control.

Supported are now the following hooks (all of them one-time hooks):

`begindocument/before` This hook is executed at the very start of `\document`, one can think of it as a hook for code at the end of the preamble section and this is how it is used by `etoolbox`'s `\AtEndPreamble`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument` This hook is added to by using `\AddToHook{begindocument}` or by using `\AtBeginDocument` and it is executed after the `.aux` file has been read and most initialization are done, so they can be altered and inspected by the hook code. It is followed by a small number of further initializations that shouldn't be altered and are therefore coming later.

The hook should not be used to add material for typesetting as we are still in $\text{\LaTeX}$'s initialization phase and not in the document body. If such material needs to be added to the document body use the next hook instead.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument/end` This hook is executed at the end of the `\document` code in other words at the beginning of the document body. The only command that follows it is `\ignorespaces`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

The generic hooks executed by `\begin` also exist, i.e., `env/document/before` and `env/document/begin`, but with this special environment it is better use the dedicated one-time hooks above.

3.3 Hooks provided by `\end{document}`

$\text{\LaTeX 2}_{\epsilon}$ has always provided `\AtEndDocument` to add code to the `\end{document}`, just in front of the code that is normally executed there. While this was a big improvement over the situation in $\text{\LaTeX 2.09}$, it was not flexible enough for a number of use cases and so packages, such as `etoolbox`, `atveryend` and others patched `\enddocument` to add additional points where code could be hooked into.

Patching using packages is always problematical as leads to conflicts (code availability, ordering of patches, incompatible patches, etc.). For this reason a number of additional hooks have been added to the `\enddocument` code to allow packages to add code in various places in a controlled way without the need for overwriting or patching the core code.

Supported are now the following hooks (all of them one-time hooks):

enddocument The hook associated with `\AtEndDocument`. It is immediately called at the beginning of `\enddocument`.

When this hook is executed there may be still unprocessed material (e.g., floats on the deferlist) and the hook may add further material to be typeset. After it, `\clearpage` is called to ensure that all such material gets typeset. If there is nothing waiting the `\clearpage` has no effect.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afterlastpage As the name indicates this hook should not receive code that generates material for further pages. It is the right place to do some final housekeeping and possibly write out some information to the `.aux` file (which is still open at this point to receive data). Just before this hook `\write` is redefined and is now always `\immediate\write`. This means that writing e.g. a label or something to the `.toc` works despite the fact that no more pages are output. It is also the correct place to set up any testing code to be run when the `.aux` file is re-read in the next step.

After this hook has been executed the `.aux` file is closed for writing and then read back in to do some tests (e.g., looking for missing references or duplicated labels, etc.).

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afteraux At this point, the `.aux` file has been reprocessed and so this is a possible place for final checks and display of information to the user. However, for the latter you might prefer the next hook, so that your information is displayed after the (possibly longish) list of files if that got requested via `\listfiles`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/info This hook is meant to receive code that write final information messages to the terminal. It follows immediately after the previous hook (so both could have been combined, but then packages adding further code would always need to also supply an explicit rule to specify where it should go).

This hook already contains some code added by the kernel (under the labels `kernel/filelist` and `kernel/warnings`), namely the list of files when `\listfiles` has been used and the warnings for duplicate labels, missing references, font substitutions etc.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/end Finally, this hook is executed just in front of the final call to `\@@end`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).is it even possible to add code after this one?

There is also the hook `shipout/lastpage`. This hook is executed as part of the last `\shipout` in the document to allow package to add final `\special`'s to that page. Where this hook is executed in relation to those from the above list can vary from document to

document. Furthermore to determine correctly which of the `\shipouts` is the last one, $\text{\LaTeX}$ needs to be run several times, so initially it might get executed on the wrong page. See section 3.4 for where to find the details.

It is also possible to use the generic `env/document/end` hook which is executed by `\end`, i.e., just in front of the first hook above. Note however that the other generic `\end` environment hook, i.e., `env/document/after` will never get executed, because by that time $\text{\LaTeX}$ has finished the document processing.

3.4 Hooks provided by `\shipout` operations

There are several hooks and mechanisms added to $\text{\LaTeX}$'s process of generating pages. These are documented in `ltshipout-doc.pdf` or with code in `ltshipout-code.pdf`.

3.5 Hooks provided for paragraphs

The paragraph processing has been augmented to include a number of internal and public hooks. These are documented in `ltpara-doc.pdf` or with code in `ltpara-code.pdf`.

3.6 Hooks provided in NFSS commands

In languages that need to support for more than one script in parallel (and thus several sets of fonts, e.g., supporting both Latin and Japanese fonts), NFSS font commands such as `\sffamily` need to switch both the Latin family to “Sans Serif” and in addition alter a second set of fonts.

To support this, several NFSS commands have hooks to which such support can be added.

rmfamily After `\rmfamily` has done its initial checks and prepared a font series update, this hook is executed before `\selectfont`.

sffamily This is like the `rmfamily` hook, but for the `\sffamily` command.

ttfamily This is like the `rmfamily` hook, but for the `\ttfamily` command.

normalfont The `\normalfont` command resets the font encoding, family, series and shape to their document defaults. It then executes this hook and finally calls `\selectfont`.

expand@font@defaults The internal `\expand@font@defaults` command expands and saves the current defaults for the metafont families (`rm/sf/tt`) and the metaseries (`bf/md`). If the NFSS machinery has been augmented, e.g., for Chinese or Japanese fonts, then further defaults may need to be set at this point. This can be done in this hook which is executed at the end of this macro.

bfseries/defaults, bfseries If the `\bfdefault` was explicitly changed by the user, its new value is used to set the `bf` series defaults for the metafont families (`rm/sf/tt`) when `\bfseries` is called. The `bfseries/defaults` hook allows further adjustments to be made in this case. This hook is only executed if such a change is detected. In contrast, the `bfseries` hook is always executed just before `\selectfont` is called to change to the new series.

mdseries/defaults, mdseries These two hooks are like the previous ones but they are in the `\mdseries` command.

selectfont This hook is executed inside `\selectfont`, after the current values for *encoding*, *family*, *series*, *shape*, and *size* are evaluated and the new font is selected (and if necessary loaded). After the hook has executed, NFSS will still do any updates necessary for a new *size* (such as changing the size of `\strut`) and any updates necessary to a change in *encoding*.

This hook is intended for use cases where, in parallel to a change in the main font, some other fonts need to be altered (e.g., in CJK processing where you may need to deal with several different alphabets).

3.7 Hook provided by the mark mechanism

See `ltmarks-doc.pdf` for details.

insertmark This hook allows for a special setup while `\InsertMark` inserts a mark. It is executed in group so local changes only apply to the mark being inserted.

4 The Implementation

```

1 <@@=hook>
2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2020/10/01}{lthooks}
5 <latexrelease> {The~hook~management~system}
```

4.1 Debugging

```

\g__hook_debug_bool Holds the current debugging state.
6 \bool_new:N \g__hook_debug_bool
(End of definition for \g__hook_debug_bool.)

\hook_debug_on: Turns debugging on and off by redefining \__hook_debug:n.
\hook_debug_off:
\__hook_debug:n
\__hook_debug_gset:
7 \cs_new_eq:NN \__hook_debug:n \use_none:n
8 \cs_new_protected:Npn \hook_debug_on:
9 {
10   \bool_gset_true:N \g__hook_debug_bool
11   \__hook_debug_gset:
12 }
13 \cs_new_protected:Npn \hook_debug_off:
14 {
15   \bool_gset_false:N \g__hook_debug_bool
16   \__hook_debug_gset:
17 }
18 \cs_new_protected:Npn \__hook_debug_gset:
19 {
20   \cs_gset_protected:Npe \__hook_debug:n ##1
21   { \bool_if:NT \g__hook_debug_bool {##1} }
22 }
```

(End of definition for `\hook_debug_on:` and others. These functions are documented on page 232.)

4.2 Borrowing from internals of other kernel modules

`\__hook_str_compare:nn` Private copy of `\__str_if_eq:nn`
`23 \cs_new_eq:NN \__hook_str_compare:nn \__str_if_eq:nn`
(End of definition for __hook_str_compare:nn.)

4.3 Declarations

`\l__hook_tmpa_bool` Scratch boolean used throughout the package.
`24 \bool_new:N \l__hook_tmpa_bool`
(End of definition for \l__hook_tmpa_bool.)

`\l__hook_return_tl` Scratch variables used throughout the package.
`\l__hook_tmpa_tl` `25 \tl_new:N \l__hook_return_tl`
`\l__hook_tmpb_tl` `26 \tl_new:N \l__hook_tmpa_tl`
`27 \tl_new:N \l__hook_tmpb_tl`
(End of definition for \l__hook_return_tl, \l__hook_tmpa_tl, and \l__hook_tmpb_tl.)

`\g__hook_all_seq` In a few places we need a list of all hook names ever defined so we keep track of them in this sequence.
`28 \seq_new:N \g__hook_all_seq`
(End of definition for \g__hook_all_seq.)

`\l__hook_cur_hook_tl` Stores the name of the hook currently being sorted.
`29 \tl_new:N \l__hook_cur_hook_tl`
(End of definition for \l__hook_cur_hook_tl.)

`\l__hook_work_prop` A property list holding a copy of the `\g__hook_⟨hook⟩_code_prop` of the hook being sorted to work on, so that changes don't act destructively on the hook data structure.
`30 \prop_new:N \l__hook_work_prop`
(End of definition for \l__hook_work_prop.)

`\g__hook_used_prop` All hooks that receive code (for use in debugging display).
`31 \prop_new:N \g__hook_used_prop`
(End of definition for \g__hook_used_prop.)

`\g__hook_hook_curr_name_tl` Default label used for hook commands, and a stack to keep track of packages within packages.
`\g__hook_name_stack_seq` `32 \tl_new:N \g__hook_hook_curr_name_tl`
`33 \seq_new:N \g__hook_name_stack_seq`
(End of definition for \g__hook_hook_curr_name_tl and \g__hook_name_stack_seq.)

`\__hook_tmp:w` Temporary macro for generic usage.
`34 \cs_new_eq:NN \__hook_tmp:w ?`
(End of definition for __hook_tmp:w.)

`\c__hook_empty_tl` An empty token list, and one containing nine parameters.

`\c__hook_nine_parameters_tl`

```

35 \tl_const:Nn \c__hook_empty_tl { }
36 \tl_const:Nn \c__hook_nine_parameters_tl { #1#2#3#4#5#6#7#8#9 }

```

(End of definition for `\c__hook_empty_tl` and `\c__hook_nine_parameters_tl`.)

`\str_count:e` Some variants of `expl3` functions.

FMi: should probably be moved to `expl3`

```

37 \cs_generate_variant:Nn \str_count:n { e }

```

(End of definition for `\str_count:e`.)

`\s__hook_mark` Scan mark used for delimited arguments.

```

38 \scan_new:N \s__hook_mark

```

(End of definition for `\s__hook_mark`.)

`\_hook_use_none_delimit_by_s_mark:w` Removes tokens until the next `\s__hook_mark`.

`\_hook_use_i_delimit_by_s_mark:nw`

```

39 \cs_new:Npn \_hook_use_none_delimit_by_s_mark:w #1 \s__hook_mark { }
40 \cs_new:Npn \_hook_use_i_delimit_by_s_mark:nw #1 #2 \s__hook_mark {#1}

```

(End of definition for `\_hook_use_none_delimit_by_s_mark:w` and `\_hook_use_i_delimit_by_s_mark:nw`.)

`\_hook_tl_set:cn` Private copies of a few `expl3` functions. `l3debug` will only add debugging to the public names, not to these copies, so we don't have to use `\debug_suspend:` and `\debug_resume:` everywhere.

Functions like `\_hook_tl_set:Nn` have to be redefined, rather than copied because in `expl3` they use `\_kernel_tl_(g)set:Nx`, which is also patched by `l3debug`.

```

41 \cs_new_protected:Npn \_hook_tl_set:cn #1#2
42   { \cs_set_nopar:cpe {#1} { \_kernel_exp_not:w {#2} } }

```

(End of definition for `\_hook_tl_set:cn`.)

`\_hook_tl_gset:Nn` Same as above.

`\_hook_tl_gset:Ne`

`\_hook_tl_gset:cn`

`\_hook_tl_gset:co`

`\_hook_tl_gset:ce`

```

43 \cs_new_protected:Npn \_hook_tl_gset:Nn #1#2
44   { \cs_gset_nopar:Npe #1 { \_kernel_exp_not:w {#2} } }
45 \cs_new_protected:Npn \_hook_tl_gset:Ne #1#2
46   { \cs_gset_nopar:Npe #1 {#2} }
47 \cs_generate_variant:Nn \_hook_tl_gset:Nn { c, co }
48 \cs_generate_variant:Nn \_hook_tl_gset:Ne { c }

```

(End of definition for `\_hook_tl_gset:Nn`.)

`\_hook_tl_gput_right:Nn` Same as above.

`\_hook_tl_gput_right:Ne`

`\_hook_tl_gput_right:cn`

```

49 \cs_new_protected:Npn \_hook_tl_gput_right:Nn #1#2
50   { \_hook_tl_gset:Ne #1 { \_kernel_exp_not:w \exp_after:wN { #1 #2 } } }
51 \cs_generate_variant:Nn \_hook_tl_gput_right:Nn { Ne, cn }

```

(End of definition for `\_hook_tl_gput_right:Nn`.)

`\__hook_tl_gput_left:Nn` Same as above.

```

52 \cs_new_protected:Npn \__hook_tl_gput_left:Nn #1#2
53 {
54   \__hook_tl_gset:Ne #1
55   { \__kernel_exp_not:w {#2} \__kernel_exp_not:w \exp_after:wN {#1} }
56 }

```

(End of definition for `\__hook_tl_gput_left:Nn`.)

`\__hook_tl_gset_eq:NN` Same as above.

```

57 \cs_new_eq:NN \__hook_tl_gset_eq:NN \tl_gset_eq:NN

```

(End of definition for `\__hook_tl_gset_eq:NN`.)

`\__hook_tl_gclear:N` Same as above.
`\__hook_tl_gclear:c`

```

58 \cs_new_protected:Npn \__hook_tl_gclear:N #1
59 { \__hook_tl_gset_eq:NN #1 \c_empty_tl }
60 \cs_generate_variant:Nn \__hook_tl_gclear:N { c }

```

(End of definition for `\__hook_tl_gclear:N`.)

4.4 Providing new hooks

4.4.1 The data structures of a hook

`\g__hook_⟨hook⟩_code_prop` Hooks have a name (called `⟨hook⟩` in the description below) and for each hook we have
`\__hook_⟨hook⟩` to provide a number of data structures. These are

`\g__hook_⟨hook⟩_reversed_tl` `\g__hook_⟨hook⟩_code_prop` A property list holding the code for the hook in separate
`\g__hook_⟨hook⟩_declared_tl` chunks. The keys are by default the package names that add code to the hook, but
`\g__hook_⟨hook⟩_parameter_tl` it is possible for packages to define other keys.
`\__hook_next_⟨hook⟩`
`\__hook_toplevel_⟨hook⟩` `\g__hook_⟨hook⟩_rule_⟨label1⟩|⟨label2⟩_tl` A token list holding the relation be-
between `⟨label1⟩` and `⟨label2⟩` in the `⟨hook⟩`. The `⟨labels⟩` are lexically (reverse)
sorted to ensure that two labels always point to the same token list. For global
rules, the `⟨hook⟩` name is `??`.

`\__hook_⟨hook⟩` The code that is actually executed when the hook is called in the doc-
ument is stored in this token list. It is constructed from the code chunks applying
the information. This token list is named like that so that in case of an error inside
the hook, the reported token list in the error is shorter, and to make it simpler to
normalize hook names in `\__hook_make_name:n`.

`\g__hook_⟨hook⟩_reversed_tl` Some hooks are “reversed”. This token list stores a `-` for
such hook so that it can be identified. The `-` character is used because `⟨reversed⟩1`
is `+1` for normal hooks and `-1` for reversed ones.

`\g__hook_⟨hook⟩_declared_tl` This token list serves as a marker for the hook being
officially declared. Its existence is tested to raise an error in case another declaration
is attempted.

`\c__hook_⟨hook⟩_parameter_tl` This token list stores the parameter text for a declared
hook (its existence almost completely intersects the token list above), which is used
for managing hooks with arguments.

`\__hook_toplevel_⟨hook⟩` This token list stores the code inserted in the hook from the user’s document, in the `top-level` label. This label is special, and doesn’t participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the `next` code chunk.

`\__hook_next_⟨hook⟩` Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main `\__hook_⟨hook⟩` token list. It is called `\__hook_next_⟨hook⟩` rather than `\__hook_next_⟨hook⟩` because otherwise a hook whose name is `next_⟨hook⟩` would clash with the next code-token list of the hook called `⟨hook⟩`.

4.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of the internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in subsequent sections.

One problem we have to solve is that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not yet been declared. For example, one package needs to write into a hook of another package, but that package may not get loaded, or is loaded only later. Another problem is that most hooks, but not the generic hooks, require a declaration.

We therefore distinguish the following states for a hook, which are managed by four different tests: structure existence (`\__hook_if_structure_exist:nTF`), creation (`\__hook_if_usable:nTF`), declaration (`\__hook_if_declared:nTF`) and disabled or not (`\__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `\__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is in this state when its basic data structure has been set up (using `\__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state “not existing”.

In this state the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.
      \__hook_if_usable:nTF returns false.
      \__hook_if_declared:nTF returns false.
      \__hook_if_disabled:nTF returns false.
```

The allowed actions are the same as in the “not existing” state.

declared A hook is in this state if it is not disabled and was explicitly declared (e.g., with `\NewHook`). In this case the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.
```

`\__hook_if_usable:nTF` returns true.
`\__hook_if_declared:nTF` returns true.
`\__hook_if_disabled:nTF` returns false.

usable A hook is in this state if it is not disabled, was not explicitly declared but nevertheless is allowed to be used (with `\UseHook` or `\hook_use:n`). This state is only possible for generic hooks as they do not need to be declared. Therefore such hooks move directly from state “not existing” to “usable” the moment a declaration such as `\AddToHook` wants to add to the hook data structure. In this state the tests give the following results:

`\__hook_if_structure_exist:nTF` returns true.
`\__hook_if_usable:nTF` returns true.
`\__hook_if_declared:nTF` returns false.
`\__hook_if_disabled:nTF` returns false.

disabled A generic hook in any state is moved to this state when `\DisableGenericHook` is used. This changes the tests to give the following results:

`\__hook_if_structure_exist:nTF` *unchanged*.
`\__hook_if_usable:nTF` returns false.
`\__hook_if_declared:nTF` returns true.
`\__hook_if_disabled:nTF` returns true.

The structure test is unchanged (if the hook was unknown before it is **false**, otherwise **true**). The usable test returns **false** so that any `\UseHook` will bypass the hook from now on. The declared test returns true so that any further `\NewHook` generates an error and the disabled test returns true so that `\AddToHook` can return an error.

FMi: maybe it should do this only after begin document?

4.4.3 Setting hooks up

`\hook_new:n`
`\hook_new_with_args:nn`
`\__hook_new:nn`

The `\hook_new:n` declaration declares a new hook and expects the hook `<name>` as its argument, e.g., `begindocument`.

```

61 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_new_with_args:nn}
62 <latexrelease> {Hooks-with-args}
63 \cs_new_protected:Npn \hook_new:n #1
64 { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} { 0 } }
65 \cs_new_protected:Npn \hook_new_with_args:nn #1 #2
66 { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} {#2} }
67 \cs_new_protected:Npn \__hook_new:nn #1 #2
68 {

```

We check if the hook was already *explicitly* declared with `\hook_new:n`, and if it already exists we complain, otherwise set the “created” flag for the hook so that it errors next time `\hook_new:n` is used.

```

69 \__hook_if_declared:nTF {#1}
70 { \msg_error:nnn { hooks } { exists } {#1} }

```

```

71 {
72   \tl_new:c { g__hook_#1_declared_tl }
73   \cs_undefine:c { __hook~#1 }
74   \cs_undefine:c { c__hook_#1_parameter_tl }
75   \__hook_make_usable:nn {#1} {#2}

```

In case there is already code in a hook, but it's undeclared, run `\__hook_update_hook_code:n` to make it ready to be executed (see test `lthooks-034`).

```

76   \__hook_update_hook_code:n {#1}
77 }
78 }
79 \<latexrelease>\EndIncludeInRelease
80 \<latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_with_args:nn}
81 \<latexrelease>{Hooks-with-args}
82 \<latexrelease>\cs_gset_protected:Npn \hook_new:n #1
83 \<latexrelease>{ \__hook_normalize_hook_args:Nn \__hook_new:n {#1} }
84 \<latexrelease>\cs_undefine:N \__hook_new:nn
85 \<latexrelease>\cs_gset_protected:Npn \__hook_new:n #1
86 \<latexrelease>{
87   \<latexrelease>\__hook_if_declared:nTF {#1}
88   \<latexrelease>{ \msg_error:nnn { hooks } { exists } {#1} }
89   \<latexrelease>{
90     \<latexrelease>\tl_new:c { g__hook_#1_declared_tl }
91     \<latexrelease>\__hook_make_usable:n {#1}
92   \<latexrelease>}
93 \<latexrelease>}
94 \<latexrelease>\cs_gset_protected:Npn \hook_new_with_args:nn #1 { }
95 \<latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_new:n`, `\hook_new_with_args:nn`, and `\__hook_new:nn`. These functions are documented on page 229.)

`\__hook_make_usable:nn` This initializes all hook data structures for the hook but if used on its own doesn't mark the hook as declared (as `\hook_new:n` does, so a later `\hook_new:n` on that hook will not result in an error. This command is internally used by `\hook_gput_code:nnn` when adding code to a generic hook.

```

96 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_make_usable:nn}
97 \<latexrelease>{Hooks-with-args}
98 \cs_new_protected:Npn \__hook_make_usable:nn #1 #2
99 {

```

Now we check if the hook's data structure can be safely created without `expl3` raising errors, then we add the hook name to the list of all hooks and allocate the necessary data structures for the new hook, otherwise just do nothing.

```

100   \__hook_if_usable:nF {#1}
101   {
102     \seq_gput_right:Nn \g__hook_all_seq {#1}

```

Here we'll define the `\c__hook_<hook>_parameter_tl` to hold a run of parameters up to the number of arguments of the hook (`#2`).

```

103     \__kernel_cs_parm_from_arg_count:nnF
104     { \tl_const:cn { c__hook_#1_parameter_tl } } {#2}
105     {
106       \msg_error:nnnn { hooks } { too-many-args } {#1} {#2}

```

```

107         \tl_const:ce { c__hook_#1_parameter_tl }
108         { \exp_not:V \c__hook_nine_parameters_tl }
109     }

```

After that, use `\__hook_normalize_cs_args:nn` to correct the number of parameters of the macros `\__hook_toplevel_<hook>` and `\__hook_next_<hook>`. We need to be able to add code with arguments to a hook without prior knowledge of the number of arguments of that hook, so `lthooks` assumes 9 until the hook is properly declared and the number of arguments is known. `\__hook_normalize_cs_args:nn` does the normalization by using the `\c__hook_<hook>_parameter_tl` defined just above.

```

110     \__hook_normalize_cs_args:nn { _toplevel } {#1}
111     \__hook_normalize_cs_args:nn { _next } {#1}

```

This is only used by the actual code of the current hook, so declare it normally:

```

112     \__hook_code_gset:nn {#1} { }

```

Now ensure that the base data structure for the hook exists:

```

113     \__hook_init_structure:n {#1}

```

The call to `\__hook_normalize_code_pool:n` will correct any improper reference to arguments that don't exist in the hook, raising a low-level `TeX` error and doubling the offending parameter tokens. It has to be done after `\__hook_init_structure:n` because it operates on `\g__hook_<hook>_code_prop`.

```

114     \__hook_normalize_code_pool:n {#1}

```

The `\g__hook_<hook>_labels_clist` holds the sorted list of labels (once it got sorted). This is used only for debugging. These are defined conditionally, in case `\__hook_make_usable:nn` is being used to redefine a hook.

```

115     \clist_if_exist:cF { g__hook_#1_labels_clist }
116     {
117         \clist_new:c { g__hook_#1_labels_clist }

```

Some hooks should reverse the default order of code chunks. To signal this we have a token list which is empty for normal hooks and contains a `-` for reversed hooks.

```

118         \tl_new:c { g__hook_#1_reversed_tl }
119     }

```

The above is all in L3 convention, but we also provide an interface to legacy `LATEX 2ε` hooks of the form `\@...hook`, e.g., `\@begindocumenthook`. There have been a few of them and they have been added to using `\g@addto@macro`. If there exists such a macro matching the name of the new hook, i.e., `\@<hook-name>hook` and it is not empty then we add its contents as a code chunk under the label `legacy`.

Warning: this support will vanish in future releases!

```

120     \__hook_include_legacy_code_chunk:n {#1}
121 }
122 }
123 <latexrelease>\EndIncludeInRelease
124 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_make_usable:nn}
125 <latexrelease>{Hooks-with-args}
126 <latexrelease>\cs_undefine:N \__hook_make_usable:nn
127 <latexrelease>\cs_gset_protected:Npn \__hook_make_usable:n #1
128 <latexrelease> {
129 <latexrelease>     \tl_if_exist:cF { __hook-#1 }
130 <latexrelease>     {

```

```

131 <latexrelease> \seq_gput_right:Nn \g__hook_all_seq {#1}
132 <latexrelease> \tl_new:c { __hook~#1 }
133 <latexrelease> \__hook_init_structure:n {#1}
134 <latexrelease> \clist_new:c { g__hook_#1_labels_clist }
135 <latexrelease> \tl_new:c { g__hook_#1_reversed_tl }
136 <latexrelease> \__hook_include_legacy_code_chunk:n {#1}
137 <latexrelease> }
138 <latexrelease> }
139 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_make_usable:nn.)

`\__hook_init_structure:n` This function declares the basic data structures for a hook without explicit declaring the hook itself. This is needed to allow adding to undeclared hooks. Here it is unnecessary to check whether all variables exist, since all three are declared at the same time (either all of them exist, or none).

It creates the hook code pool (`\g__hook_<hook>_code_prop`) and the top-level and next token lists. A hook is initialized with `\__hook_init_structure:n` the first time anything is added to it. Initializing a hook just with `\__hook_init_structure:n` will not make it usable with `\hook_use:n`.

```

140 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_init_structure:n}
141 <latexrelease> {Hooks-with-args}
142 \cs_new_protected:Npn \__hook_init_structure:n #1
143 {
144   \__hook_if_structure_exist:nF {#1}
145   {
146     \prop_new:c { g__hook_#1_code_prop }
147     \__hook_toplevel_gset:nn {#1} { }
148     \__hook_next_gset:nn {#1} { }
149   }
150 }
151 <latexrelease> \EndIncludeInRelease

152 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_init_structure:n}
153 <latexrelease> {Hooks-with-args}
154 <latexrelease> \cs_gset_protected:Npn \__hook_init_structure:n #1
155 <latexrelease> {
156   \__hook_if_structure_exist:nF {#1}
157   {
158     \prop_new:c { g__hook_#1_code_prop }
159     \tl_new:c { __hook_toplevel~#1 }
160     \tl_new:c { __hook_next~#1 }
161   }
162 <latexrelease> }
163 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_init_structure:n.)

`\hook_new_reversed:n` Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

```

\hook_new_reversed_with_args:nn
\__hook_new_reversed:nn
164 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_new_reversed_with_args:nn}
165 <latexrelease> {Hooks-with-args}
166 \cs_new_protected:Npn \hook_new_reversed:n #1
167 { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} { 0 } }

```



```

168 \cs_new_protected:Npn \hook_new_reversed_with_args:nn #1 #2
169 { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} {#2} }
170 \cs_new_protected:Npn \__hook_new_reversed:nn #1 #2
171 {
172   \__hook_if_declared:nTF {#1}
173   { \msg_error:nnn { hooks } { exists } {#1} }
174   {
175     \__hook_new:nn {#1} {#2}
176     \tl_gset:cn { g__hook_#1_reversed_tl } { - }
177   }
178 }
179 <latexrelease>\EndIncludeInRelease

180 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_reversed_with_args:nn}
181 <latexrelease> {Hooks-with-args}
182 <latexrelease>\cs_gset_protected:Npn \hook_new_reversed:n #1
183 <latexrelease> { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
184 <latexrelease>\cs_undefine:N \__hook_new_reversed:nn
185 <latexrelease>\cs_gset_protected:Npn \__hook_new_reversed:n #1
186 <latexrelease> {
187 <latexrelease>   \__hook_new:n {#1}
188 <latexrelease>   \tl_gset:cn { g__hook_#1_reversed_tl } { - }
189 <latexrelease> }
190 <latexrelease>\cs_undefine:N \__hook_new_reversed:nn
191 <latexrelease>\cs_gset_protected:Npn \hook_new_reversed_with_args:nn #1 #2 { }
192 <latexrelease>\EndIncludeInRelease

```

(End of definition for \hook_new_reversed:n, \hook_new_reversed_with_args:nn, and
__hook_new_reversed:nn. These functions are documented on page 229.)

\hook_new_pair:nn A shorthand for declaring a normal and a (matching) reversed hook in one go.

```

\hook_new_pair_with_args:nnn
193 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_new_pair_with_args:nnn}
194 <latexrelease> {Hooks-with-args}
195 \cs_new_protected:Npn \hook_new_pair:nn #1#2
196 { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} { 0 } }
197 \cs_new_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
198 { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} {#3} }
199 \cs_new_protected:Npn \__hook_new_pair:nnn #1 #2 #3
200 {
201   \__hook_if_declared:nTF {#1}
202   { \msg_error:nnn { hooks } { exists } {#1} }
203   {
204     \__hook_if_declared:nTF {#2}
205     { \msg_error:nnn { hooks } { exists } {#2} }
206     {
207       \__hook_new:nn {#1} {#3}
208       \__hook_new_reversed:nn {#2} {#3}
209     }
210   }
211 }
212 <latexrelease>\EndIncludeInRelease

213 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_pair_with_args:nnn}
214 <latexrelease> {Hooks-with-args}
215 <latexrelease>\cs_gset_protected:Npn \hook_new_pair:nn #1#2
216 <latexrelease> {

```

```

217 <latexrelease> \hook_new:n {#1}
218 <latexrelease> \hook_new_reversed:n {#2}
219 <latexrelease> }
220 <latexrelease> \cs_gset_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
221 <latexrelease> { }
222 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_new_pair:nn` and `\hook_new_pair_with_args:nnn`. These functions are documented on page 229.)

`\_hook_include_legacy_code_chunk:n` The L^AT_EX legacy concept for hooks uses with hooks the following naming scheme in the code: `\@...hook`.

If this macro is not empty we add it under the label `legacy` to the current hook and then empty it globally. This way packages or classes directly manipulating commands such as `\@begindocumenthook` still get their hook data added.

Warning: this support will vanish in future releases!

```

223 <latexrelease> \IncludeInRelease{2023/06/01}{\_hook_include_legacy_code_chunk:n}
224 <latexrelease> {Hooks-with-args}
225 \cs_new_protected:Npn \_hook_include_legacy_code_chunk:n #1
226 {

```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```

227 \tl_if_exist:cT { @#1hook }
228 {

```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```

229 \tl_if_empty:cF { @#1hook }
230 {

```

Here we set `\_hook_replacing_args_false:` because no legacy code will reference hook arguments.

```

231 \_hook_replacing_args_false:
232 \use:e
233 {
234 \_hook_hook_gput_code_do:nnn {#1} { legacy }
235 { \exp_not:v { @#1hook } }
236 }
237 \_hook_replacing_args_reset:

```

Once added to the hook, we need to clear it otherwise it might get added again later if the hook data gets updated.

```

238 \_hook_tl_gclear:c { @#1hook }
239 }
240 }
241 }

```

```

242 <latexrelease> \EndIncludeInRelease
243 <latexrelease> \IncludeInRelease{2020/10/01}{\_hook_include_legacy_code_chunk:n}
244 <latexrelease> {Hooks-with-args}
245 <latexrelease> \cs_gset_protected:Npn \_hook_include_legacy_code_chunk:n #1
246 <latexrelease> {
247 <latexrelease> \tl_if_exist:cT { @#1hook }
248 <latexrelease> {
249 <latexrelease> \tl_if_empty:cF { @#1hook }

```

```

250 <latexrelease>          {
251 <latexrelease>          \exp_args:Nnnv \__hook_hook_gput_code_do:nnn
252 <latexrelease>          {#1} { legacy } { @#1hook }
253 <latexrelease>          \__hook_tl_gclear:c { @#1hook }
254 <latexrelease>          }
255 <latexrelease>      }
256 <latexrelease>  }
257 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_include_legacy_code_chunk:n.)

4.4.4 Disabling and providing hooks

\hook_disable_generic:n Disables a hook by creating its `\g__hook_⟨hook⟩_declared_tl` so that the hook errors when used with `\hook_new:n`, then it undefines `\__hook_⟨hook⟩` so that it may not be executed.

`\__hook_disable:n`
`\__hook_if_disabled_p:n`
`\__hook_if_disabled:nTF`

This does not clear any code that may be already stored in the hook's structure, but doesn't allow adding more code. `\__hook_if_disabled:nTF` uses that specific combination to check if the hook is disabled.

```

258 <latexrelease> \IncludeInRelease{2021/06/01}{\hook_disable_generic:n}
259 <latexrelease>          {Disable~hooks}

260 \cs_new_protected:Npn \hook_disable_generic:n #1
261   { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
262 \cs_new_protected:Npn \__hook_disable:n #1
263   {
264     \tl_gclear_new:c { g__hook_#1_declared_tl }
265     \cs_undefine:c { __hook~#1 }
266   }
267 \prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
268   {
269     \bool_lazy_and:nnTF
270       { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
271       { ! \cs_if_exist_p:c { __hook~#1 } }
272     { \prg_return_true: }
273     { \prg_return_false: }
274   }
275 <latexrelease> \EndIncludeInRelease

276 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_disable_generic:n}
277 <latexrelease>          {Disable~hooks}
278 <latexrelease>
279 <latexrelease> \cs_new_protected:Npn \hook_disable_generic:n #1 {}
280 <latexrelease>
281 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_disable_generic:n`, `\__hook_disable:n`, and `\__hook_if_disabled:nTF`.
This function is documented on page 230.)

\hook_activate_generic:n The `\hook_activate_generic:n` declaration declares a new hook if it wasn't declared already, in which case it only checks that the already existing hook is not a reversed hook.

```

282 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_activate_generic:n}
283 <latexrelease>          {Providing~hooks}

```

```

284 \cs_new_protected:Npn \hook_activate_generic:n #1
285 { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { } }
286 \cs_new_protected:Npn \__hook_activate_generic:nn #1 #2
287 {

```

If the hook to be activated was disabled we warn (for now — this may change).

```

288 \__hook_if_disabled:nTF {#1}
289 { \msg_warning:nnn { hooks } { activate-disabled } {#1} }

```

Otherwise we check if the hook is not declared, and if it isn't, figure out if it's reversed or not, then declare it accordingly.

```

290 {
291 \__hook_if_declared:nF {#1}
292 {
293 \tl_new:c { g__hook_#1_declared_tl }
294 \__hook_make_usable:nn {#1} { 0 }
295 \tl_gset:ce { g__hook_#1_reversed_tl }
296 { \__hook_if_generic_reversed:nT {#1} { - } }

```

Reflect that we have activated the generic hook and set its execution code.

```

297 \__hook_update_hook_code:n {#1}
298 }
299 }
300 }

```

```

301 <latexrelease> \EndIncludeInRelease
302 <latexrelease> \IncludeInRelease{2021/06/01}{\hook_activate_generic:n}
303 <latexrelease> {Providing-hooks}
304 <latexrelease> \cs_gset_protected:Npn \__hook_activate_generic:nn #1 #2
305 <latexrelease> {
306 <latexrelease> \__hook_if_disabled:nTF {#1}
307 <latexrelease> { \msg_warning:nnn { hooks } { activate-disabled } {#1} }
308 <latexrelease> {
309 <latexrelease> \__hook_if_declared:nF {#1}
310 <latexrelease> {
311 <latexrelease> \tl_new:c { g__hook_#1_declared_tl }
312 <latexrelease> \__hook_make_usable:n {#1}
313 <latexrelease> \tl_gset:cx { g__hook_#1_reversed_tl }
314 <latexrelease> { \__hook_if_generic_reversed:nT {#1} { - } }
315 <latexrelease> \__hook_update_hook_code:n {#1}
316 <latexrelease> }
317 <latexrelease> }
318 <latexrelease> }
319 <latexrelease> \EndIncludeInRelease
320 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_activate_generic:n}
321 <latexrelease> {Providing-hooks}
322 <latexrelease> \cs_gset_protected:Npn \hook_activate_generic:n #1 { }
323 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_activate_generic:n` and `\__hook_activate_generic:n`. This function is documented on page [230](#).)

4.5 Parsing a label

`\_hook_parse_label_default:nN` This macro checks if a label was given (not `\c_novalue_tl`), and if so, tries to parse the label looking for a leading `.` to replace by `\_hook_currname_or_default:.` #2 is a boolean representing if #1 is a label name.

```

324 \cs_new:Npn \_hook_parse_label_default:nN #1#2
325 {
326   \tl_if_novalue:nTF {#1}
327     { \_hook_currname_or_default: }
328     { \tl_trim_spaces_apply:nN {#1} \_hook_parse_dot_label:nN #2 }
329 }
```

(End of definition for `\_hook_parse_label_default:nN`.)

`\_hook_parse_dot_label:nN` Start by checking if the label is empty, which raises an error, and uses the fallback value.
`\_hook_parse_dot_label:w` If not, split the label at a `./`, if any, and check if no tokens are before the `./`, or if the
`\_hook_parse_dot_label_cleanup:w` only character is a `..`. If these requirements are fulfilled, the leading `.` is replaced with
`\_hook_parse_dot_label_aux:w` `\_hook_currname_or_default:.` Otherwise the label is returned unchanged. #2 is a boolean representing if #1 is a label name.

```

330 \cs_new:Npn \_hook_parse_dot_label:nN #1#2
331 {
332   \tl_if_empty:nTF {#1}
333     {
334       \bool_if:NTF #2
335         { \msg_expandable_error:nn { hooks } { empty-label } }
336         { \msg_expandable_error:nn { hooks } { empty-hook } }
337       \_hook_currname_or_default:
338     }
339     {
340       \str_if_eq:nnTF {#1} { . }
341       { \_hook_currname_or_default: }
342       { \_hook_parse_dot_label:w #1 ./ \s_hook_mark }
343     }
344 }
345 \cs_new:Npn \_hook_parse_dot_label:w #1 ./ #2 \s_hook_mark
346 {
347   \tl_if_empty:nTF {#1}
348     { \_hook_parse_dot_label_aux:w #2 \s_hook_mark }
349     {
350       \tl_if_empty:nTF {#2}
351       { \_hook_make_name:n {#1} }
352       { \_hook_parse_dot_label_cleanup:w #1 ./ #2 \s_hook_mark }
353     }
354 }
355 \cs_new:Npn \_hook_parse_dot_label_cleanup:w #1 ./ \s_hook_mark {#1}
356 \cs_new:Npn \_hook_parse_dot_label_aux:w #1 ./ \s_hook_mark
357 { \_hook_currname_or_default: / \_hook_make_name:n {#1} }
```

(End of definition for `\_hook_parse_dot_label:nN` and others.)

`\_hook_currname_or_default:` This uses `\g_hook_hook_curr_name_tl` if it is set, otherwise it tries `\@currname`. If neither is set, it raises an error and uses the fallback value `label-missing`.

```

358 \cs_new:Npn \_hook_currname_or_default:
359 {
```

```

360 \tl_if_empty:NTF \g__hook_hook_curr_name_tl
361 {
362   \tl_if_empty:NTF \@currname
363   {
364     \msg_expandable_error:nnn { latex2e } { should-not-happen }
365     { Empty~default~label. }
366     \__hook_make_name:n { label-missing }
367   }
368   { \@currname }
369 }
370 { \g__hook_hook_curr_name_tl }
371 }

```

(End of definition for __hook_currname_or_default:.)

`\__hook_make_name:n` This provides a standard sanitization of a hook’s name. It uses `\cs:w` to build a control sequence out of the hook name, then uses `\cs_to_str:N` to get the string representation of that, without the escape character. `\cs:w`-based expansion is used instead of `e`-based because Unicode characters don’t behave well inside `\expanded`. The macro adds the `\_hook_` prefix to the hook name to reuse the hook’s code token list to build the csname and avoid leaving “public” control sequences defined (as `\relax`) in TeX’s memory.

`\__hook_make_name:w`

```

372 \cs_new:Npn \__hook_make_name:n #1
373 {
374   \exp_after:wN \exp_after:wN \exp_after:wN \__hook_make_name:w
375   \exp_after:wN \token_to_str:N \cs:w __hook~ #1 \cs_end:
376 }
377 \exp_last_unbraced:NNNNo
378 \cs_new:Npn \__hook_make_name:w #1 \tl_to_str:n { __hook~ } { }

```

(End of definition for __hook_make_name:n and __hook_make_name:w.)

`\_hook_normalize_hook_args:Nn` This is the standard route for normalizing hook and label arguments. The main macro does the entire operation within a group so that csnames made by `\__hook_make_name:n` are wiped off before continuing. This means that this function cannot be used for `\hook_use:n`!

`\_hook_normalize_hook_args:Nnn`
`\_hook_normalize_hook_rule_args:Nnnnn`
`\_hook_normalize_hook_args_aux:Nn`

```

379 \cs_new_protected:Npn \__hook_normalize_hook_args_aux:Nn #1 #2
380 {
381   \group_begin:
382   \use:e
383   {
384     \group_end:
385     \exp_not:N #1 #2
386   }
387 }
388 \cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
389 {
390   \__hook_normalize_hook_args_aux:Nn #1
391   { { \__hook_parse_label_default:nN {#2} \c_false_bool } }
392 }
393 \cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
394 {
395   \__hook_normalize_hook_args_aux:Nn #1
396   {
397     { \__hook_parse_label_default:nN {#2} \c_false_bool }

```

```

398     { \_hook_parse_label_default:nN {#3} \c_true_bool }
399   }
400 }
401 \cs_new_protected:Npn \_hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
402 {
403   \_hook_normalize_hook_args_aux:Nn #1
404   {
405     { \_hook_parse_label_default:nN {#2} \c_false_bool }
406     { \_hook_parse_label_default:nN {#3} \c_true_bool }
407     { \tl_trim_spaces:n {#4} }
408     { \_hook_parse_label_default:nN {#5} \c_true_bool }
409   }
410 }

```

(End of definition for `\_hook_normalize_hook_args:Nn` and others.)

`\_hook_curr_name_push:n` The token list `\g_hook_hook_curr_name_tl` stores the name of the current package/file to be used as the default label in hooks. Providing a consistent interface is tricky because packages can be loaded within packages, and some packages may not use `\SetDefaultHookLabel` to change the default label (in which case `\@currname` is used).

To pull that one off, we keep a stack that contains the default label for each level of input. The bottom of the stack contains the default label for the `top-level` (this stack should never go empty). If we're building the format, set the default label to be `top-level`:

```

411 \tl_gset:Nn \g_hook_hook_curr_name_tl { top-level }

```

Then, in case we're in `latexrelease` we push something on the stack to support roll forward. But in some rare cases, `latexrelease` may be loaded inside another package (notably `platexrelease`), so we'll first push the `top-level` entry:

```

412 <latexrelease> \seq_if_empty:NT \g_hook_name_stack_seq
413 <latexrelease> { \seq_gput_right:Nn \g_hook_name_stack_seq { top-level } }

```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```

414 <latexrelease> \cs_set_protected:Npn \_hook_tmp:w #1 #2 #3
415 <latexrelease> {
416 <latexrelease>   \quark_if_recursion_tail_stop:n {#1}
417 <latexrelease>   \seq_gput_right:Nn \g_hook_name_stack_seq {#1}
418 <latexrelease>   \_hook_tmp:w
419 <latexrelease> }
420 <latexrelease> \exp_after:wN \_hook_tmp:w \@currnamestack
421 <latexrelease> \q_recursion_tail \q_recursion_tail
422 <latexrelease> \q_recursion_tail \q_recursion_stop

```

and finally set the default label to be the `\@currname`:

```

423 <latexrelease> \tl_gset:Nx \g_hook_hook_curr_name_tl { \@currname }
424 <latexrelease> \seq_gpop_right:NN \g_hook_name_stack_seq \l_hook_tmpa_tl

```

Two commands keep track of the stack: when a file is input, `\_hook_curr_name_push:n` pushes the current default label onto the stack and sets the new default label (all in one go):

```

425 \cs_new_protected:Npn \_hook_curr_name_push:n #1
426 {
427   \exp_args:Ne \_hook_curr_name_push_aux:n
428   {
429     \tl_if_blank:nF {#1}

```

```

430         { \__hook_parse_dot_label:nN {#1} \c_false_bool }
431     }
432 }
433 \cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
434 {
435     \tl_if_blank:nTF {#1}
436     { \msg_error:nn { hooks } { no-default-label } }
437     {
438         \str_if_eq:nnTF {#1} { top-level }
439         {
440             \msg_error:nnnnn { hooks } { set-top-level }
441             { to } { PushDefaultHookLabel } {#1}
442         }
443         {
444             \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
445             \tl_gset:Nn \g__hook_hook_curr_name_tl {#1}
446         }
447     }
448 }

```

and when an input is over, the topmost item of the stack is popped, since that label will not be used again, and `\g__hook_hook_curr_name_tl` is updated to equal the now topmost item of the stack:

```

449 \cs_new_protected:Npn \__hook_curr_name_pop:
450 {
451     \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
452     { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
453     { \msg_error:nn { hooks } { extra-pop-label } }
454 }

```

At the end of the document we want to check if there was no `\__hook_curr_name_push:n` without a matching `\__hook_curr_name_pop:` (not a critical error, but it might indicate that something else is not quite right):

```

455 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
456 { \__hook_end_document_label_check: }
457 \cs_new_protected:Npn \__hook_end_document_label_check:
458 {
459     \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl
460     {
461         \msg_error:nne { hooks } { missing-pop-label }
462         { \g__hook_hook_curr_name_tl }
463         \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
464         \__hook_end_document_label_check:
465     }
466 }

```

The token list `\g__hook_hook_curr_name_tl` is but a mirror of the top of the stack.

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```

467 \cs_new_protected:Npn \__hook_set_default_hook_label:n #1
468 {
469     \__hook_set_default_hook_label:n
470     \seq_if_empty:NTF \g__hook_name_stack_seq
471     {
472         \msg_error:nnnnn { hooks } { set-top-level }

```



```

472     { for } { SetDefaultHookLabel } {#1}
473   }
474   {
475     \exp_args:Ne \__hook_set_default_label:n
476     {
477       \tl_if_blank:nF {#1}
478       { \__hook_parse_dot_label:nN {#1} \c_false_bool }
479     }
480   }
481 }
482 \cs_new_protected:Npn \__hook_set_default_label:n #1
483 {
484   \str_if_eq:nnTF {#1} { top-level }
485   {
486     \msg_error:nnnnn { hooks } { set-top-level }
487     { to } { SetDefaultHookLabel } {#1}
488   }
489   { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
490 }

```

(End of definition for `\__hook_curr_name_push:n` and others.)

4.6 Adding or removing hook code

With `\hook_gput_code:nnn{<hook>}{<label>}{<code>}` a chunk of `<code>` is added to an existing `<hook>` labeled with `<label>`.

```

\hook_gput_code:nnn
\hook_gput_code_with_args:nnn
\__hook_gput_code:nnn
\__hook_gput_code_store:nnn
\__hook_hook_gput_code_do:nnn
\__hook_prop_gput_labeled_cleanup:nnn
\__hook_prop_gput_labeled_do:Nnnn
\__hook_hash_check:nTF
\__hook_hash_check_aux:w
491 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_gput_code:nnn}
492 <latexrelease> {Hooks-with-args}
493 \cs_new_protected:Npn \hook_gput_code:nnn #1 #2 #3
494 {
495   \__hook_replacing_args_false:
496   \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
497   \__hook_replacing_args_reset:
498 }
499 \cs_new_protected:Npn \hook_gput_code_with_args:nnn #1 #2 #3
500 {
501   \__hook_replacing_args_true:
502   \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
503   \__hook_replacing_args_reset:
504 }

```

If `\AddToHookWithArguments` was used, do some sanity checking, and if it's not possible to use arguments at this point, fall back to regular `\AddToHook` by using `\__hook_replacing_args_false:`.

```

505 \cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
506 {
507   \__hook_chk_args_allowed:nn {#1} { AddToHook }

```

Then check if the code should be executed immediately, rather than stored:

```

508   \__hook_if_execute_immediately:nTF {#1}
509   {

```

`\AddToHookWithArguments` can't be used on one-time hooks (that were already used).

```

510     \__hook_if_replacing_args:TF
511     {

```

```

512         \msg_error:nnnn { hooks } { one-time-args }
513         {#1} { AddToHook }
514     }
515     { }
516     \use:n
517 }
518 { \__hook_gput_code_store:nnn {#1} {#2} }
519     {#3}
520 }
521 \cs_new_protected:Npn \__hook_gput_code_store:nnn #1 #2 #3
522 {

```

Then check if the hook is usable.

```

523     \__hook_if_usable:nTF {#1}

```

If so we simply add (or append) the new code to the property list holding different chunks for the hook. At `\begin{document}` this is then sorted into a token list for fast execution.

```

524     {
525         \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}

```

However, if there is an update within the document we need to alter this execution code which is done by `\__hook_update_hook_code:n`. In the preamble this does nothing.

```

526         \__hook_update_hook_code:n {#1}
527     }

```

If the hook is not usable, before giving up, check if it's not disabled and otherwise try to declare it as a generic hook, if its name matches one of the valid patterns.

```

528     {
529         \__hook_if_disabled:nTF {#1}
530         { \msg_error:nnn { hooks } { hook-disabled } {#1} }
531         { \__hook_try_declaring_generic_hook:nnn {#1} {#2} {#3} }
532     }
533 }

```

This macro will unconditionally add a chunk of code to the given hook.

```

534 \cs_new_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
535 {

```

However, first some debugging info if debugging is enabled:

```

536     \__hook_debug:n{\iow_term:e{[lthooks]~ Add~ to~
537         \__hook_if_usable:nF {#1} { undeclared~ }
538         hook~ '#1'~ (#2) \on@line
539         ^^J [lthooks] \@spaces
540         <-- \tl_to_str:n{#3}} }

```

Then try to get the code chunk labeled #2 from the hook. If there's code already there, then append #3 to that, otherwise just put #3. If the current label is `top-level`, the code is added to a dedicated token list `\__hook_toplevel_<hook>` that goes at the end of the hook (or at the beginning, for a reversed hook), just before `\__hook_next_<hook>`.

```

541     \str_if_eq:nnTF {#2} { top-level }
542     {
543         \str_if_eq:eeTF { top-level } { \__hook_currname_or_default: }
544         {

```

If the hook's basic structure does not exist, we need to declare it with `\__hook_init_-structure:n`.

```
545         \__hook_init_structure:n {#1}
```

Then append to the `_toplevel` container for the hook.

```
546         \__hook_cs_gput_right:nnn { _toplevel } {#1} {#3}
547     }
548     { \msg_error:nnn { hooks } { misused-top-level } {#1} }
549 }
550 {
```

When adding to the code pool, we have to double hashes if `\AddToHook` was used (`replacing_args` is false), so that later it is turned into a single parameter token, rather than a parameter to the hook macro. We skip this step if there are no hashes at all in the argument: the token-by-token approach otherwise becomes a major performance issue when the contents of the hook are long.

```
551     \exp_args:Ne \__hook_prop_gput_labeled_cleanup:nnn
552     {
553         \__hook_if_replacing_args:TF
554         { \exp_not:n }
555         {
556             \__hook_hash_check:nTF {#3}
557             { \exp_not:n }
558             { \__hook_double_hashes:n }
559         }
560         {#3}
561     }
562     {#1} {#2}
563 }
564 }
565 \cs_new:Npe \__hook_hash_check:nTF #1
566 {
567     \exp_not:N \exp_after:wN \exp_not:N \__hook_hash_check_aux:w
568     \exp_not:N \tl_to_str:n {#1} \c_hash_str \c_hash_str
569     \exp_not:N \q_stop
570 }
571 \use:e
572 {
573     \cs_new:Npn \exp_not:N \__hook_hash_check_aux:w
574     #1 \c_hash_str #2 \c_hash_str #3 \exp_not:N \q_stop
575 }
576 { \tl_if_empty:nTF {#3} }
```

Adds code to a hook's code pool.

```
577 \cs_new_protected:Npn \__hook_prop_gput_labeled_cleanup:nnn #1 #2 #3
578 {
579     \tl_set:Nn \l__hook_return_tl {#1}
580     \__hook_if_replacing_args:TF
581     {
582         \__hook_if_usable:nT {#2}
583         {
584             \__hook_set_normalize_fn:nn {#2}
585             { Invalid~code~added~\msg_line_context: }
586             \__hook_normalize_fn:nn {#3} {#1}
587             \prop_get:NnN \l__hook_work_prop {#3} \l__hook_return_tl
```

```

588     }
589   }
590   { }
591   \exp_args:NcV \__hook_prop_gput_labeled_do:Nnn
592   { g__hook_#2_code_prop } \l__hook_return_tl {#3}
593 }
594 \cs_new_protected:Npn \__hook_prop_gput_labeled_do:Nnn #1 #2 #3
595 {
596   \prop_get:NnNTF #1 {#3} \l__hook_return_tl
597   { \prop_gput:Nno #1 {#3} { \l__hook_return_tl #2 } }
598   { \prop_gput:Nnn #1 {#3} {#2} }
599 }
600 \<latexrelease>\EndIncludeInRelease
601 \<latexrelease>\IncludeInRelease{2020/10/01}{\hook_gput_code:nnn}
602 \<latexrelease>{Providing-hooks}
603 \<latexrelease>\cs_gset_protected:Npn \hook_gput_code:nnn #1 #2
604 \<latexrelease>{ \__hook_normalize_hook_args:Nnn
605 \<latexrelease> \__hook_gput_code:nnn {#1} {#2} }
606 \<latexrelease>\cs_gset_protected:Npn \__hook_gput_code:nnn #1 #2 #3
607 \<latexrelease>{
608 \<latexrelease> \__hook_if_execute_immediately:nTF {#1}
609 \<latexrelease> {#3}
610 \<latexrelease> {
611 \<latexrelease> \__hook_if_usable:nTF {#1}
612 \<latexrelease> {
613 \<latexrelease> \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
614 \<latexrelease> \__hook_update_hook_code:n {#1}
615 \<latexrelease> }
616 \<latexrelease> {
617 \<latexrelease> \__hook_if_disabled:nTF {#1}
618 \<latexrelease> { \msg_error:nnn { hooks } { hook-disabled } {#1} }
619 \<latexrelease> { \__hook_try_declaring_generic_hook:nnn
620 \<latexrelease> {#1} {#2} {#3} }
621 \<latexrelease> }
622 \<latexrelease> }
623 \<latexrelease> }
624 \<latexrelease>\cs_gset_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
625 \<latexrelease>{
626 \<latexrelease> \__hook_debug:n{\iow_term:x{****~ Add~ to~
627 \<latexrelease> \__hook_if_usable:nF {#1} { undeclared~ }
628 \<latexrelease> hook~ #1~ (#2)
629 \<latexrelease> \on@line\space <-- \tl_to_str:n{#3}} }
630 \<latexrelease>\str_if_eq:nnTF {#2} { top-level }
631 \<latexrelease>{
632 \<latexrelease> \str_if_eq:eeTF { top-level }
633 \<latexrelease> { \__hook_currname_or_default: }
634 \<latexrelease> {
635 \<latexrelease> \__hook_init_structure:n {#1}
636 \<latexrelease> \__hook_tl_gput_right:cn { \__hook_toplevel~#1 } {#3}
637 \<latexrelease> }
638 \<latexrelease> { \msg_error:nnn { hooks } { misused-top-level } {#1} }
639 \<latexrelease> }
640 \<latexrelease> {

```

```

641 <latexrelease> \prop_get:cnNTF
642 <latexrelease> { g__hook_#1_code_prop } {#2} \l__hook_return_tl
643 <latexrelease> {
644 <latexrelease> \prop_gput:cno { g__hook_#1_code_prop } {#2}
645 <latexrelease> { \l__hook_return_tl #3 }
646 <latexrelease> }
647 <latexrelease> { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
648 <latexrelease> }
649 <latexrelease> }
650 <latexrelease> \cs_gset_protected:Npn \hook_gput_code_with_args:nnn #1#2#3 { }
651 <latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_gput_code:nnn and others. These functions are documented on page 231.)

`\__hook_chk_args_allowed:nn` This macro checks if it is possible to add code with references to a hook's arguments for hook #1. It only does something if the function being run is `replacing_args`. This macro will error if the hook is declared and takes no arguments, then it will set `\__hook_replacing_args_false:` so that the macro which called it will add the code normally.

```

652 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_chk_args_allowed:nn}
653 <latexrelease> {Hooks-with-args}
654 \cs_new_protected:Npn \__hook_chk_args_allowed:nn #1 #2
655 {
656 \__hook_if_replacing_args:TF
657 {
658 \__hook_if_declared:nT {#1}
659 { \tl_if_empty:cT { c__hook_#1_parameter_tl } { \use_ii:nn } }
660 \use_none:n
661 {
662 \msg_error:nnnn { hooks } { without-args } {#1} {#2}
663 \__hook_replacing_args_false:
664 }
665 }
666 { }
667 }
668 <latexrelease> \EndIncludeInRelease
669 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_chk_args_allowed:nn}
670 <latexrelease> {Hooks-with-args}
671 <latexrelease> \cs_undefine:N \__hook_chk_args_allowed:nn
672 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_chk_args_allowed:nn.)

`\__hook_gput_undeclared_hook:nnn` Often it may happen that a package *A* defines a hook `foo`, but package *B*, that adds code to that hook, is loaded before *A*. In such case we need to add code to the hook before it is declared. An implicitly declared hook doesn't have arguments (in principle), so use `\c_false_bool` here.

```

673 \cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
674 {
675 \__hook_init_structure:n {#1}
676 \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
677 }

```

(End of definition for __hook_gput_undeclared_hook:nnn.)

`\_hook_try_declaring_generic_hook:nnn`
`\_hook_try_declaring_generic_next_hook:nn`

These entry-level macros just pass the arguments along to the common `\_hook_try_declaring_generic_hook:nnnn` with the right functions to execute when some action is to be taken.

The wrapper `\_hook_try_declaring_generic_hook:nnn` then defers `\hook_gput_code:nnn` if the generic hook was declared, or to `\_hook_gput_undeclared_hook:nnn` otherwise (the hook was tested for existence before, so at this point if it isn't generic, it doesn't exist).

The wrapper `\_hook_try_declaring_generic_next_hook:nn` for next-execution hooks does the same: it defers the code to `\hook_gput_next_code:nn` if the generic hook was declared, or to `\_hook_gput_next_do:nn` otherwise.

```

678 <latexrelease> \IncludeInRelease{2023/06/01}
679 <latexrelease> {\_hook_try_declaring_generic_hook:nnn}
680 <latexrelease> {Hooks-with-args}
681 \cs_new_protected:Npn \_hook_try_declaring_generic_hook:nnn #1
682 {
683   \_hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
684   \_hook_gput_code:nnn
685   \_hook_gput_undeclared_hook:nnn
686   {#1}
687 }
688 \cs_new_protected:Npn \_hook_try_declaring_generic_next_hook:nn #1
689 {
690   \_hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
691   \_hook_gput_next_code:nn
692   \_hook_gput_next_do:nn
693   {#1}
694 }
695 <latexrelease> \EndIncludeInRelease
696 <latexrelease> \IncludeInRelease{2021/11/15}
697 <latexrelease> {\_hook_try_declaring_generic_hook:nnn}
698 <latexrelease> {Standardize-generic-hook-names}
699 <latexrelease> \cs_gset_protected:Npn \_hook_try_declaring_generic_hook:nnn #1
700 <latexrelease> {
701 <latexrelease>   \_hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
702 <latexrelease>   {#1}
703 <latexrelease>   \hook_gput_code:nnn
704 <latexrelease>   \_hook_gput_undeclared_hook:nnn
705 <latexrelease>   {#1}
706 <latexrelease> }
707 <latexrelease> \cs_gset_protected:Npn
708 <latexrelease>   \_hook_try_declaring_generic_next_hook:nn #1
709 <latexrelease> {
710 <latexrelease>   \_hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
711 <latexrelease>   {#1}
712 <latexrelease>   \hook_gput_next_code:nn
713 <latexrelease>   \_hook_gput_next_do:nn
714 <latexrelease>   {#1}
715 <latexrelease> }
716 <latexrelease> \EndIncludeInRelease
717 <latexrelease> \IncludeInRelease{2020/10/01}
718 <latexrelease> {\_hook_try_declaring_generic_hook:nnn}
719 <latexrelease> {Standardize-generic-hook-names}
720 <latexrelease> \cs_new_protected:Npn

```

```

721 <latexrelease> \__hook_try_declaring_generic_hook:nnn #1
722 <latexrelease> {
723 <latexrelease> \__hook_try_declaring_generic_hook:nNNnn {#1}
724 <latexrelease> \hook_gput_code:nnn \__hook_gput_undeclared_hook:nnn
725 <latexrelease> }
726 <latexrelease> \cs_new_protected:Npn
727 <latexrelease> \__hook_try_declaring_generic_next_hook:nn #1
728 <latexrelease> {
729 <latexrelease> \__hook_try_declaring_generic_hook:nNNnn {#1}
730 <latexrelease> \hook_gput_next_code:nn \__hook_gput_next_do:nn
731 <latexrelease> }

```

(End of definition for __hook_try_declaring_generic_hook:nnn and
__hook_try_declaring_generic_next_hook:nn.)

__hook_try_declaring_generic_hook:nNNnn
hook_try_declaring_generic_hook_split:nNNnn

__hook_try_declaring_generic_hook:nNNnn now splits the hook name at the first / (if any) and first checks if it is a file-specific hook (they require some normalization) using __hook_if_file_hook:wTF. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns <true> for a generic hook and <false> in other cases.

```

732 <latexrelease> \cs_new_protected:Npn \__hook_try_declaring_generic_hook:nNNnn #1
733 <latexrelease> {
734 <latexrelease> \__hook_if_file_hook:wTF #1 / / \s__hook_mark
735 <latexrelease> {
736 <latexrelease> \exp_args:Ne
737 <latexrelease> \__hook_try_declaring_generic_hook_split:nNNnn
738 <latexrelease> { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
739 <latexrelease> }
740 <latexrelease> { \__hook_try_declaring_generic_hook_split:nNNnn {#1} }
741 <latexrelease> }
742 <latexrelease> \cs_new_protected:Npn
743 <latexrelease> \__hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
744 <latexrelease> {
745 <latexrelease> \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop:
746 <latexrelease> {#1}
747 <latexrelease> { #2 }
748 <latexrelease> { #3 } {#1}
749 <latexrelease> }
750 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_try_declaring_generic_hook:nNNnn and
__hook_try_declaring_generic_hook_split:nNNnn.)

__hook_try_declaring_generic_hook:wnTF

```

751 <latexrelease> \IncludeInRelease{2023/06/01}
752 <latexrelease> { \__hook_try_declaring_generic_hook:wn }
753 <latexrelease> { Hooks-with-args }
754 \prg_new_protected_conditional:Npnn
755 \__hook_try_declaring_generic_hook:wn
756 #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
757 {
758 \__hook_if_generic:nTF {#5}
759 {
760 \__hook_if_usable:nF {#5}
761 {

```

If the hook doesn't exist yet we check if it is a `cmd` hook and if so we attempt patching the command in addition to declaring the hook.

For some commands this will not be possible, in which case `\__hook_patch_cmd_-or_delay:Nnn` (defined in `ltxcmdhooks`) will generate an appropriate error message.

```

762         \str_if_eq:nnT {#1} { cmd }
763         {
764             \__hook_try_put_cmd_hook:n {#5}
765             \__hook_make_usable:nn {#5} { 9 }
766             \use_none:nnn
767         }

```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use `\__hook_make_usable:nn`, so that a `\hook_new:n` is still possible later. Generic hooks (except `cmd` hooks) take no arguments, so use zero as the second argument.

```

768         \__hook_make_usable:nn {#5} { 0 }
769     }
770     \__hook_if_generic_reversed:nT {#5}
771     { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
772     \prg_return_true:
773 }
774 {

```

Generic hooks are all named `<type>/<name>/<place>`, where `<type>` and `<place>` are predefined (`\c__hook_generic_<type>/./<place>_tl`), and `<name>` is the variable component. Older releases had some hooks with the `<name>` in the third part, so the code below supports that syntax for a while, with a warning.

The `\exp_after:wN ... \exp:w` trick is there to remove the conditional structure inserted by `\__hook_try_declaring_generic_hook:wnTF` and thus allow access to the tokens that follow it, as is needed to keep things going.

When the deprecation cycle ends, the lines below should all be replaced by `\prg_return_false:`.

```

775     \__hook_if_deprecated_generic:nTF {#5}
776     {
777         \__hook_deprecated_generic_warn:n {#5}
778         \exp_after:wN \__hook_declare_deprecated_generic:NNn
779         \exp:w % \exp_end:
780     }
781     { \prg_return_false: }
782 }
783 }

```

`\__hook_deprecated_generic_warn:n` will issue a deprecation warning for a given hook, and mark that hook such that the warning will not be issued again (multiple warnings can be issued, but only once per hook).

```

\__hook_deprecated_generic_warn:Nn
\__hook_deprecated_generic_warn:Nw

```

```

784 \cs_new_protected:Npn \__hook_deprecated_generic_warn:n #1
785 { \__hook_deprecated_generic_warn:w #1 \s__hook_mark }
786 \cs_new_protected:Npn \__hook_deprecated_generic_warn:w
787 #1 / #2 / #3 \s__hook_mark
788 {
789     \if_cs_exist:w __hook~#1/#2/#3 \cs_end: \else:
790         \msg_warning:nnnnn { hooks } { generic-deprecated } {#1} {#2} {#3}
791     \fi:

```



```

792 \cs_gset_eq:cN { __hook~#1/#2/#3 } \scan_stop:
793 }

```

Now that the user has been told about the deprecation, we proceed by swapping `<name>` and `<place>` and adding the code to the correct hook.

```

__hook_do_deprecated_generic:Nn
__hook_do_deprecated_generic:Nw
794 \cs_new_protected:Npn __hook_do_deprecated_generic:Nn #1 #2
795 { __hook_do_deprecated_generic:Nw #1 #2 \s_hook_mark }
__hook_declare_deprecated_generic:NNw
796 \cs_new_protected:Npn __hook_do_deprecated_generic:Nw #1
797 #2 / #3 / #4 \s_hook_mark
798 { #1 { #2 / #4 / #3 } }
799 \cs_new_protected:Npn __hook_declare_deprecated_generic:NNn #1 #2 #3
800 { __hook_declare_deprecated_generic:NNw #1 #2 #3 \s_hook_mark }
801 \cs_new_protected:Npn __hook_declare_deprecated_generic:NNw #1 #2
802 #3 / #4 / #5 \s_hook_mark
803 {
804 __hook_try_declaring_generic_hook:wnTF #3 / #5 / #4 / \scan_stop:
805 { #3 / #5 / #4 }
806 #1 #2 { #3 / #5 / #4 }
807 }
808 \<latexrelease>\EndIncludeInRelease

809 \<latexrelease>\IncludeInRelease{2021/11/15}
810 \<latexrelease> {__hook_try_declaring_generic_hook:wn}
811 \<latexrelease> {Standardize-generic-hook-names}
812 \<latexrelease>\prg_new_protected_conditional:Npnn
813 \<latexrelease> __hook_try_declaring_generic_hook:wn
814 \<latexrelease> #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
815 \<latexrelease> {
816 \<latexrelease> __hook_if_generic:nTF {#5}
817 \<latexrelease> {
818 \<latexrelease> __hook_if_usable:nF {#5}
819 \<latexrelease> {
820 \<latexrelease> \str_if_eq:nnT {#1} { cmd }
821 \<latexrelease> { __hook_try_put_cmd_hook:n {#5} }
822 \<latexrelease> __hook_make_usable:n {#5}
823 \<latexrelease> }
824 \<latexrelease> __hook_if_generic_reversed:nT {#5}
825 \<latexrelease> { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
826 \<latexrelease> \prg_return_true:
827 \<latexrelease> }
828 \<latexrelease> {
829 \<latexrelease> __hook_if_deprecated_generic:nTF {#5}
830 \<latexrelease> {
831 \<latexrelease> __hook_deprecated_generic_warn:n {#5}
832 \<latexrelease> \exp_after:wN __hook_declare_deprecated_generic:NNn
833 \<latexrelease> \exp:w % \exp_end:
834 \<latexrelease> }
835 \<latexrelease> { \prg_return_false: }
836 \<latexrelease> }
837 \<latexrelease> }
838 \<latexrelease>\EndIncludeInRelease

839 \<latexrelease>\IncludeInRelease{2021/06/01}
840 \<latexrelease> {__hook_try_declaring_generic_hook:wn}
841 \<latexrelease> {Support~cmd~hooks}

```

```

842 <latexrelease> \prg_new_protected_conditional:Npnn
843 <latexrelease>     \__hook_try_declaring_generic_hook:wn
844 <latexrelease>     #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
845 <latexrelease> {
846 <latexrelease>     \tl_if_empty:nTF {#2}
847 <latexrelease>     { \prg_return_false: }
848 <latexrelease>     {
849 <latexrelease>         \prop_if_in:NnTF \c__hook_generics_prop {#1}
850 <latexrelease>         {
851 <latexrelease>             \__hook_if_usable:nF {#5}
852 <latexrelease>             {
853 <latexrelease>                 \str_if_eq:nnT {#1} { cmd }
854 <latexrelease>                 { \__hook_try_put_cmd_hook:n {#5} }
855 <latexrelease>                 \__hook_make_usable:n {#5}
856 <latexrelease>             }
857 <latexrelease>         \prop_if_in:NnTF
858 <latexrelease>         \c__hook_generics_reversed_ii_prop {#2}
859 <latexrelease>         { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
860 <latexrelease>         {
861 <latexrelease>             \prop_if_in:NnT
862 <latexrelease>             \c__hook_generics_reversed_iii_prop {#3}
863 <latexrelease>             { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
864 <latexrelease>         }
865 <latexrelease>         \prg_return_true:
866 <latexrelease>     }
867 <latexrelease>     { \prg_return_false: }
868 <latexrelease> }
869 <latexrelease> }
870 <latexrelease> \EndIncludeInRelease

871 <latexrelease> \IncludeInRelease{2020/10/01}
872 <latexrelease>     { \__hook_try_declaring_generic_hook:wn }
873 <latexrelease>     { Support~cmd-hooks }
874 <latexrelease> \prg_new_protected_conditional:Npnn
875 <latexrelease>     \__hook_try_declaring_generic_hook:wn
876 <latexrelease>     #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
877 <latexrelease> {
878 <latexrelease>     \tl_if_empty:nTF {#2}
879 <latexrelease>     { \prg_return_false: }
880 <latexrelease>     {
881 <latexrelease>         \prop_if_in:NnTF \c__hook_generics_prop {#1}
882 <latexrelease>         {
883 <latexrelease>             \__hook_if_declared:nF {#5} { \hook_new:n {#5} }
884 <latexrelease>             \prop_if_in:NnTF
885 <latexrelease>             \c__hook_generics_reversed_ii_prop {#2}
886 <latexrelease>             { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
887 <latexrelease>             {
888 <latexrelease>                 \prop_if_in:NnT
889 <latexrelease>                 \c__hook_generics_reversed_iii_prop {#3}
890 <latexrelease>                 { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
891 <latexrelease>             }
892 <latexrelease>             \prg_return_true:
893 <latexrelease>         }
894 <latexrelease>         { \prg_return_false: }
895 <latexrelease>     }

```

```

896 <latexrelease> }
897 <latexrelease> \EndIncludeInRelease

(End of definition for \__hook_try_declaring_generic_hook:wTF and others.)

```

`\__hook_if_file_hook:w` `\__hook_if_file_hook:wTF` checks if the argument is a valid file-specific hook (not, for example, `file/before`, but `file/foo.tex/before`). If it is a file-specific hook, then it executes the `<true>` branch, otherwise `<false>`.

```

898 <latexrelease> \IncludeInRelease{2021/11/15}{\__hook_if_file_hook:w}
899 <latexrelease> {Standardize-generic-hook-names}
900 <latexrelease> \EndIncludeInRelease
901 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_if_file_hook:w}
902 <latexrelease> {Standardize-generic-hook-names}
903 <latexrelease> \prg_new_conditional:Npnn \__hook_if_file_hook:w
904 <latexrelease> #1 / #2 / #3 \s__hook_mark { TF }
905 <latexrelease> {
906 <latexrelease> \str_if_eq:nnTF {#1} { file }
907 <latexrelease> {
908 <latexrelease> \bool_lazy_or:nnTF
909 <latexrelease> { \tl_if_empty_p:n {#3} }
910 <latexrelease> { \str_if_eq_p:nn {#3} { / } }
911 <latexrelease> { \prg_return_false: }
912 <latexrelease> {
913 <latexrelease> \prop_if_in:NnTF \c__hook_generics_file_prop {#2}
914 <latexrelease> { \prg_return_true: }
915 <latexrelease> { \prg_return_false: }
916 <latexrelease> }
917 <latexrelease> }
918 <latexrelease> { \prg_return_false: }
919 <latexrelease> }
920 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\__hook_if_file_hook:wTF`.)

```

\__hook_file_hook_normalize:n
\__hook_strip_double_slash:n 921 <latexrelease> \IncludeInRelease{2021/11/15}{\__hook_file_hook_normalize:n}
\__hook_strip_double_slash:w 922 <latexrelease> {Standardize-generic-hook-names}
923 <latexrelease> \EndIncludeInRelease

```

When a file-specific hook is found, before being declared it is lightly normalized by `\__hook_file_hook_normalize:n`. The current implementation just replaces two consecutive slashes (`//`) by a single one, to cope with simple cases where the user did something like `\def\input@path{./mypath/}`, in which case a hook would have to be `\AddToHook{file}/./mypath//file.tex/after`.

```

924 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_file_hook_normalize:n}
925 <latexrelease> {Standardize-generic-hook-names}
926 <latexrelease> \cs_new:Npn \__hook_file_hook_normalize:n #1
927 <latexrelease> { \__hook_strip_double_slash:n {#1} }
928 <latexrelease> \cs_new:Npn \__hook_strip_double_slash:n #1
929 <latexrelease> { \__hook_strip_double_slash:w #1 // \s__hook_mark }

```

This function is always called after testing if the argument is a file hook with `\__hook_if_file_hook:wTF`, so we can assume it has three parts (it is either `file/.../before` or `file/.../after`), so we use `#1/#2/#3 //` instead of just `#1 //` to prevent losing a slash if the file name is empty.

```

930 <latexrelease>\cs_new:Npn \__hook_strip_double_slash:w #1/#2/#3//#4\s__hook_mark
931 <latexrelease> {
932 <latexrelease> \tl_if_empty:nTF {#4}
933 <latexrelease> { #1/#2/#3 }
934 <latexrelease> { \__hook_strip_double_slash:w #1/#2/#3 /#4\s__hook_mark }
935 <latexrelease> }
936 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_file_hook_normalize:n, __hook_strip_double_slash:n, and __hook_strip_double_slash:w.)

```

\c__hook_generic_cmd/.before_tl
\c__hook_generic_cmd/.after_tl
\c__hook_generic_env/.before_tl
\c__hook_generic_env/.after_tl
\c__hook_generic_file/.before_tl
\c__hook_generic_file/.after_tl
\c__hook_generic_package/.before_tl
\c__hook_generic_package/.after_tl
\c__hook_generic_class/.before_tl
\c__hook_generic_class/.after_tl
\c__hook_generic_include/.before_tl
\c__hook_generic_include/.after_tl
\c__hook_generic_env/.begin_tl
\c__hook_generic_env/.end_tl
\c__hook_generic_include/.end_tl

```

Token lists defining the possible generic hooks. We don't provide any user interface to this as this is meant to be static.

cmd The generic hooks used for commands.

env The generic hooks used in \begin and \end.

file, package, class, include The generic hooks used when loading a file

```

937 <latexrelease>\IncludeInRelease{2021/11/15}{\c__hook_generics_prop}
938 <latexrelease> {Standardize-generic-hook-names}
939 \clist_map_inline:nn { cmd , env , file , package , class , include }
940 {
941 \tl_const:cn { c__hook_generic_#1/.before_tl } { + }
942 \tl_const:cn { c__hook_generic_#1/.after_tl } { - }
943 }
944 \tl_const:cn { c__hook_generic_env/.begin_tl } { + }
945 \tl_const:cn { c__hook_generic_env/.end_tl } { + }
946 \tl_const:cn { c__hook_generic_include/.end_tl } { - }
947 \tl_const:cn { c__hook_generic_include/.excluded_tl } { + }

```

Deprecated generic hooks:

```

948 \clist_map_inline:nn { file , package , class , include }
949 {
950 \tl_const:cn { c__hook_deprecated_#1/.before_tl } { }
951 \tl_const:cn { c__hook_deprecated_#1/.after_tl } { }
952 }
953 \tl_const:cn { c__hook_deprecated_include/.end_tl } { }
954 <latexrelease>\EndIncludeInRelease
955 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_generics_prop}
956 <latexrelease> {Standardize-generic-hook-names}
957 <latexrelease>\prop_const_from_keyval:Nn \c__hook_generics_prop
958 <latexrelease> {cmd=,env=,file=,package=,class=,include=}
959 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_generic_cmd/.before_tl and others.)

```

\c__hook_generics_reversed_ii_prop
\c__hook_generics_reversed_iii_prop
\c__hook_generics_file_prop

```

The following generic hooks are supposed to use reverse ordering (the ii and iii names are kept for the deprecation cycle):

```

960 <latexrelease>\IncludeInRelease{2021/11/15}{\c__hook_generics_reversed_ii_prop}
961 <latexrelease> {Standardize-generic-hook-names}
962 <latexrelease>\EndIncludeInRelease

```

```

963 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_generics_reversed_ii_prop}
964 <latexrelease>{Standardize-generic-hook-names}
965 <latexrelease>\prop_const_from_keyval:Nn
966 <latexrelease>\c__hook_generics_reversed_ii_prop {after=,end=}
967 <latexrelease>\prop_const_from_keyval:Nn
968 <latexrelease>\c__hook_generics_reversed_iii_prop {after=}
969 <latexrelease>\prop_const_from_keyval:Nn
970 <latexrelease>\c__hook_generics_file_prop {before=,after=}
971 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_generics_reversed_ii_prop, \c__hook_generics_reversed_iii_prop, and \c__hook_generics_file_prop.)

\c__hook_parameter_cmd/.before_tl
\c__hook_parameter_cmd/.after_tl

Token lists defining the number of arguments for a given type of generic hook.

```

972 <latexrelease>\IncludeInRelease{2023/06/01}{\c__hook_parameter_cmd/.before_tl}
973 <latexrelease>{Hooks-with-args}

```

cmd hooks are declared with 9 arguments because they have a variable number of arguments (depending on the command they are attached to), so we use the maximum here.

```

974 \tl_const:cn { c__hook_parameter_cmd/.before_tl } { #1#2#3#4#5#6#7#8#9 }
975 \tl_const:cn { c__hook_parameter_cmd/.after_tl } { #1#2#3#4#5#6#7#8#9 }
976 <latexrelease>\EndIncludeInRelease
977 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_parameter_cmd/.before_tl}
978 <latexrelease>{Hooks-with-args}
979 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_parameter_cmd/.before_tl and \c__hook_parameter_cmd/.after_tl.)

\hook_gremove_code:nn
_hook_gremove_code:nn

With \hook_gremove_code:nn{<hook>}{<label>} any code for <hook> stored under <label> is removed.

```

980 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_gremove_code:nn}
981 <latexrelease>{Hooks-with-args}
982 \cs_new_protected:Npn \hook_gremove_code:nn #1 #2
983 { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
984 \cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
985 {

```

First check that the hook code pool exists. __hook_if_usable:nTF isn't used here because it should be possible to remove code from a hook before its defined (see section 2.1.8).

```

986 \__hook_if_structure_exist:nTF {#1}
987 {

```

Then remove the chunk and run __hook_update_hook_code:n so that the execution token list reflects the change if we are after \begin{document}.

If all code is to be removed, clear the code pool \g__hook_<hook>_code_prop, the top-level code __hook_toplevel_<hook>, and the next-execution code __hook_next_<hook>.

```

988 \str_if_eq:nnTF {#2} {*}
989 {
990 \prop_gclear:c { g__hook_#1_code_prop }
991 \__hook_toplevel_gset:nn {#1} { }
992 \__hook_next_gset:nn {#1} { }

```

```

993     }
994     {

```

If the label is `top-level` then clear the token list, as all code there is under the same label.

```

995         \str_if_eq:nnTF {#2} { top-level }
996         { \__hook_toplevel_gset:nn {#1} { } }
997         {
998             \prop_gpop:cnNF { g__hook_#1_code_prop }
999             {#2} \l__hook_return_tl
1000             { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
1001         }
1002     }

```

Finally update the code, if the hook exists.

```

1003     \__hook_if_usable:nT {#1}
1004     { \__hook_update_hook_code:n {#1} }
1005 }

```

If the code pool for this hook doesn't exist, show a warning:

```

1006 {
1007     \__hook_if_deprecated_generic:nTF {#1}
1008     {
1009         \__hook_deprecated_generic_warn:n {#1}
1010         \__hook_do_deprecated_generic:Nn
1011         \__hook_gremove_code:nn {#1} {#2}
1012     }
1013     { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
1014 }
1015 }

```

```

1016 \<latexrelease>\EndIncludeInRelease

```

```

1017 \<latexrelease>\IncludeInRelease{2020/10/01}{\hook_gremove_code:nn}
1018 \<latexrelease>{Hooks-with-args}
1019 \<latexrelease>\cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
1020 \<latexrelease> {
1021     \<latexrelease> \__hook_if_structure_exist:nTF {#1}
1022     {
1023         \<latexrelease> \str_if_eq:nnTF {#2} { *}
1024         {
1025             \<latexrelease> \prop_gclear:c { g__hook_#1_code_prop }
1026             \<latexrelease> \__hook_tl_gclear:c { __hook_toplevel~#1 }
1027             \<latexrelease> \__hook_tl_gclear:c { __hook_next~#1 }
1028         }
1029         {
1030             \<latexrelease> \str_if_eq:nnTF {#2} { top-level }
1031             { \<latexrelease> \__hook_tl_gclear:c { __hook_toplevel~#1 } }
1032             {
1033                 \<latexrelease> \prop_gpop:cnNF { g__hook_#1_code_prop }
1034                 {#2} \l__hook_return_tl
1035                 { \<latexrelease> \msg_warning:nnnn { hooks } { cannot-remove }
1036                     {#1} {#2} }
1037             }
1038         }
1039         \<latexrelease> \__hook_if_usable:nT {#1}
1040         { \<latexrelease> \__hook_update_hook_code:n {#1} }

```

```

1041 <latexrelease>      }
1042 <latexrelease>      {
1043 <latexrelease>      \_hook_if_deprecated_generic:nTF {#1}
1044 <latexrelease>      {
1045 <latexrelease>      \_hook_deprecated_generic_warn:n {#1}
1046 <latexrelease>      \_hook_do_deprecated_generic:Nn
1047 <latexrelease>      \_hook_gremove_code:nn {#1} {#2}
1048 <latexrelease>      }
1049 <latexrelease>      { \msg_warning:nnnn { hooks } { cannot-remove }
1050 <latexrelease>      {#1} {#2} }
1051 <latexrelease>      }
1052 <latexrelease>    }
1053 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_gremove_code:nn` and `\_hook_gremove_code:nn`. This function is documented on page [231](#).)

```

\_hook_cs_gput_right:nnn
\_hook_cs_gput_right_fast:nnn
\_hook_cs_gput_right_slow:nnn
\_hook_code_gset_auxi:nnnn
\_hook_code_gset_auxi:eeen

```

This macro is used to append code to the `toplevel` and `next` token lists, treating them correctly depending on their number of arguments, and depending on whether the code being added should have parameter tokens understood as parameters, or doubled to be stored as parameter tokens.

```

1054 <latexrelease> \IncludeInRelease{2023/06/01}{\_hook_cs_gput_right:nnn}
1055 <latexrelease> {Hooks~with~args}

```

Check if the current hook is declared and takes no arguments. In this case, we short-circuit and use the simpler and much faster approach that doesn't require hash-doubling.

```

1056 \cs_new_protected:Npn \_hook_cs_gput_right:nnn #1 #2
1057 {
1058   \if:w T
1059     \_hook_if_declared:nF {#2} { F }
1060     \tl_if_empty:cF { c\_hook\_#2\_parameter\_tl } { F }
1061     T
1062     \exp_after:wN \_hook_cs_gput_right_fast:nnn
1063   \else:
1064     \exp_after:wN \_hook_cs_gput_right_slow:nnn
1065   \fi:
1066   {#1} {#2}
1067 }
1068 \cs_new_protected:Npn \_hook_cs_gput_right_fast:nnn #1 #2 #3
1069 { \cs_gset:cpe { \_hook#1~#2 }
1070   { \exp_not:v { \_hook#1~#2 } \exp_not:n {#3} } }
1071 \cs_new_protected:Npn \_hook_cs_gput_right_slow:nnn #1 #2 #3
1072 {

```

The auxiliary `\_hook_code_gset_auxi:eeen` just does the assignment at the end. Its first argument is the parameter text of the macro, which is chosen here depending if `\c\_hook\_hook\_parameter\_tl` exists, if the hook is declared, and if it's a generic hook.

```

1073   \cs_if_exist:cF { \_hook#1~#2 }
1074   { \_hook_code_gset_aux:nnn {#1} {#2} { } }
1075   \_hook_code_gset_auxi:eeen
1076   {
1077     \_hook_if_declared:nTF {#2}
1078     { \tl_use:c { c\_hook\_#2\_parameter\_tl } }
1079     {
1080       \_hook_if_generic:nTF {#2}

```

```

1081         { \_hook_generic_parameter:n {#2} }
1082         { \c_hook_nine_parameters_tl }
1083     }
1084 }

```

Here we take the existing code in the macro, expand it with as many arguments as it takes, then double the hashes so the code can be reused.

PhO: Maybe can be improved. The case of adding to an empty cs can be optimized by quickly checking \cs_replacement_spec.

```

1085 {
1086     \exp_args:NNo \exp_args:No \_hook_double_hashes:n
1087     {
1088         \cs:w \_hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1089         { \_hook_braced_cs_parameter:n { \_hook#1~#2 } }
1090     }
1091 }

```

Now the new code: if we are replacing arguments, then hashes are left untouched, otherwise they are doubled.

```

1092 {
1093     \_hook_if_replacing_args:TF
1094     { \exp_not:n }
1095     { \_hook_double_hashes:n }
1096     {#3}
1097 }

```

And finally, the csname which we'll define with all the above.

```

1098 { \_hook#1~#2 }
1099 }

```

And as promised, the auxiliary that does the definition.

```

1100 \cs_new_protected:Npn \_hook_code_gset_auxi:nnnn #1 #2 #3 #4
1101 { \cs_gset:cpn {#4} #1 { #2 #3 } }
1102 \cs_generate_variant:Nn \_hook_code_gset_auxi:nnnn { een }
1103 <latexrelease>\EndIncludeInRelease
1104 <latexrelease>\IncludeInRelease{2020/10/01}{\_hook_cs_gput_right:nnn}
1105 <latexrelease>{Hooks-with-args}
1106 <latexrelease>\cs_undefine:N \_hook_cs_gput_right:nnn
1107 <latexrelease>\cs_undefine:N \_hook_cs_gput_right_fast:nnn
1108 <latexrelease>\cs_undefine:N \_hook_cs_gput_right_slow:nnn
1109 <latexrelease>\cs_undefine:N \_hook_code_gset_auxi:nnnn
1110 <latexrelease>\EndIncludeInRelease

```

(End of definition for _hook_cs_gput_right:nnn and others.)

These macros define _hook<type>_<hook> (with <type> being _next, _toplevel, or empty) with the given code and the parameters stored in \c_hook_<hook>_parameter_tl (or none, if that doesn't exist).

```

\_hook_code_gset:nn
\_hook_code_gset:ne
\_hook_toplevel_gset:nn
\_hook_next_gset:nn
\_hook_code_gset_aux:nnn
1111 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_code_gset:nn}
1112 <latexrelease>{Hooks-with-args}
1113 \cs_new_protected:Npn \_hook_code_gset:nn
1114 { \_hook_code_gset_aux:nnn { } }
1115 \cs_new_protected:Npn \_hook_toplevel_gset:nn
1116 { \_hook_code_gset_aux:nnn { _toplevel } }
1117 \cs_new_protected:Npn \_hook_next_gset:nn
1118 { \_hook_code_gset_aux:nnn { _next } }
1119 \cs_new_protected:Npn \_hook_code_gset_aux:nnn #1 #2 #3

```



```

1120 {
1121   \cs_gset:cpn { __hook#1~#2 \exp_last_unbraced:Ne }
1122   { \__hook_parameter:n {#2} }
1123   {#3}
1124 }
1125 \cs_generate_variant:Nn \__hook_code_gset:nn { ne }
1126 <latexrelease>\EndIncludeInRelease
1127 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_code_gset:nn}
1128 <latexrelease>{Hooks-with-args}
1129 <latexrelease>\cs_undefine:N \__hook_code_gset:nn
1130 <latexrelease>\cs_undefine:N \__hook_toplevel_gset:nn
1131 <latexrelease>\cs_undefine:N \__hook_next_gset:nn
1132 <latexrelease>\cs_undefine:N \__hook_code_gset_aux:nnn
1133 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_code_gset:nn and others.)

__hook_normalize_cs_args:nn This macro normalizes the parameters of the macros __hook<type>_hook to take the right number of arguments after a hook is declared. At this point we know \c__hook_hook_hook_parameter_tl exists, so use that to count the arguments and use that as <parameter text> for the newly (re)defined macro.

```

1134 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_normalize_cs_args:nn}
1135 <latexrelease>{Hooks-with-args}
1136 \cs_new_protected:Npn \__hook_normalize_cs_args:nn #1 #2
1137 {
1138   \cs_if_exist:cT { __hook#1~#2 }
1139   {
1140     \__hook_code_gset_auxi:eeen
1141     { \tl_use:c { c__hook_#2_parameter_tl } }
1142     {
1143       \exp_args:NNo \exp_args:No \__hook_double_hashes:n
1144       {
1145         \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1146         { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
1147       }
1148     }
1149     { }
1150     { __hook#1~#2 }
1151   }
1152 }
1153 <latexrelease>\EndIncludeInRelease
1154 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_normalize_cs_args:nn}
1155 <latexrelease>{Hooks-with-args}
1156 <latexrelease>\cs_undefine:N \__hook_normalize_cs_args:nn
1157 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_normalize_cs_args:nn.)

__hook_normalize_code_pool:n This one's a bit of a hack. It takes a hook, and iterates over its code pool (\g__hook_hook_code_prop), redefining each code label to use only valid arguments. This is used when, for example, a code is added referencing arguments #1 and #2, but the hook has only #1. In this example, every reference to #2 is changed to ##2. This is done because

otherwise T_EX will throw a low-level error every time some change happens to the hook (code is added, a rule is set, etc), which can get quite repetitive for no good reason.

```

1158 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_normalize_code_pool:n}
1159 <latexrelease>{Hooks-with-args}
1160 \cs_new_protected:Npn \__hook_normalize_code_pool:n #1
1161 {

```

First, call `\__hook_set_normalize_fn:nn` with the hook name to set everything up, then we'll loop over the hook's code pool applying the normalization above. After that's done, copy the temporary property list back to the hook's.

```

1162   \__hook_set_normalize_fn:nn {#1} { Offending~label:~'##1' }
1163   \prop_clear:N \l__hook_work_prop
1164   \prop_map_function:cN { g__hook_#1_code_prop } \__hook_normalize_fn:nn
1165   \prop_gset_eq:cN { g__hook_#1_code_prop } \l__hook_work_prop
1166 }

```

The sole purpose of this function is to define `\__hook_normalize_fn:nn`, which will then do the correcting of the code being added to the hook.

```

1167 \cs_new_protected:Npn \__hook_set_normalize_fn:nn #1 #2
1168 {

```

To start, we define two auxiliary token lists. `\l__hook_tmpb_tl` contains:

```

    {\c__hook_hashes_tl 1}
    {\c__hook_hashes_tl 2}
    ...
    {\c__hook_hashes_tl 9}

1169   \cs_set:Npn \__hook_tmp:w ##1##2##3##4##5##6##7##8##9 { }
1170   \tl_set:Nx \l__hook_tmpb_tl
1171     { \__hook_braced_cs_parameter:n { __hook_tmp:w } }
1172   \group_begin:
1173     \__hook_tl_set:cn { c__hook_hash_tl } { \exp_not:N \c__hook_hashes_tl }
1174     \use:e
1175     {
1176       \group_end:
1177       \tl_set:Nn \exp_not:N \l__hook_tmpb_tl { \l__hook_tmpb_tl }
1178     }

```

And `\l__hook_tmpa_tl` contains:

```

    {\c__hook_hash_tl 1}
    {\c__hook_hash_tl 2}
    ...
    {\c__hook_hash_tl <n>}

```

with `<n>` being the number of arguments declared for the hook.

```

1179   \exp_last_unbraced:NNf
1180   \cs_set:Npn \__hook_tmp:w { \__hook_parameter:n {#1} } { }
1181   \tl_set:Nx \l__hook_tmpa_tl
1182     { \__hook_braced_cs_parameter:n { __hook_tmp:w } }

```

Now this function does the fun part. It is meant to be used with `\prop_map_function:NN`, taking a label name in `##1` and the code stored in that label in `##2`.

```

1183   \cs_gset_protected:Npe \__hook_normalize_fn:nn ##1 ##2
1184   {

```

Here we'll define two auxiliary macros: the first one throws an error when it detects an invalid argument reference. It piggybacks on T_EX's low-level "Illegal parameter number" error, but it defines a weirdly-named control sequence so that the error comes out nicely formatted. For example, if the label "badpkg" adds some code that references argument #3 in the hook "foo", which takes only two arguments, the error will be:

```
! Illegal parameter number in definition of hook 'foo'.
(hooks)          Offending label: 'badpkg'.
<to be read again>
3
```

At the point of this definition, the error is raised if the code happens to reference an invalid argument. If it was possible to detect that this definition raised no error, the next step would be unnecessary. We'll do all this in a group so this weird definition doesn't leak out, and set `\tex_escapechar:D` to -1 so this hack shows up extra nice in the case of an error.

```
1185 \group_begin:
1186 \int_set:Nn \tex_escapechar:D { -1 }
1187 \cs_set:cpn
1188 {
1189 hook~'#1'. ^^J
1190 (hooks) \prg_replicate:nn { 13 } { ~ }
1191 #2 % more message text
1192 }
1193 \exp_not:v { c__hook_#1_parameter_tl }
1194 {##2}
1195 \group_end:
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code `##2` under the label `##1` to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `\__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```
1196 \cs_set:Npn \exp_not:N \__hook_tmp:w
1197 \exp_not:V \c__hook_nine_parameters_tl
1198 {
1199 \prop_put:Nne \exp_not:N \l__hook_work_prop
1200 {##1} { \exp_not:N \__hook_double_hashes:n {##2} }
1201 }
```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code `##2` under the label `##1` to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `\__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```
1202 \exp_not:N \__hook_tmp:w
1203 \exp_not:V \l__hook_tmpa_tl
1204 \exp_args:No \exp_not:o
1205 { \exp_after:wN \__hook_tmp:w \l__hook_tmpb_tl }
1206 }
1207 }
```

```

1208 \cs_new_eq:NN \__hook_normalize_fn:nn ?
1209 \<latexrelease>\EndIncludeInRelease
1210 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_normalize_code_pool:n}
1211 \<latexrelease>{Hooks-with-args}
1212 \<latexrelease>\cs_undefine:N \__hook_normalize_code_pool:n
1213 \<latexrelease>\EndIncludeInRelease

```

Check if the expansion of a control sequence is empty by looking at its replacement text.

```

\__hook_cs_if_empty_p:c
\__hook_cs_if_empty:cTF

```

```

1214 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_cs_if_empty:c}
1215 \<latexrelease>{Hooks-with-args}
1216 \prg_new_conditional:Npnn \__hook_cs_if_empty:c #1 { p, T, F, TF }
1217 {
1218   \if:w \scan_stop: \__hook_replacement_spec:c {#1} \scan_stop:
1219   \prg_return_true:
1220   \else:
1221   \prg_return_false:
1222   \fi:
1223 }
1224 \cs_new:Npn \__hook_replacement_spec:c #1
1225 {
1226   \exp_args:Nc \token_if_macro:NT {#1}
1227   { \cs_replacement_spec:c {#1} }
1228 }
1229 \<latexrelease>\EndIncludeInRelease
1230 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_cs_if_empty:c}
1231 \<latexrelease>{Hooks-with-args}
1232 \<latexrelease>\cs_undefine:N \__hook_cs_if_empty:c
1233 \<latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_normalize_code_pool:n, __hook_set_normalize_fn:nn, and __hook_cs_if_empty:cTF.)

```

\__hook_braced_cs_parameter:n
\__hook_braced_hidden_loop:w
\__hook_cs_parameter_count:N
\__hook_cs_parameter_count:w
\__hook_cs_end:w

```

Looks at the *<parameter text>* of a control sequence, and returns a run of “hidden” braced parameters for that macro. This works as long as the macros take a simple run of zero to nine arguments. The parameters are “hidden” because the parameter tokens are returned inside \c__hook_hash_tl instead of explicitly, so that __hook_double_hashes:n won’t touch these.

```

1234 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_braced_cs_parameter:n}
1235 \<latexrelease>{Hooks-with-args}
1236 \cs_new:Npn \__hook_braced_cs_parameter:n #1
1237 {
1238   \exp_last_unbraced:Ne \__hook_braced_hidden_loop:w
1239   { \exp_args:Nc \__hook_cs_parameter_count:N {#1} } ? \s__hook_mark
1240 }
1241 \cs_new:Npn \__hook_braced_hidden_loop:w #1
1242 {
1243   \if:w ? #1
1244   \__hook_use_i_delimit_by_s_mark:nw
1245   \fi:
1246   { \exp_not:N \c__hook_hash_tl #1 }
1247   \__hook_braced_hidden_loop:w
1248 }

```

```

1249 \cs_new:Npn \__hook_cs_parameter_count:N #1
1250 {
1251   \exp_last_unbraced:Nf \__hook_cs_parameter_count:w
1252   { \token_if_macro:NT #1 { \cs_parameter_spec:N #1 } }
1253   ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1254   ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1255   ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1256   \s__hook_mark
1257 }
1258 \cs_new:Npn \__hook_cs_parameter_count:w #1#2 #3#4 #5#6 #7#8
1259 { #2 #4 #6 #8 \__hook_cs_parameter_count:w }
1260 \cs_new:Npn \__hook_cs_end:w #1 \s__hook_mark { }
1261 \<latexrelease>\EndIncludeInRelease

```

This function can't be undefined when rolling back because it's used at the end of this module to adequate the hook data structures to previous versions.

```

1262 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_braced_cs_parameter:n}
1263 \<latexrelease>{Hooks-with-args}
1264 \<latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_braced_cs_parameter:n and others.)

__hook_braced_parameter:n
 __hook_braced_real_loop:w

This one is used in simpler cases, where no special handling of hashes is required. This is used only inside __hook_initialize_hook_code:n, so it assumes \c__hook_{hook}_parameter_tl is defined, but should work otherwise.

```

1265 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_braced_parameter:n}
1266 \<latexrelease>{Hooks-with-args}
1267 \cs_new:Npn \__hook_braced_parameter:n #1
1268 {
1269   \if_case:w
1270     \int_eval:n
1271     { \exp_args:Nv \str_count:n { c__hook_#1_parameter_tl } / 3 }
1272     \exp_stop_f:
1273     \or: {##1}
1274     \or: {##1} {##2}
1275     \or: {##1} {##2} {##3}
1276     \or: {##1} {##2} {##3} {##4}
1277     \or: {##1} {##2} {##3} {##4} {##5}
1278     \or: {##1} {##2} {##3} {##4} {##5} {##6}
1279     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7}
1280     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8}
1281     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8} {##9}
1282     \else:
1283       \msg_expandable_error:nnn { latex2e } { should-not-happen }
1284       { Invalid-parameter-spec. }
1285     \fi:
1286 }
1287 \<latexrelease>\EndIncludeInRelease
1288 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_braced_parameter:n}
1289 \<latexrelease>{Hooks-with-args}
1290 \<latexrelease>\cs_undefine:N \__hook_braced_parameter:n
1291 \<latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_braced_parameter:n and __hook_braced_real_loop:w.)

`\__hook_parameter:n` This is just a shortcut to e- or f-expand to the `<parameter text>` of the hook.

```

1292 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_parameter:n}
1293 <latexrelease>{Hooks-with-args}
1294 \cs_new:Npn \__hook_parameter:n #1
1295 {
1296   \cs:w c__hook_
1297   \tl_if_exist:cTF { c__hook_#1_parameter_tl }
1298     { #1_parameter } { empty }
1299   _tl \cs_end:
1300 }
1301 \cs_new:Npn \__hook_generic_parameter:n #1
1302 { \__hook_generic_parameter:w #1 / / / \s__hook_mark }
1303 \cs_new:Npn \__hook_generic_parameter:w #1 / #2 / #3 / #4 \s__hook_mark
1304 {
1305   \cs_if_exist_use:cF { c__hook_parameter_#1/./#3_tl }
1306     { \c__hook_empty_tl }
1307 }
1308 <latexrelease>\EndIncludeInRelease
1309 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_parameter:n}
1310 <latexrelease>{Hooks-with-args}
1311 <latexrelease>\cs_undefine:N \__hook_parameter:n
1312 <latexrelease>\cs_undefine:N \__hook_generic_parameter:n
1313 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\__hook_parameter:n`.)

4.7 Setting rules for hooks code

`\g__hook_??_code_prop` Initially these variables simply used an empty “label” name (not two question marks).
`\__hook_??` This was a bit unfortunate, because then `l3doc` complains about `__` in the middle of a
`\g__hook_??_reversed_tl` command name when trying to typeset the documentation. However using a “normal”
`\c__hook_??_parameter_tl` name such as `default` has the disadvantage of that being not really distinguishable from
a real hook name. I now have settled for `??` which needs some gymnastics to get it into
the csname, but since this is used a lot, the code should be fast, so this is not done with
`c` expansion in the code later on.

`\__hook_?` isn’t used, but it has to be defined to trick the code into thinking that
`??` is actually a hook.

```

1314 \prop_new:c { g__hook_??_code_prop }
1315 \prop_new:c { __hook-?? }

```

Default rules are always given in normal ordering (never in reversed ordering). If
such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., **after**
becomes **before**) because those rules are applied first and then the order is reversed.

```

1316 \tl_new:c { g__hook_??_reversed_tl }

```

The parameter text for the “default” hook is empty.

```

1317 <latexrelease>\IncludeInRelease{2023/06/01}{\c__hook_??_parameter_tl}
1318 <latexrelease>{Hooks-with-args}
1319 \tl_const:cn { c__hook_??_parameter_tl } { }
1320 <latexrelease>\EndIncludeInRelease
1321 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_??_parameter_tl}
1322 <latexrelease>{Hooks-with-args}
1323 <latexrelease>\cs_undefine:c { c__hook_??_parameter_tl }
1324 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\g__hook_??_code_prop` and others.)

`\hook_gset_rule:nnnn` With `\hook_gset_rule:nnnn{<hook>}{<label1>}{<relation>}{<label2>}` a relation is defined between the two code labels for the given `<hook>`. The special hook `??` stands for *any* hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```

1325 \cs_new_protected:Npn \hook_gset_rule:nnnn #1#2#3#4
1326 {
1327   \__hook_normalize_hook_rule_args:Nnnnn \__hook_gset_rule:nnnn
1328   {#1} {#2} {#3} {#4}
1329 }
1330 <latexrelease>\IncludeInRelease{2022/06/01}{\__hook_gset_rule:nnnn}
1331 <latexrelease>      {Refuse-setting-rule-for-one-time-hooks}
1332 \cs_new_protected:Npn \__hook_gset_rule:nnnn #1#2#3#4
1333 {
1334   \__hook_if_deprecated_generic:nT {#1}
1335   {
1336     \__hook_deprecated_generic_warn:n {#1}
1337     \__hook_do_deprecated_generic:Nn \__hook_gset_rule:nnnn {#1}
1338     {#2} {#3} {#4}
1339     \__hook_use_none_delimit_by_s_mark:w
1340   }
1341   \__hook_if_execute_immediately:nT {#1}
1342   {
1343     \msg_error:nnnnnn { hooks } { rule-too-late }
1344     {#1} {#2} {#3} {#4}
1345     \__hook_use_none_delimit_by_s_mark:w
1346   }

```

First we ensure the basic data structure of the hook exists:

```

1347   \__hook_init_structure:n {#1}

```

Then we clear any previous relationship between both labels.

```

1348   \__hook_rule_gclear:nnn {#1} {#2} {#4}

```

Then we call the function to handle the given rule. Throw an error if the rule is invalid.

```

1349   \cs_if_exist_use:cTF { __hook_rule_#3_gset:nnn }
1350   {
1351     {#1} {#2} {#4}
1352     \__hook_update_hook_code:n {#1}
1353   }
1354   {
1355     \msg_error:nnnnnn { hooks } { unknown-rule }
1356     {#1} {#2} {#3} {#4}
1357   }
1358   \s__hook_mark
1359 }
1360 <latexrelease>\EndIncludeInRelease
1361 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_gset_rule:nnnn}
1362 <latexrelease>      {Refuse-setting-rule-for-one-time-hooks}
1363 <latexrelease>\cs_new_protected:Npn \__hook_gset_rule:nnnn #1#2#3#4
1364 <latexrelease> {
1365 <latexrelease>   \__hook_if_deprecated_generic:nT {#1}

```

```

1366 <latexrelease>      {
1367 <latexrelease>      \_hook_deprecated_generic_warn:n {#1}
1368 <latexrelease>      \_hook_do_deprecated_generic:Nn \_hook_gset_rule:nnnn
1369 <latexrelease>      {#1} {#2} {#3} {#4}
1370 <latexrelease>      \exp_after:wN \use_none:nnnnnnnnn \use_none:n
1371 <latexrelease>      }
1372 <latexrelease>      \_hook_init_structure:n {#1}
1373 <latexrelease>      \_hook_rule_gclear:nnn {#1} {#2} {#4}
1374 <latexrelease>      \cs_if_exist_use:cTF { \_hook_rule_#3_gset:nnn }
1375 <latexrelease>      {
1376 <latexrelease>          {#1} {#2} {#4}
1377 <latexrelease>          \_hook_update_hook_code:n {#1}
1378 <latexrelease>      }
1379 <latexrelease>      {
1380 <latexrelease>          \msg_error:nnnnnn { hooks } { unknown-rule }
1381 <latexrelease>          {#1} {#2} {#3} {#4}
1382 <latexrelease>      }
1383 <latexrelease>  }
1384 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_gset_rule:nnnn` and `\_hook_gset_rule:nnnn`. This function is documented on page 232.)

`\_hook_rule_before_gset:nnn` Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B , the rule $l_A > l_B$ is the same as $l_B < l_A$ only presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using `\(pdf)strcmp` to lexically sort labels l_A and l_B to a fixed order. This order is then enforced every time these two labels are used together.

Here we use `\_hook_label_pair:nn {hook} {l_A} {l_B}` to build a string $l_B|l_A$ with a fixed order, and use `\_hook_label_ordered:nnTF` to apply the correct rule to the pair of labels, depending if it was sorted or not.

```

1385 \cs_new_protected:Npn \_hook_rule_before_gset:nnn #1#2#3
1386 {
1387   \_hook_tl_gset:ce
1388   { g_hook_#1_rule_ \_hook_label_pair:nn {#2} {#3} _tl }
1389   { \_hook_label_ordered:nnTF {#2} {#3} { < } { > } }
1390 }
1391 \cs_new_eq:cN { \_hook_rule_<_gset:nnn } \_hook_rule_before_gset:nnn
1392 \cs_new_protected:Npn \_hook_rule_after_gset:nnn #1#2#3
1393 {
1394   \_hook_tl_gset:ce
1395   { g_hook_#1_rule_ \_hook_label_pair:nn {#3} {#2} _tl }
1396   { \_hook_label_ordered:nnTF {#3} {#2} { < } { > } }
1397 }
1398 \cs_new_eq:cN { \_hook_rule_>_gset:nnn } \_hook_rule_after_gset:nnn

```

(End of definition for `\_hook_rule_before_gset:nnn` and others.)

`\_hook_rule_voids_gset:nnn` This rule removes (clears, actually) the code from label #3 if label #2 is in the hook #1.

```

1399 \cs_new_protected:Npn \_hook_rule_voids_gset:nnn #1#2#3
1400 {

```



```

1401 \__hook_tl_gset:ce
1402 { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1403 { \__hook_label_ordered:nnTF {#2} {#3} { -> } { <- } }
1404 }

```

(End of definition for __hook_rule_voids_gset:nnn.)

These relations make an error/warning if labels #2 and #3 appear together in hook #1.

```

1405 \cs_new_protected:cpn { \__hook_rule_incompatible-error_gset:nnn } #1#2#3
1406 { \__hook_tl_gset:cn
1407   { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1408   { xE }
1409 }
1410 \cs_new_protected:cpn { \__hook_rule_incompatible-warning_gset:nnn } #1#2#3
1411 { \__hook_tl_gset:cn
1412   { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1413   { xW }
1414 }

```

(End of definition for __hook_rule_incompatible-error_gset:nnn and
__hook_rule_incompatible-warning_gset:nnn.)

Undo a setting. __hook_rule_unrelated_gset:nnn doesn't need to do anything, since we use __hook_rule_gclear:nnn before setting any rule.

```

1415 \cs_new_protected:Npn \__hook_rule_unrelated_gset:nnn #1#2#3 { }
1416 \cs_new_protected:Npn \__hook_rule_gclear:nnn #1#2#3
1417 { \cs_undefine:c { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl } }

```

(End of definition for __hook_rule_unrelated_gset:nnn and __hook_rule_gclear:nnn.)

Ensure that the lexically greater label comes first.

```

1418 \cs_new:Npn \__hook_label_pair:nn #1#2
1419 {
1420   \if_case:w \__hook_str_compare:nn {#1} {#2} \exp_stop_f:
1421     #1 | #1 % 0
1422   \or:   #1 | #2 % +1
1423   \else: #2 | #1 % -1
1424   \fi:
1425 }

```

(End of definition for __hook_label_pair:nn.)

Check that labels #1 and #2 are in the correct order (as returned by __hook_label_pair:nn) and if so return true, else return false.

```

1426 \prg_new_conditional:Npnn \__hook_label_ordered:nn #1#2 { TF }
1427 {
1428   \if_int_compare:w \__hook_str_compare:nn {#1} {#2} > 0 \exp_stop_f:
1429   \prg_return_true:
1430   \else:
1431   \prg_return_false:
1432   \fi:
1433 }

```

(End of definition for __hook_label_ordered:nnTF.)

`\__hook_if_label_case:nnnnn` To avoid doing the string comparison twice in `\__hook_initialize_single:NNn` (once with `\str_if_eq:nn` and again with `\__hook_label_ordered:nn`), we use a three-way branching macro that will compare #1 and #2 and expand to `\use_i:nnn` if they are equal, `\use_ii:nn` if #1 is lexically greater, and `\use_iii:nn` otherwise.

```

1434 \cs_new:Npn \__hook_if_label_case:nnnnn #1#2
1435 {
1436   \cs:w use_
1437   \if_case:w \__hook_str_compare:nn {#1} {#2}
1438     i \or: ii \else: iii \fi: :nnn
1439   \cs_end:
1440 }

```

(End of definition for `\__hook_if_label_case:nnnnn`.)

`\__hook_update_hook_code:n` Before `\begin{document}` this does nothing, in the body it reinitializes the hook code using the altered data.

```

1441 \cs_new_eq:NN \__hook_update_hook_code:n \use_none:n

```

(End of definition for `\__hook_update_hook_code:n`.)

`\__hook_initialize_all:` Initialize all known hooks (at `\begin{document}`), i.e., update the fast execution token lists to hold the necessary code in the right order.

```

1442 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_initialize_all:}
1443 <latexrelease> {Hooks-with-args}
1444 \cs_new_protected:Npn \__hook_initialize_all:
1445 {

```

First we change `\__hook_update_hook_code:n` which so far was a no-op to now initialize one hook. This way any later updates to the hook will run that code and also update the execution token list.

```

1446   \cs_gset_eq:NN \__hook_update_hook_code:n \__hook_initialize_hook_code:n

```

Now we loop over all hooks that have been defined and update each of them. Here we have to determine if the hook has arguments so that auxiliaries know what to do with hashes. We look at `\c__hook_<hook>_parameter_tl`, if it has any parameters, and set `replacing_args` accordingly.

```

1447   \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
1448   \seq_map_inline:Nn \g__hook_all_seq
1449   {
1450     \tl_if_empty:cTF { c__hook_##1_parameter_tl }
1451     { \__hook_replacing_args_false: }
1452     { \__hook_replacing_args_true: }
1453     \__hook_update_hook_code:n {##1}
1454     \__hook_replacing_args_reset:
1455   }

```

If we are debugging we show results hook by hook for all hooks that have data.

```

1456   \__hook_debug:n
1457   {
1458     \iow_term:e { ^^J[lthooks]~ All-initialized~(non-empty)~hooks: }
1459     \prop_map_inline:Nn \g__hook_used_prop
1460     {
1461       \iow_term:e
1462       { ^^J ~ ##1 ~ -> ~ \cs_replacement_spec:c { __hook-##1 } ~ }
1463     }
1464   }

```

After all hooks are initialized we change the “use” to just call the hook code and not initialize it (as this was already done in the preamble.

```

1465 \_hook_post_initialization_defs:
1466 }

1467 <latexrelease>\EndIncludeInRelease
1468 <latexrelease>\IncludeInRelease{2020/10/01}{\_hook_initialize_all:}
1469 <latexrelease> {Hooks-with-args}
1470 <latexrelease>\cs_gset_protected:Npn \_hook_initialize_all:
1471 <latexrelease> {
1472 <latexrelease> \cs_gset_eq:NN \_hook_update_hook_code:n
1473 <latexrelease> \_hook_initialize_hook_code:n
1474 <latexrelease> \_hook_debug:n { \prop_gc_clear:N \g__hook_used_prop }
1475 <latexrelease> \seq_map_inline:Nn \g__hook_all_seq
1476 <latexrelease> { \_hook_update_hook_code:n {##1} }
1477 <latexrelease> \_hook_debug:n
1478 <latexrelease> {
1479 <latexrelease> \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}
1480 <latexrelease> \prop_map_inline:Nn \g__hook_used_prop
1481 <latexrelease> {
1482 <latexrelease> \iow_term:x
1483 <latexrelease> { ^^J ~ ##1 ~ -> ~
1484 <latexrelease> \cs_replacement_spec:c { \_hook-##1 } ~ }
1485 <latexrelease> }
1486 <latexrelease> }
1487 <latexrelease> \cs_gset_eq:NN \hook_use:n \_hook_use_initialized:n
1488 <latexrelease> \cs_gset_eq:NN \_hook_preamble_hook:n \use_none:n
1489 <latexrelease> }
1490 <@@=
1491 <latexrelease>\cs_gset_eq:NN \@expl@@@initialize@all@@
1492 <latexrelease> \_hook_initialize_all:
1493 <@@=hook>
1494 <latexrelease>\EndIncludeInRelease

```

(End of definition for _hook_initialize_all:.)

_hook_initialize_hook_code:n

Initializing or reinitializing the fast execution hook code. In the preamble this is selectively done in case a hook gets used and at \begin{document} this is done for all hooks and afterwards only if the hook code changes.

```

1495 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_initialize_hook_code:n}
1496 <latexrelease> {Hooks-with-args}
1497 \cs_new_protected:Npn \_hook_initialize_hook_code:n #1
1498 {
1499 \_hook_debug:n
1500 { \iow_term:e { ^^J[lthooks]~ Update-code-for-hook~'##1' \on@line :^^J } }

```

This does the sorting and the updates. First thing we do is to check if a legacy hook macro exists and if so we add it to the hook under the label `legacy`. This might make the hook non-empty so we have to do this before the then following test.

```

1501 \_hook_include_legacy_code_chunk:n {##1}

```

If there aren't any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update _hook_hook to hold the top-level and next code chunks. If there are code chunks we call _hook_initialize_single:NNn and pass to it ready made csnames as they are

needed several times inside. This way we save a bit on processing time if we do that up front.

```

1502   \__hook_if_usable:nT {#1}
1503   {
1504     \prop_if_empty:cTF { g__hook_#1_code_prop }
1505     {
1506       \__hook_code_gset:ne {#1}
1507       {

```

The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1508         \exp_not:c { __hook_toplevel~#1 }
1509         \__hook_braced_parameter:n {#1}
1510         \exp_not:c { __hook_next~#1 }
1511         \__hook_braced_parameter:n {#1}
1512       }
1513     }
1514   {

```

By default the algorithm sorts the code chunks and then saves the result in a token list for fast execution; this is done by adding the code chunks one after another, using `\tl_gput_right:NV`. When we sort code for a reversed hook, all we have to do is to add the code chunks in the opposite order into the token list. So all we have to do in preparation is to change two definitions that are used later on.

```

1515   \__hook_if_reversed:nTF {#1}
1516   { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_left:Nn
1517     \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV }
1518   { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_right:Nn
1519     \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_right:NV }

```

When sorting, some relations (namely voids) need to act destructively on the code property lists to remove code that shouldn't appear in the sorted hook token list, so we make a copy of the code property list that we can safely work on without changing the main one.

```

1520   \prop_set_eq:Nc \l__hook_work_prop { g__hook_#1_code_prop }
1521   \__hook_initialize_single:ccn
1522   { __hook~#1 } { g__hook_#1_labels_clist } {#1}

```

For debug display we want to keep track of those hooks that actually got code added to them, so we record that in `plist`. We use a `plist` to ensure that we record each hook name only once, i.e., we are only interested in storing the keys and the value is arbitrary.

```

1523   \__hook_debug:n
1524   { \exp_args:NNe \prop_gput:Nnn \g__hook_used_prop {#1} { } }
1525   }
1526 }
1527 }
1528 <latexrelease>\EndIncludeInRelease
1529 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_initialize_hook_code:n}
1530 <latexrelease>{Hooks-with-args}
1531 <latexrelease>\cs_gset_protected:Npn \__hook_initialize_hook_code:n #1
1532 <latexrelease>{
1533   \__hook_debug:n
1534   { \iow_term:x { ^^J Update-code-for-hook~'#1'
1535     \on@line :^^J } }

```

```

1536 <latexrelease> \_hook_include_legacy_code_chunk:n {#1}
1537 <latexrelease> \_hook_if_usable:nT {#1}
1538 <latexrelease> {
1539 <latexrelease> \prop_if_empty:cTF { g__hook_#1_code_prop }
1540 <latexrelease> {
1541 <latexrelease> \_hook_tl_gset:co { __hook~#1 }
1542 <latexrelease> {
1543 <latexrelease> \cs:w __hook_toplevel~#1 \exp_after:wN \cs_end:
1544 <latexrelease> \cs:w __hook_next~#1 \cs_end:
1545 <latexrelease> }
1546 <latexrelease> }
1547 <latexrelease> {
1548 <latexrelease> \_hook_if_reversed:nTF {#1}
1549 <latexrelease> { \cs_set_eq:NN \_hook_tl_gput:Nn
1550 <latexrelease> \_hook_tl_gput_left:Nn
1551 <latexrelease> \cs_set_eq:NN \_hook_clist_gput:NV
1552 <latexrelease> \clist_gput_left:NV }
1553 <latexrelease> { \cs_set_eq:NN \_hook_tl_gput:Nn
1554 <latexrelease> \_hook_tl_gput_right:Nn
1555 <latexrelease> \cs_set_eq:NN \_hook_clist_gput:NV
1556 <latexrelease> \clist_gput_right:NV }
1557 <latexrelease> \prop_set_eq:Nc \l__hook_work_prop
1558 <latexrelease> { g__hook_#1_code_prop }
1559 <latexrelease> \_hook_initialize_single:ccn
1560 <latexrelease> { __hook~#1 } { g__hook_#1_labels_clist } {#1}
1561 <latexrelease> \_hook_debug:n
1562 <latexrelease> { \exp_args:NNx \prop_gput:Nnn \g__hook_used_prop
1563 <latexrelease> {#1} { } }
1564 <latexrelease> }
1565 <latexrelease> }
1566 <latexrelease> }
1567 <latexrelease> \EndIncludeInRelease

```

(End of definition for _hook_initialize_hook_code:n.)

_hook_tl_csname:n It is faster to pass a single token and expand it when necessary than to pass a bunch of
_hook_seq_csname:n character tokens around.

FMi: note to myself: verify

```

1568 \cs_new:Npn \_hook_tl_csname:n #1 { l__hook_label_#1_tl }
1569 \cs_new:Npn \_hook_seq_csname:n #1 { l__hook_label_#1_seq }

```

(End of definition for _hook_tl_csname:n and _hook_seq_csname:n.)

\l__hook_labels_seq For the sorting I am basically implementing Knuth's algorithm for topological sorting as
given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local
variables:

```

\l__hook_labels_int
\l__hook_front_tl
\l__hook_rear_tl
\l__hook_label_0_tl

```

- List of labels used in the current hook to label code chunks:

```

1570 \seq_new:N \l__hook_labels_seq

```

- Number of labels used in the current hook. In Knuth's algorithm this is called N :

```

1571 \int_new:N \l__hook_labels_int

```

- The sorted code list to be build is managed using two pointers one to the front of the queue and one to the rear. We model this using token list pointers. Knuth calls them F and R :

```
1572 \tl_new:N \l__hook_front_tl
1573 \tl_new:N \l__hook_rear_tl
```

- The data for the start of the queue is kept in this token list, it corresponds to what Don calls `QLINK[0]` but since we aren't manipulating individual words in memory it is slightly differently done:

```
1574 \tl_new:c { \__hook_tl_csname:n { 0 } }
```

(End of definition for `\l__hook_labels_seq` and others.)

`\__hook_initialize_single:NNn` implements the sorting of the code chunks for a hook and saves the result in the token list for fast execution (#4). The arguments are `<hook-code-plist>`, `<hook-code-tl>`, `<hook-top-level-code-tl>`, `<hook-next-code-tl>`, `<hook-ordered-labels-clist>` and `<hook-name>` (the latter is only used for debugging—the `<hook-rule-plist>` is accessed using the `<hook-name>`).

The additional complexity compared to Don's algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don's data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can't be equal to the number 0!

FMi: Needs checking for, just in case ... maybe

```
1575 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_initialize_single:NNn}
1576 <latexrelease> {Hooks-with-args}
1577 \cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
1578 {
```

Step T1: Initialize the data structure ...

```
1579 \seq_clear:N \l__hook_labels_seq
1580 \int_zero:N \l__hook_labels_int
```

Store the name of the hook:

```
1581 \tl_set:Nn \l__hook_cur_hook_tl {#3}
```

We loop over the property list holding the code and record all the labels listed there. Only the rules for those labels are of interest to us. While we are at it we count them (which gives us the N in Knuth's algorithm). The prefix `label_` is added to the variables to ensure that labels named `front`, `rear`, `labels`, or `return` don't interact with our code.

```
1582 \prop_map_inline:Nn \l__hook_work_prop
1583 {
1584     \int_incr:N \l__hook_labels_int
1585     \seq_put_right:Nn \l__hook_labels_seq {##1}
1586     \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
1587     \seq_clear_new:c { \__hook_seq_csname:n {##1} }
1588 }
```

Steps T2 and T3: Here we sort the relevant rules into the data structure...

This loop constitutes a square matrix of the labels in `\l__hook_work_prop` in the vertical and the horizontal directions. However, since the rule $l_A\langle rel \rangle l_B$ is the same as $l_B\langle rel \rangle^{-1}l_A$ we can cut the loop short at the diagonal of the matrix (*i.e.*, when both labels are equal), saving a good amount of time. The way the rules were set up (see the implementation of `\__hook_rule_before_gset:nnn` above) ensures that we have no rule in the ignored side of the matrix, and all rules are seen. The rules are applied in `\__hook_apply_label_pair:nnn`, which takes the properly-ordered pair of labels as argument.

```

1589   \prop_map_inline:Nn \l__hook_work_prop
1590   {
1591     \prop_map_inline:Nn \l__hook_work_prop
1592     {
1593       \__hook_if_label_case:nnnnn {##1} {####1}
1594       { \prop_map_break: }
1595       { \__hook_apply_label_pair:nnn {##1} {####1} }
1596       { \__hook_apply_label_pair:nnn {####1} {##1} }
1597       {#3}
1598     }
1599   }

```

Now take a breath, and look at the data structures that have been set up:

```

1600   \__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }

```

Step T4:

```

1601   \tl_set:Nn \l__hook_rear_tl { 0 }
1602   \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
1603   \seq_map_inline:Nn \l__hook_labels_seq
1604   {
1605     \int_compare:nNnT { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1606     {
1607       \tl_set:cn { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
1608       \tl_set:Nn \l__hook_rear_tl {##1}
1609     }
1610   }
1611   \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
1612   \__hook_tl_gclear:N #1
1613   \clist_gclear:N #2

```

The whole loop gets combined in steps T5–T7:

```

1614   \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
1615   {

```

This part is step T5:

```

1616     \int_decr:N \l__hook_labels_int
1617     \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
1618     \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
1619     \__hook_clist_gput:NV #2 \l__hook_front_tl
1620     \__hook_debug:n{ \iow_term:e{[lthooks]~ Handled~ code~ for~ \l__hook_front_tl} }

```

This is step T6, except that we don't use a pointer P to move through the successors, but instead use `##1` of the mapping function.

```

1621     \seq_map_inline:cn { \__hook_seq_csname:n { \l__hook_front_tl } }

```

```

1622     {
1623       \tl_set:ce { \__hook_tl_csname:n {##1} }
1624       { \int_eval:n
1625         { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
1626       }
1627       \int_compare:nNnT
1628         { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1629       {
1630         \tl_set:cn
1631           { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
1632         \tl_set:Nn \l__hook_rear_tl {##1}
1633       }
1634     }

```

and here is step T7:

```

1635     \tl_set_eq:Nc \l__hook_front_tl
1636     { \__hook_tl_csname:n { \l__hook_front_tl } }

```

This is step T8: If we haven't moved the code for all labels (i.e., if `\l__hook_labels_int` is still greater than zero) we have a loop and our partial order can't be flattened out.

```

1637   }
1638   \int_compare:nNnF \l__hook_labels_int = 0
1639   {
1640     \iow_term:e{=====}
1641     \iow_term:e{Error:~ label~ rules~ are~ incompatible:}

```

This is not really the information one needs in the error case but it will do for now
...

FMi: improve output on a rainy day

```

1642     \__hook_debug_label_data:N \l__hook_work_prop
1643     \iow_term:e{=====}
1644   }

```

After we have added all hook code to #1, we finish it off by adding extra code for the **top-level** (#2) and for one time execution (#3). These should normally be empty. The **top-level** code is added with `\__hook_tl_gput:Nn` as that might change for a reversed hook (then **top-level** is the very first code chunk added). The **next** code is always added last (to the right). The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1645     \exp_args:NNe \__hook_tl_gput:Nn #1
1646     { \exp_not:c { \__hook_toplevel~#3 } \__hook_braced_parameter:n {#3} }
1647     \__hook_tl_gput_right:Ne #1
1648     { \exp_not:c { \__hook_next~#3 } \__hook_braced_parameter:n {#3} }
1649     \use:e
1650     {
1651       \cs_gset:cpn { \__hook~#3 } \use:c { c__hook_#3_parameter_tl }
1652       { \exp_not:V #1 }
1653     }
1654   }

1655   \cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }
1656   \latexrelease\EndIncludeInRelease

```



```

1657 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_initialize_single:Nn}
1658 <latexrelease> {Hooks~with~args}
1659 <latexrelease> \cs_new_protected:Npn \__hook_initialize_single:Nn #1#2#3
1660 <latexrelease> {
1661 <latexrelease> \seq_clear:N \l__hook_labels_seq
1662 <latexrelease> \int_zero:N \l__hook_labels_int
1663 <latexrelease> \tl_set:Nn \l__hook_cur_hook_tl {#3}
1664 <latexrelease> \prop_map_inline:Nn \l__hook_work_prop
1665 <latexrelease> {
1666 <latexrelease> \int_incr:N \l__hook_labels_int
1667 <latexrelease> \seq_put_right:Nn \l__hook_labels_seq {##1}
1668 <latexrelease> \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
1669 <latexrelease> \seq_clear_new:c { \__hook_seq_csname:n {##1} }
1670 <latexrelease> }
1671 <latexrelease> \prop_map_inline:Nn \l__hook_work_prop
1672 <latexrelease> {
1673 <latexrelease> \prop_map_inline:Nn \l__hook_work_prop
1674 <latexrelease> {
1675 <latexrelease> \__hook_if_label_case:nnnnn {##1} {####1}
1676 <latexrelease> { \prop_map_break: }
1677 <latexrelease> { \__hook_apply_label_pair:nnn {##1} {####1} }
1678 <latexrelease> { \__hook_apply_label_pair:nnn {####1} {##1} }
1679 <latexrelease> {#3}
1680 <latexrelease> }
1681 <latexrelease> }
1682 <latexrelease> \__hook_debug:n
1683 <latexrelease> { \__hook_debug_label_data:N \l__hook_work_prop }
1684 <latexrelease> \tl_set:Nn \l__hook_rear_tl { 0 }
1685 <latexrelease> \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
1686 <latexrelease> \seq_map_inline:Nn \l__hook_labels_seq
1687 <latexrelease> {
1688 <latexrelease> \int_compare:nNnT
1689 <latexrelease> { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1690 <latexrelease> {
1691 <latexrelease> \tl_set:cn { \__hook_tl_csname:n
1692 <latexrelease> { \l__hook_rear_tl } } {##1}
1693 <latexrelease> \tl_set:Nn \l__hook_rear_tl {##1}
1694 <latexrelease> }
1695 <latexrelease> }
1696 <latexrelease> \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
1697 <latexrelease> \__hook_tl_gclear:N #1
1698 <latexrelease> \clist_gclear:N #2
1699 <latexrelease> \bool_while_do:nn
1700 <latexrelease> { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
1701 <latexrelease> {
1702 <latexrelease> \int_decr:N \l__hook_labels_int
1703 <latexrelease> \prop_get:NVN \l__hook_work_prop
1704 <latexrelease> \l__hook_front_tl \l__hook_return_tl
1705 <latexrelease> \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
1706 <latexrelease> \__hook_clist_gput:NV #2 \l__hook_front_tl
1707 <latexrelease> \__hook_debug:n{ \iow_term:x
1708 <latexrelease> {Handled~ code~ for~ \l__hook_front_tl} }
1709 <latexrelease> \seq_map_inline:cn
1710 <latexrelease> { \__hook_seq_csname:n { \l__hook_front_tl } }

```

```

1711 <latexrelease>      {
1712 <latexrelease>      \tl_set:cx { \__hook_tl_csname:n {##1} }
1713 <latexrelease>      { \int_eval:n
1714 <latexrelease>      { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
1715 <latexrelease>      }
1716 <latexrelease>      \int_compare:nNnT
1717 <latexrelease>      { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1718 <latexrelease>      {
1719 <latexrelease>      \tl_set:cn { \__hook_tl_csname:n
1720 <latexrelease>      { \l__hook_rear_tl } } {##1}
1721 <latexrelease>      \tl_set:Nn \l__hook_rear_tl {##1}
1722 <latexrelease>      }
1723 <latexrelease>      }
1724 <latexrelease>      \tl_set_eq:Nc \l__hook_front_tl
1725 <latexrelease>      { \__hook_tl_csname:n { \l__hook_front_tl } }
1726 <latexrelease>      }
1727 <latexrelease>      \int_compare:nNnF \l__hook_labels_int = 0
1728 <latexrelease>      {
1729 <latexrelease>      \iow_term:x{=====}
1730 <latexrelease>      \iow_term:x{Error:~ label~ rules~ are~ incompatible:}
1731 <latexrelease>      \__hook_debug_label_data:N \l__hook_work_prop
1732 <latexrelease>      \iow_term:x{=====}
1733 <latexrelease>      }
1734 <latexrelease>      \exp_args:NNo \__hook_tl_gput:Nn #1
1735 <latexrelease>      { \cs:w \__hook_toplevel~#3 \cs_end: }
1736 <latexrelease>      \__hook_tl_gput_right:No #1 { \cs:w \__hook_next~#3 \cs_end: }
1737 <latexrelease>      }
1738 <latexrelease>      \cs_generate_variant:Nn \__hook_tl_gput_right:Nn { No }
1739 <latexrelease>      \EndIncludeInRelease

```

(End of definition for __hook_initialize_single:Nn.)

__hook_tl_gput:Nn These append either on the right (normal hook) or on the left (reversed hook). This is
 __hook_clist_gput:NV setup up in __hook_initialize_hook_code:n, elsewhere their behavior is undefined.

```

1740 \cs_new:Npn \__hook_tl_gput:Nn { \ERROR }
1741 \cs_new:Npn \__hook_clist_gput:NV { \ERROR }

```

(End of definition for __hook_tl_gput:Nn and __hook_clist_gput:NV.)

__hook_apply_label_pair:nnn This is the payload of steps T2 and T3 executed in the loop described above. This macro
 __hook_label_if_exist_apply:nnnF assumes #1 and #2 are ordered, which means that any rule pertaining the pair #1 and #2
 is \g__hook_<hook>_rule_#1|#2_tl, and not \g__hook_<hook>_rule_#2|#1_tl. This
 also saves a great deal of time since we only need to check the order of the labels once.

The arguments here are <label1>, <label2>, <hook>, and <hook-code-plist>. We
 are about to apply the next rule and enter it into the data structure. __hook_apply_-
 label_pair:nnn will just call __hook_label_if_exist_apply:nnnF for the <hook>,
 and if no rule is found, also try the <hook> name ?? denoting a default hook rule.

__hook_label_if_exist_apply:nnnF will check if the rule exists for the given
 hook, and if so call __hook_apply_rule:nnn.

```

1742 \cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
1743 {

```

Extra complication: as we use default rules and local hook specific rules we first have to check if there is a local rule and if that exist use it. Otherwise check if there is a default rule and use that.

```
1744 \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
1745 {
```

If there is no hook-specific rule we check for a default one and use that if it exists.

```
1746 \__hook_label_if_exist_apply:nnnF {#1} {#2} { ?? } { }
1747 }
1748 }
```

```
1749 \cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
1750 {
```

```
1751 \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
```

What to do precisely depends on the type of rule we have encountered. If it is a `before` rule it will be handled by the algorithm but other types need to be managed differently. All this is done in `\__hook_apply_rule:nnnN`.

```
1752 \__hook_apply_rule:nnn {#1} {#2} {#3}
1753 \exp_after:wN \use_none:n
1754 \else:
1755 \use:nn
1756 \fi:
1757 }
```

(End of definition for `\__hook_apply_label_pair:nnn` and `\__hook_label_if_exist_apply:nnnF`.)

`\__hook_apply_rule:nnn` This is the code executed in steps T2 and T3 while looping through the matrix This is part of step T3. We are about to apply the next rule and enter it into the data structure. The arguments are $\langle label1 \rangle$, $\langle label2 \rangle$, $\langle hook-name \rangle$, and $\langle hook-code-plist \rangle$.

```
1758 \cs_new_protected:Npn \__hook_apply_rule:nnn #1#2#3
1759 {
1760 \cs:w __hook_apply_
1761 \cs:w g__hook_#3_reversed_tl \cs_end: rule_
1762 \cs:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end: :nnn \cs_end:
1763 {#1} {#2} {#3}
1764 }
```

(End of definition for `\__hook_apply_rule:nnn`.)

`\__hook_apply_rule_<:nnn` The most common cases are `<` and `>` so we handle that first. They are relations `<` and `>` in TAOCP, and they dictate sorting.

```
\__hook_apply_rule_>:nnn
1765 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
1766 {
1767 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1768 \tl_set:ce { \__hook_tl_csname:n {#2} }
1769 { \int_eval:n{ \cs:w \__hook_tl_csname:n {#2} \cs_end: + 1 } }
1770 \seq_put_right:cn{ \__hook_seq_csname:n {#1} }{#2}
1771 }
1772 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
1773 {
1774 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1775 \tl_set:ce { \__hook_tl_csname:n {#1} }
1776 { \int_eval:n{ \cs:w \__hook_tl_csname:n {#1} \cs_end: + 1 } }
1777 \seq_put_right:cn{ \__hook_seq_csname:n {#2} }{#1}
1778 }
```

(End of definition for `\__hook_apply_rule_<:nnn` and `\__hook_apply_rule_>:nnn`.)

`\__hook_apply_rule_xE:nnn` These relations make two labels incompatible within a hook. `xE` makes raises an error if the labels are found in the same hook, and `xW` makes it a warning.

```

1779 \cs_new_protected:cpn { __hook_apply_rule_xE:nnn } #1#2#3
1780 {
1781   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1782   \msg_error:nnnnnn { hooks } { labels-incompatible }
1783   {#1} {#2} {#3} { 1 }
1784   \use:c { __hook_apply_rule_>:nnn } {#1} {#2} {#3}
1785   \use:c { __hook_apply_rule_<:nnn } {#1} {#2} {#3}
1786 }
1787 \cs_new_protected:cpn { __hook_apply_rule_xW:nnn } #1#2#3
1788 {
1789   \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1790   \msg_warning:nnnnnn { hooks } { labels-incompatible }
1791   {#1} {#2} {#3} { 0 }
1792 }

```

(End of definition for `\__hook_apply_rule_xE:nnn` and `\__hook_apply_rule_xW:nnn`.)

`\__hook_apply_rule_>:nnn` If we see `->` we have to drop code for label `#3` and carry on. We could do a little better and drop everything for that label since it doesn't matter where we put such empty code. However that would complicate the algorithm a lot with little gain.¹² So we still unnecessarily try to sort it in and depending on the rules that might result in a loop that is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from `\l__hook_cur_hook_tl` rather than `#3` because the latter may be `??`, and the default hook doesn't store any code. Removing it instead from `\l__hook_cur_hook_tl` makes the default rules `->` and `<-` work properly.

```

1793 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
1794 {
1795   \__hook_debug:n
1796   {
1797     \__hook_msg_pair_found:nnn {#1} {#2} {#3}
1798     \iow_term:e{--->~ Drop~ '#2'~ code~ from~
1799       \iow_char:N \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
1800       because~ of~ '#1' }
1801   }
1802   \prop_put:Nnn \l__hook_work_prop {#2} { }
1803 }
1804 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
1805 {
1806   \__hook_debug:n
1807   {
1808     \__hook_msg_pair_found:nnn {#1} {#2} {#3}
1809     \iow_term:e{--->~ Drop~ '#1'~ code~ from~
1810       \iow_char:N \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
1811       because~ of~ '#2' }
1812   }
1813   \prop_put:Nnn \l__hook_work_prop {#1} { }
1814 }

```

¹²This also has the advantage that the result of the sorting doesn't change, as it might otherwise do (for unrelated chunks) if we aren't careful.

(End of definition for `\__hook_apply_rule->:nnn` and `\__hook_apply_rule<-:nnn`.)

```

\__hook_apply_rule<:nnn   Reversed rules.
\__hook_apply_rule>:nnn
\__hook_apply_rule<-:nnn 1815 \cs_new_eq:cc { __hook_apply_rule<:nnn } { __hook_apply_rule>:nnn }
\__hook_apply_rule->:nnn 1816 \cs_new_eq:cc { __hook_apply_rule>:nnn } { __hook_apply_rule<:nnn }
\__hook_apply_rule_xW:nnn 1817 \cs_new_eq:cc { __hook_apply_rule<-:nnn } { __hook_apply_rule->:nnn }
\__hook_apply_rule_xE:nnn 1818 \cs_new_eq:cc { __hook_apply_rule->:nnn } { __hook_apply_rule_xE:nnn }
\__hook_apply_rule_xE:nnn 1819 \cs_new_eq:cc { __hook_apply_rule_xE:nnn } { __hook_apply_rule_xE:nnn }
\__hook_apply_rule_xW:nnn 1820 \cs_new_eq:cc { __hook_apply_rule_xW:nnn } { __hook_apply_rule_xW:nnn }

```

(End of definition for `\__hook_apply_rule<:nnn` and others.)

```

\__hook_msg_pair_found:nnn A macro to avoid moving this many tokens around.
1821 \cs_new_protected:Npn \__hook_msg_pair_found:nnn #1#2#3
1822 {
1823   \iow_term:e{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
1824     rule~ \__hook_label_pair:nn {#1} {#2}:~
1825     \use:c { g__hook_#3_rule_ \__hook_label_pair:nn {#1} {#2} _tl } ~
1826     found}
1827 }

```

(End of definition for `\__hook_msg_pair_found:nnn`.)

```

\__hook_debug_label_data:N
1828 \cs_new_protected:Npn \__hook_debug_label_data:N #1 {
1829   \iow_term:e{Code~ labels~ for~ sorting:}
1830   \iow_term:e{~ \seq_use:Nnnn\l__hook_labels_seq {~and~}{,~}{~and~} }
1831   \iow_term:e{^^J Data~ structure~ for~ label~ rules:}
1832   \prop_map_inline:Nn #1
1833   {
1834     \iow_term:e{~ ##1~ =~ \tl_use:c{ \__hook_tl_csname:n {##1} }~ ->~
1835       \seq_use:cnnn{ \__hook_seq_csname:n {##1} }{~>~}{~>~}{~>~}
1836     }
1837   }
1838   \iow_term:e{}
1839 }

```

(End of definition for `\__hook_debug_label_data:N`.)

\hook_show:n This writes out information about the hook given in its argument onto the `.log` file and the terminal, if `\show_hook:n` is used. Internally both share the same structure, except that at the end, `\hook_show:n` triggers T_EX's prompt.

\hook_log:n

```

\__hook_log_line:e
\__hook_log_line_indent:e 1840 \cs_new_protected:Npn \hook_log:n #1
\__hook_log:nN 1841 {
1842   \cs_set_eq:NN \__hook_log_cmd:e \iow_log:e
1843   \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_log:e
1844 }
1845 \cs_if_exist:NTF \iow_show:e
1846 {
1847   \cs_new_protected:Npn \hook_show:n #1
1848   {
1849     \cs_set_eq:NN \__hook_log_cmd:e \iow_show:e
1850     \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:e
1851   }

```

```

1852 }
1853 {
1854   \cs_new_protected:Npn \hook_show:n #1
1855   {
1856     \cs_set_eq:NN \__hook_log_cmd:e \iow_term:e
1857     \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:e
1858   }
1859 }
1860 \cs_new_protected:Npn \__hook_log_line:e #1
1861 { \__hook_log_cmd:e { >~#1 } }
1862 \cs_new_protected:Npn \__hook_log_line_indent:e #1
1863 { \__hook_log_cmd:e { >~\@spaces #1 } }
1864 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_log:nN}
1865 <latexrelease>{Hooks-with-args}
1866 \cs_new_protected:Npn \__hook_log:nN #1 #2
1867 {
1868   \__hook_if_deprecated_generic:nT {#1}
1869   {
1870     \__hook_deprecated_generic_warn:n {#1}
1871     \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
1872     \exp_after:wN \use_none:nnnnnnnnn \use_none:nnnnn
1873   }
1874   \__hook_preamble_hook:n {#1}
1875   \__hook_log_cmd:e
1876   {
1877     ^^J ->~The~
1878     \__hook_if_generic:nT {#1} { generic~ }
1879     hook~'~#1'
1880     \__hook_if_disabled:nF {#1}
1881     {
1882       \exp_args:Nne \__hook_print_args:nn {#1}
1883       {
1884         \int_eval:n
1885         { \str_count:e { \__hook_parameter:n {#1} } / 3 }
1886       }
1887     }
1888   }
1889 }
1890 \__hook_if_usable:nF {#1}
1891 { \__hook_log_line:e { The~hook~is~not~declared. } }
1892 \__hook_if_disabled:nT {#1}
1893 { \__hook_log_line:e { The~hook~is~disabled. } }
1894 \hook_if_empty:nTF {#1}
1895 { #2 { The~hook~is~empty } }
1896 {
1897   \__hook_log_line:e { Code~chunks: }
1898   \bool_lazy_or:nnTF
1899   { ! \prop_if_exist_p:c { g__hook_#1_code_prop } }
1900   { \prop_if_empty_p:c { g__hook_#1_code_prop } }
1901   { \__hook_log_line_indent:e { --- } }
1902   {
1903     \prop_map_inline:cn { g__hook_#1_code_prop }
1904     {

```

```

1905         \exp_after:wN \cs_set:Npn \exp_after:wN \__hook_tmp:w
1906         \c__hook_nine_parameters_tl {##2}
1907         \__hook_log_line_indent:e
1908         { ##1~-->~\cs_replacement_spec:N \__hook_tmp:w }
1909     }
1910 }

```

If there is code in the top-level token list, print it:

```

1911 \__hook_log_line:e
1912 {
1913     Document-level~(top-level)~code
1914     \__hook_if_usable:nT {#1}
1915     { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) } :
1916 }
1917 \__hook_log_line_indent:e
1918 {
1919     \__hook_cs_if_empty:cTF { __hook_toplevel~#1 }
1920     { --- }
1921     { -> ~ \cs_replacement_spec:c { __hook_toplevel~#1 } }
1922 }
1923 \__hook_log_line:e { Extra~code~for~next~invocation: }
1924 \__hook_log_line_indent:e
1925 {
1926     \__hook_cs_if_empty:cTF { __hook_next~#1 }
1927     { --- }

```

If the token list is not empty we want to display it but without the first tokens (the code to clear itself) so we call a helper command to get rid of them.

```

1928     {
1929         -> ~ \exp_last_unbraced:Nf \__hook_log_next_code:w
1930         { \cs_replacement_spec:c { __hook_next~#1 } }
1931     }
1932 }

```

Loop through the rules in a hook and for every rule found, print it. If no rule is there, print ---. The boolean \l__hook_tmpa_bool here indicates if the hook has no rules.

```

1933 \__hook_log_line:e { Rules: }
1934 \bool_set_true:N \l__hook_tmpa_bool
1935 \__hook_list_rules:nm {#1}
1936 {
1937     \bool_set_false:N \l__hook_tmpa_bool
1938     \__hook_log_line_indent:e
1939     {
1940         ##2~ with~
1941         \str_if_eq:nnT {##3} {??} { default~ }
1942         relation~ ##1
1943     }
1944 }
1945 \bool_if:NT \l__hook_tmpa_bool
1946 { \__hook_log_line_indent:e { --- } }

```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```

1947 \bool_lazy_and:nnTF
1948 { \__hook_if_usable_p:n {#1} }
1949 { ! \hook_if_empty_p:n {#1} }
1950 {
1951 \__hook_log_line:e
1952 {
1953 Execution-order
1954 \bool_if:NTF \l__hook_tmpa_bool
1955 { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
1956 { ~(after~
1957 \__hook_if_reversed:nT {#1} { reversal~and~ }
1958 applying~rules)
1959 } :
1960 }
1961 #2 % \tl_show:n
1962 {
1963 \@spaces
1964 \clist_if_empty:cTF { g__hook_#1_labels_clist }
1965 { --- }
1966 { \clist_use:cn { g__hook_#1_labels_clist } { ,~ } }
1967 }
1968 }
1969 {
1970 \__hook_log_line:e { Execution-order: }
1971 #2
1972 {
1973 \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
1974 { code~pool~is~empty }
1975 { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }
1976 }
1977 }
1978 }
1979 }
1980 <latexrelease> \EndIncludeInRelease
1981 %
1982 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_log:nN}
1983 <latexrelease> {Hooks~with~args}
1984 <latexrelease> \cs_new_protected:Npn \__hook_log:nN #1 #2
1985 <latexrelease> {
1986 <latexrelease> \__hook_if_deprecated_generic:nT {#1}
1987 <latexrelease> {
1988 <latexrelease> \__hook_deprecated_generic_warn:n {#1}
1989 <latexrelease> \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
1990 <latexrelease> \exp_after:wN \use_none:nnnnnnnnn \use_none:nnnnn
1991 <latexrelease> }
1992 <latexrelease> \__hook_preamble_hook:n {#1}
1993 <latexrelease> \__hook_log_cmd:e
1994 <latexrelease> { ^^J ->~The~ \__hook_if_generic:nT
1995 <latexrelease> {#1} { generic~ } hook~'~#1': }
1996 <latexrelease> \__hook_if_usable:nF {#1}
1997 <latexrelease> { \__hook_log_line:e { The~hook~is~not~declared. } }
1998 <latexrelease> \__hook_if_disabled:nT {#1}
1999 <latexrelease> { \__hook_log_line:e { The~hook~is~disabled. } }
2000 <latexrelease> \hook_if_empty:nTF {#1}

```



```

2001 <latexrelease> { #2 { The~hook~is~empty } }
2002 <latexrelease> {
2003 <latexrelease>   \__hook_log_line:e { Code~chunks: }
2004 <latexrelease>   \prop_if_empty:cTF { g__hook_#1_code_prop }
2005 <latexrelease>     { \__hook_log_line_indent:e { --- } }
2006 <latexrelease>     {
2007 <latexrelease>       \prop_map_inline:cn { g__hook_#1_code_prop }
2008 <latexrelease>       { \__hook_log_line_indent:e
2009 <latexrelease>         { ##1~-->~\tl_to_str:n {##2} } }
2010 <latexrelease>     }
2011 <latexrelease>   \__hook_log_line:e
2012 <latexrelease>   {
2013 <latexrelease>     Document~level~(top~level)~code
2014 <latexrelease>     \__hook_if_usable:nT {#1}
2015 <latexrelease>     { ~(executed~
2016 <latexrelease>       \__hook_if_reversed:nTF {#1} {first} {last} ) } :
2017 <latexrelease>   }
2018 <latexrelease>   \__hook_log_line_indent:e
2019 <latexrelease>   {
2020 <latexrelease>     \tl_if_empty:cTF { __hook_toplevel~#1 }
2021 <latexrelease>     { --- }
2022 <latexrelease>     { -> ~ \exp_args:Nv \tl_to_str:n
2023 <latexrelease>       { __hook_toplevel~#1 } }
2024 <latexrelease>   }
2025 <latexrelease>   \__hook_log_line:e { Extra~code~for~next~invocation: }
2026 <latexrelease>   \__hook_log_line_indent:e
2027 <latexrelease>   {
2028 <latexrelease>     \tl_if_empty:cTF { __hook_next~#1 }
2029 <latexrelease>     { --- }
2030 <latexrelease>     { ->~ \exp_args:Nv \__hook_log_next_code:n
2031 <latexrelease>       { __hook_next~#1 } }
2032 <latexrelease>   }
2033 <latexrelease>   \__hook_log_line:e { Rules: }
2034 <latexrelease>   \bool_set_true:N \l__hook_tmpa_bool
2035 <latexrelease>   \__hook_list_rules:nn {#1}
2036 <latexrelease>   {
2037 <latexrelease>     \bool_set_false:N \l__hook_tmpa_bool
2038 <latexrelease>     \__hook_log_line_indent:e
2039 <latexrelease>     {
2040 <latexrelease>       ##2~ with~
2041 <latexrelease>       \str_if_eq:nnT {##3} {??} { default~ }
2042 <latexrelease>       relation~ ##1
2043 <latexrelease>     }
2044 <latexrelease>   }
2045 <latexrelease>   \bool_if:NT \l__hook_tmpa_bool
2046 <latexrelease>   { \__hook_log_line_indent:e { --- } }
2047 <latexrelease>   \bool_lazy_and:nnTF
2048 <latexrelease>     { \__hook_if_usable_p:n {#1} }
2049 <latexrelease>     { ! \hook_if_empty_p:n {#1} }
2050 <latexrelease>   {
2051 <latexrelease>     \__hook_log_line:e
2052 <latexrelease>     {
2053 <latexrelease>       Execution~order
2054 <latexrelease>       \bool_if:NTF \l__hook_tmpa_bool

```

```

2055 <latexrelease>          { \_hook_if_reversed:nT
2056 <latexrelease>          {#1}{ ~(after~reversal) } }
2057 <latexrelease>          { ~(after~
2058 <latexrelease>          \_hook_if_reversed:nT {#1} { reversal~and~
2059 <latexrelease>          applying~rules)
2060 <latexrelease>          } :
2061 <latexrelease>          }
2062 <latexrelease>          #2 % \tl_show:n
2063 <latexrelease>          {
2064 <latexrelease>          \@spaces
2065 <latexrelease>          \clist_if_empty:cTF { g__hook_#1_labels_clist }
2066 <latexrelease>          { --- }
2067 <latexrelease>          { \clist_use:cn
2068 <latexrelease>          { g__hook_#1_labels_clist } { ,~ } }
2069 <latexrelease>          }
2070 <latexrelease>          }
2071 <latexrelease>          {
2072 <latexrelease>          \_hook_log_line:e { Execution-order: }
2073 <latexrelease>          #2
2074 <latexrelease>          {
2075 <latexrelease>          \@spaces Not~set~because~the~hook~
2076 <latexrelease>          \_hook_if_usable:nTF {#1}
2077 <latexrelease>          { code~pool~is~empty }
2078 <latexrelease>          { is~\_hook_if_disabled:nTF
2079 <latexrelease>          {#1} {disabled} {undeclared} }
2080 <latexrelease>          }
2081 <latexrelease>          }
2082 <latexrelease>          }
2083 <latexrelease>          }
2084 <latexrelease> \EndIncludeInRelease

```

To display the code for next invocation only (i.e., from \AddToHookNext we have to remove the string _hook_clear_next:n{hook}), so the simplest is to use a macro delimited by a }₁2.

```

2085 <latexrelease> \IncludeInRelease{2023/06/01}{\_hook_log_next_code:n}
2086 <latexrelease>          {Hooks-with-args}
\_hook_log_next_code:n 2087 \exp_last_unbraced:NNNno
2088 \cs_new:Npn \_hook_log_next_code:w #1 \c_right_brace_str { }
2089 <latexrelease> \EndIncludeInRelease
2090 <latexrelease> \IncludeInRelease{2020/10/01}{\_hook_log_next_code:n}
2091 <latexrelease>          {Hooks-with-args}
2092 <latexrelease> \cs_gset:Npn \_hook_log_next_code:n #1
2093 <latexrelease> { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }
2094 <latexrelease> \EndIncludeInRelease

```

Pretty-prints the number of arguments of a hook.

```

2095 \cs_new:Npn \_hook_print_args:nn #1 #2
2096 {
2097   \int_compare:nNnT {#2} > { 0 }
2098   {
\_hook_print_args:n 2099     \_hook_if_declared:nT {#1} { \use_none:nnn }
2100     \_hook_if_cmd_hook:nT {#1}
2101     { \use_i:nnn { ~ (unknown ~ ) } }
2102     \use:n { ~ (#2 ~ ) }

```

```

2103     argument \int_compare:nNtT {#2} > { 1 } { s } )
2104   }
2105 }

```

(End of definition for `\hook_show:n` and others. These functions are documented on page 232.)

`\__hook_list_rules:nn` This macro takes a *<hook>* and an *<inline function>* and loops through each pair of *<labels>* in the *<hook>*, and if there is a relation between this pair of *<labels>*, the *<inline function>* is executed with `#1 = <relation>`, `#2 = <label1>|<label2>`, and `#3 = <hook>` (the latter may be the argument `#1` to `\__hook_list_rules:nn`, or `??` if it is a default rule).

```

2106 \cs_new_protected:Npn \__hook_list_rules:nn #1 #2
2107 {
2108   \prop_if_exist:cT { g__hook_#1_code_prop }
2109   {
2110     \cs_set_protected:Npn \__hook_tmp:w ##1 ##2 ##3 {#2}
2111     \prop_map_inline:cn { g__hook_#1_code_prop }
2112     {
2113       \prop_map_inline:cn { g__hook_#1_code_prop }
2114       {
2115         \__hook_if_label_case:nnnnn {##1} {####1}
2116         { \prop_map_break: }
2117         { \__hook_list_one_rule:nnn {##1} {####1} }
2118         { \__hook_list_one_rule:nnn {####1} {##1} }
2119         {#1}
2120       }
2121     }
2122   }
2123 }

```

These two are quite similar to `\__hook_apply_label_pair:nnn` and `\__hook_label_if_exist_apply:nnnF`, respectively, but rather than applying the rule, they pass it to the *<inline function>*.

```

2124 \cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
2125 {
2126   \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
2127   { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
2128 }
2129 \cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
2130 {
2131   \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
2132   \exp_args:Nv \__hook_tmp:w
2133   { g__hook_ #3 _rule_ #1 | #2 _tl } { #1 | #2 } {#3}
2134   \exp_after:wN \use_none:nn
2135   \fi:
2136   \use:n
2137 }

```

(End of definition for `\__hook_list_rules:nn`, `\__hook_list_one_rule:nnn`, and `\__hook_list_if_rule_exists:nnnF`.)

`\__hook_debug_print_rules:n` A shorthand for debugging that prints similar to `\prop_show:N`.

```

2138 \cs_new_protected:Npn \__hook_debug_print_rules:n #1
2139 {

```

```

2140 \iow_term:n { The~hook~#1~contains~the~rules: }
2141 \cs_set_protected:Npn \__hook_tmp:w ##1
2142 {
2143   \__hook_list_rules:nn {#1}
2144   {
2145     \iow_term:e
2146     {
2147       > ##1 {####2} ##1 => ##1 {####1}
2148       \str_if_eq:nnT {####3} {??} { ~(default) }
2149     }
2150   }
2151 }
2152 \exp_args:No \__hook_tmp:w { \use:nn { ~ } { ~ } }
2153 }

```

(End of definition for __hook_debug_print_rules:n.)

4.8 Specifying code for next invocation

`\hook_gput_next_code:nn`

```

2154 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_gput_next_code:nn}
2155 <latexrelease> {Hooks~with~args}
2156 \cs_new_protected:Npn \hook_gput_next_code:nn #1 #2
2157 {
2158   \__hook_replacing_args_false:
2159   \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
2160
2161   \__hook_debug:n{\iow_term:e{[lthooks]~ Add~ to~
2162     \__hook_if_usable:nF {#1} { undeclared~ }
2163     hook~ '#1'~ ( next~ invocation~ only )
2164     \on@line
2165     ^^J [lthooks] \@spaces
2166     <-- \tl_to_str:n{#2}
2167   }
2168   \__hook_replacing_args_reset:
2169 }
2170 \cs_new_protected:Npn \hook_gput_next_code_with_args:nn #1 #2
2171 {
2172   \__hook_replacing_args_true:
2173   \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
2174
2175   \__hook_debug:n{\iow_term:e{[lthooks]~ Add~ to~
2176     \__hook_if_usable:nF {#1} { undeclared~ }
2177     hook~ '#1'~ ( next~ invocation~ only )
2178     \on@line
2179     ^^J [lthooks] \@spaces
2180     <-- \tl_to_str:n{#2}
2181   }
2182   \__hook_replacing_args_reset:
2183 }
2184 <latexrelease>\EndIncludeInRelease
2185 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_gput_next_code:nn}

```

```

2186 <latexrelease> {Hooks-with-args}
2187 <latexrelease> \cs_gset_protected:Npn \hook_gput_next_code:nn #1
2188 <latexrelease> { \__hook_normalize_hook_args:Nn
2189 <latexrelease> \__hook_gput_next_code:nn {#1} }
2190 <latexrelease> \cs_gset_protected:Npn \hook_gput_next_code_with_args:nn #1 #2 { }
2191 <latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_gput_next_code:nn. This function is documented on page 231.)

__hook_gput_next_code:nn

```

2192 \cs_new_protected:Npn \__hook_gput_next_code:nn #1 #2
2193 {
2194   \__hook_if_disabled:nTF {#1}
2195   { \msg_error:nnn { hooks } { hook-disabled } {#1} }
2196   {
2197     \__hook_if_structure_exist:nTF {#1}
2198     { \__hook_gput_next_do:nn }
2199     { \__hook_try_declaring_generic_next_hook:nn
2200       {#1} {#2}
2201     }
2202   }

```

(End of definition for __hook_gput_next_code:nn.)

__hook_gput_next_do:nn

Start by sanity-checking with __hook_chk_args_allowed:nn. Then check if the “next code” token list is empty: if so we need to add a \tl_gclear:c to clear it, so the code lasts for one usage only. The token list is cleared early so that nested usages don’t get lost. \tl_gclear:c is used instead of \tl_gclear:N in case the hook is used in an expansion-only context, so the token list doesn’t expand before \tl_gclear:N: that would make an infinite loop. Also in case the main code token list is empty, the hook code has to be updated to add the next execution token list.

```

2203 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_gput_next_do:nn}
2204 <latexrelease> {Hooks-with-args}
2205 \cs_new_protected:Npn \__hook_gput_next_do:nn #1
2206 {
2207   \__hook_init_structure:n {#1}
2208   \__hook_chk_args_allowed:nn {#1} { AddToHookNext }
2209   \__hook_cs_if_empty:cT { __hook~#1 }
2210   { \__hook_update_hook_code:n {#1} }
2211   \__hook_cs_if_empty:cT { __hook_next~#1 }
2212   { \__hook_next_gset:nn {#1} { \__hook_clear_next:n {#1} } }
2213   \__hook_cs_gput_right:nnn { _next } {#1}
2214 }
2215 <latexrelease> \EndIncludeInRelease
2216 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_gput_next_do:nn}
2217 <latexrelease> {Hooks-with-args}
2218 <latexrelease> \cs_gset_protected:Npn \__hook_gput_next_do:nn #1
2219 <latexrelease> {
2220 <latexrelease> \exp_args:Nc \__hook_gput_next_do:Nnn
2221 <latexrelease> { __hook_next~#1 } {#1}
2222 <latexrelease> }
2223 <latexrelease> \cs_gset_protected:Npn \__hook_gput_next_do:Nnn #1 #2
2224 <latexrelease> {
2225 <latexrelease> \tl_if_empty:cT { __hook~#2 }

```

```

2226 <latexrelease> { \__hook_update_hook_code:n {#2} }
2227 <latexrelease> \tl_if_empty:NT #1
2228 <latexrelease> { \__hook_tl_gset:Nn #1 { \__hook_clear_next:n {#2} } }
2229 <latexrelease> \__hook_tl_gput_right:Nn #1
2230 <latexrelease> }
2231 <latexrelease> \EndIncludeInRelease

(End of definition for \__hook_gput_next_do:nn.)

```

\hook_gclear_next_code:n Discard anything set up for next invocation of the hook.

```

2232 \cs_new_protected:Npn \hook_gclear_next_code:n #1
2233 { \__hook_normalize_hook_args:Nn \__hook_clear_next:n {#1} }

(End of definition for \hook_gclear_next_code:n. This function is documented on page 231.)

```

__hook_clear_next:n

```

2234 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_clear_next:n}
2235 <latexrelease> {Hooks-with-args}
2236 \cs_new_protected:Npn \__hook_clear_next:n #1
2237 { \__hook_next_gset:nn {#1} { } }
2238 <latexrelease> \EndIncludeInRelease
2239 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_clear_next:n}
2240 <latexrelease> {Hooks-with-args}
2241 <latexrelease> \cs_gset_protected:Npn \__hook_clear_next:n #1
2242 <latexrelease> { \cs_gset_eq:cN { \__hook_next~#1 } \c_empty_tl }
2243 <latexrelease> \EndIncludeInRelease

(End of definition for \__hook_clear_next:n.)

```

4.9 Using the hook

\hook_use:n **\hook_use:n** as defined here is used in the preamble, where hooks aren't initialized by default. **__hook_use_initialized:n** is also defined, which is the non-**\protected** version for use within the document. Their definition is identical, except for the **__hook_preamble_hook:n** (which wouldn't hurt in the expandable version, but it would be an unnecessary extra expansion).

__hook_use_initialized:n holds the expandable definition while in the preamble. **__hook_preamble_hook:n** initializes the hook in the preamble, and is redefined to **\use_none:n** at **\begin{document}**.

Both versions do the same thing internally: they check that the hook exists as given, and if so they use it as quickly as possible.

At **\begin{document}**, all hooks are initialized, and any change in them causes an update, so **\hook_use:n** can be made expandable. This one is better not protected so that it can expand into nothing if containing no code. Also important in case of generic hooks that we do not generate a **\relax** as a side effect of checking for a csname. In contrast to the T_EX low-level **\csname ... \endcsname** construct **\tl_if_exist:c** is careful to avoid this.

```

2244 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_use:n}
2245 <latexrelease> {Hooks-with-args}
2246 \cs_new_protected:Npn \hook_use:n #1
2247 {
2248   \__hook_preamble_hook:n {#1}
2249   \__hook_use_initialized:n {#1}

```

```

2250 }
2251 \cs_new:Npn \__hook_use_initialized:n #1
2252 {
2253   \if_cs_exist:w __hook~#1 \cs_end:
2254   \cs:w __hook~#1 \use_i:nn
2255   \fi:
2256   \use_none:n
2257   \cs_end:
2258 }
2259 \cs_new_protected:Npn \__hook_preamble_hook:n #1
2260 {
2261   \if_cs_exist:w __hook~#1 \cs_end:
2262   \__hook_initialize_hook_code:n {#1}
2263   \fi:
2264 }
2265 <latexrelease> \EndIncludeInRelease

2266 <latexrelease> \IncludeInRelease{2021/11/15}{\hook_use:n}
2267 <latexrelease> {Standardize-generic-hook-names}
2268 <latexrelease> \cs_new_protected:Npn \hook_use:n #1
2269 <latexrelease> {
2270 <latexrelease>   \tl_if_exist:cT { __hook~#1 }
2271 <latexrelease>   {
2272 <latexrelease>     \__hook_preamble_hook:n {#1}
2273 <latexrelease>     \cs:w __hook~#1 \cs_end:
2274 <latexrelease>   }
2275 <latexrelease> }
2276 <latexrelease> \cs_new:Npn \__hook_use_initialized:n #1
2277 <latexrelease> {
2278 <latexrelease>   \if_cs_exist:w __hook~#1 \cs_end:
2279 <latexrelease>   \cs:w __hook~#1 \exp_after:wN \cs_end:
2280 <latexrelease>   \fi:
2281 <latexrelease> }
2282 <latexrelease> \cs_new_protected:Npn \__hook_preamble_hook:n #1
2283 <latexrelease> { \__hook_initialize_hook_code:n {#1} }
2284 <latexrelease> \cs_new:Npn \hook_use:nnw #1 { }
2285 <latexrelease> \EndIncludeInRelease

2286 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_use:n}
2287 <latexrelease> {Standardize-generic-hook-names}
2288 <latexrelease> \cs_new_protected:Npn \hook_use:n #1
2289 <latexrelease> {
2290 <latexrelease>   \tl_if_exist:cTF { __hook~#1 }
2291 <latexrelease>   {
2292 <latexrelease>     \__hook_preamble_hook:n {#1}
2293 <latexrelease>     \cs:w __hook~#1 \cs_end:
2294 <latexrelease>   }
2295 <latexrelease>   { \__hook_use:wn #1 / \s__hook_mark {#1} }
2296 <latexrelease> }
2297 <latexrelease> \cs_new:Npn \__hook_use_initialized:n #1
2298 <latexrelease> {
2299 <latexrelease>   \if_cs_exist:w __hook~#1 \cs_end:
2300 <latexrelease>   \else:
2301 <latexrelease>     \__hook_use_undefined:w
2302 <latexrelease>   \fi:

```

```

2303 <latexrelease> \cs:w __hook~#1 \__hook_use_end:
2304 <latexrelease> }
2305 <latexrelease> \cs_new:Npn \__hook_use_undefined:w
2306 <latexrelease> #1 #2 __hook~#3 \__hook_use_end:
2307 <latexrelease> {
2308 <latexrelease> #1 % fi
2309 <latexrelease> \__hook_use:wn #3 / \s__hook_mark {#3}
2310 <latexrelease> }
2311 <latexrelease> \cs_new_protected:Npn \__hook_preamble_hook:n #1
2312 <latexrelease> { \__hook_initialize_hook_code:n {#1} }
2313 <latexrelease> \cs_new_eq:NN \__hook_use_end: \cs_end:
2314 <latexrelease> \cs_new:Npn \hook_use:nnw #1 { }
2315 <latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_use:n, __hook_use_initialized:n, and __hook_preamble_hook:n. This function is documented on page 230.)

\hook_use:nnw

```

\__hook_use_initialized:nnw 2316 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_use:nnw}
2317 <latexrelease> {Hooks~with~args}
2318 \cs_new_protected:Npn \hook_use:nnw #1
2319 {
2320 \__hook_preamble_hook:n {#1}
2321 \__hook_use_initialized:nnw {#1}
2322 }
2323 \cs_new:Npn \__hook_use_initialized:nnw #1 #2
2324 {
2325 \cs:w
2326 \if_cs_exist:w __hook~#1 \cs_end:
2327 __hook~#1
2328 \else:
2329 use_none: \prg_replicate:nn {#2} { n }
2330 \fi:
2331 \cs_end:
2332 }
2333 <latexrelease> \EndIncludeInRelease
2334 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_use:nnw}
2335 <latexrelease> {Hooks~with~args}
2336 <latexrelease> \cs_gset:Npn \hook_use:nnw #1 #2
2337 <latexrelease> { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2338 <latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_use:nnw and __hook_use_initialized:nnw. This function is documented on page 230.)

__hook_post_initialization_defs:

```

2339 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_post_initialization_defs:}
2340 <latexrelease> {Hooks~with~args}
2341 \cs_new_protected:Npn \__hook_post_initialization_defs:
2342 {
2343 \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
2344 \cs_gset_eq:NN \hook_use:nnw \__hook_use_initialized:nnw
2345 \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
2346 \cs_gset_eq:NN \__hook_post_initialization_defs: \prg_do_nothing:
2347 }

```



```

2348 <latexrelease>\EndIncludeInRelease
2349 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_post_initialization_defs:}
2350 <latexrelease>{Hooks~with~args}
2351 <latexrelease>\cs_undefine:N \__hook_post_initialization_defs:
2352 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_post_initialization_defs:.)

__hook_use:wn __hook_use:wn does a quick check to test if the current hook is a file hook: those need a special treatment. If it is not, the hook does not exist. If it is, then __hook_try_file_hook:n is called, and checks that the current hook is a file-specific hook using __hook_if_file_hook:wTF. If it's not, then it's a generic file/ hook and is used if it exist.

If it is a file-specific hook, it passes through the same normalization as during declaration, and then it is used if defined. __hook_if_usable_use:n checks if the hook exist, and calls __hook_preamble_hook:n if so, then uses the hook.

```

2353 <latexrelease>\IncludeInRelease{2021/11/15}{\__hook_use:wn}
2354 <latexrelease>{Standardize-generic-hook-names}
2355 <latexrelease>\EndIncludeInRelease
2356 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use:wn}
2357 <latexrelease>{Standardize-generic-hook-names}
2358 <latexrelease>\cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
2359 <latexrelease> {
2360 <latexrelease> \str_if_eq:nnTF {#1} { file }
2361 <latexrelease> { \__hook_try_file_hook:n {#3} }
2362 <latexrelease> { } % Hook doesn't exist
2363 <latexrelease> }

2364 <latexrelease>\cs_new_protected:Npn \__hook_try_file_hook:n #1
2365 <latexrelease> {
2366 <latexrelease> \__hook_if_file_hook:wTF #1 / / \s__hook_mark
2367 <latexrelease> {
2368 <latexrelease> \exp_args:Ne \__hook_if_usable_use:n
2369 <latexrelease> { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
2370 <latexrelease> }
2371 <latexrelease> { \__hook_if_usable_use:n {#1} }
2372 <latexrelease> % file/ generic hook (e.g. file/before)
2373 <latexrelease> }

2374 <latexrelease>\cs_new_protected:Npn \__hook_if_usable_use:n #1
2375 <latexrelease> {
2376 <latexrelease> \tl_if_exist:cT { __hook~#1 }
2377 <latexrelease> {
2378 <latexrelease> \__hook_preamble_hook:n {#1}
2379 <latexrelease> \cs:w __hook~#1 \cs_end:
2380 <latexrelease> }
2381 <latexrelease> }
2382 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use:wn, __hook_try_file_hook:n, and __hook_if_usable_use:n.)

\hook_use_once:n For hooks that can and should be used only once we have a special use command that further inhibits the hook from getting more code added to it. This has the effect that any further code added to the hook is executed immediately rather than stored in the hook.

The code needs some gymnastics to prevent space trimming from the hook name, since `\hook_use:n` and `\hook_use_once:n` are documented to not trim spaces.

```

2383 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_use_once:nnw}
2384 <latexrelease> {Hooks-with-args}
2385 \cs_new_protected:Npn \hook_use_once:n #1
2386 {
2387   \__hook_if_execute_immediately:nF {#1}
2388   { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
2389     { \use:n {#1} } { 0 } }
2390 }
2391 \cs_new_protected:Npn \hook_use_once:nnw #1 #2
2392 {
2393   \__hook_if_execute_immediately:nF {#1}
2394   { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
2395     { \use:n {#1} } {#2} }
2396 }
2397 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_use_once:n` and `\hook_use_once:nnw`. These functions are documented on page 230.)

```

2398 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_use_once:nnw}
2399 <latexrelease> {Hooks-with-args}
2400 <latexrelease>\cs_gset_protected:Npn \hook_use_once:n #1
2401 <latexrelease> {
2402 <latexrelease>   \__hook_if_execute_immediately:nF {#1}
2403 <latexrelease>   { \__hook_normalize_hook_args:Nn \__hook_use_once:n
2404 <latexrelease>     { \use:n {#1} } } }
2405 <latexrelease> }
2406 <latexrelease>\cs_gset:Npn \hook_use_once:nnw #1 #2
2407 <latexrelease> { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2408 <latexrelease>\EndIncludeInRelease

```

`\__hook_use_once:nn`

```

2409 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_use_once:nn}
2410 <latexrelease> {Hooks-with-args}
2411 \cs_new_protected:Npn \__hook_use_once:nn #1 #2
2412 {
2413   \__hook_preamble_hook:n {#1}
2414   \__hook_use_once_set:n {#1}

```

When a hook has arguments, the call to `\__hook_use_initialized:n`, should be the very last thing to happen, otherwise the arguments grabbed will be wrong. So, to clean up after the hook we need to cheat a bit and sneak the cleanup code at the end of the hook, along with the next execution code.

```

2415   \__hook_replacing_args_false:
2416   \__hook_cs_gput_right:nnn { _next } {#1}
2417   { \__hook_use_once_clear:n {#1} }
2418   \__hook_replacing_args_reset:
2419   \__hook_if_usable:nTF {#1}
2420   { \__hook_use_initialized:n {#1} }
2421   {
2422     \int_compare:nNnT {#2} > { 0 }
2423     { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2424   }

```

```

2425 }
2426 <latexrelease>\EndIncludeInRelease
2427 %
2428 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use_once:nn}
2429 <latexrelease> {Hooks-with-args}
2430 <latexrelease>\cs_gset_protected:Npn \__hook_use_once:n #1
2431 <latexrelease> {
2432 <latexrelease> \__hook_preamble_hook:n {#1}
2433 <latexrelease> \__hook_use_once_set:n {#1}
2434 <latexrelease> \__hook_use_initialized:n {#1}
2435 <latexrelease> \__hook_use_once_clear:n {#1}
2436 <latexrelease> }
2437 <latexrelease>\cs_undefine:N \__hook_use_once:nn
2438 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use_once:nn.)

__hook_use_once_set:n __hook_use_once_set:n is used before the actual hook code is executed so that any usage of \AddToHook inside the hook causes the code to execute immediately. Setting \g__hook_⟨hook⟩_reversed_tl to I prevents further code from being added to the hook. __hook_use_once_clear:n then clears the hook so that any further call to \hook_use:n or \hook_use_once:n will expand to nothing.

```

2439 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_use_once_clear:n}
2440 <latexrelease> {Hooks-with-args}
2441 \cs_new_protected:Npn \__hook_use_once_set:n #1
2442 { \__hook_tl_gset:cn { g__hook_#1_reversed_tl } { I } }
2443 \cs_new_protected:Npn \__hook_use_once_clear:n #1
2444 {
2445 \__hook_code_gset:nn {#1} { }
2446 \__hook_next_gset:nn {#1} { }
2447 \__hook_toplevel_gset:nn {#1} { }
2448 \prop_gclear_new:c { g__hook_#1_code_prop }
2449 }
2450 <latexrelease>\EndIncludeInRelease
2451 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use_once_clear:n}
2452 <latexrelease> {Hooks-with-args}
2453 <latexrelease>\cs_new_protected:Npn \__hook_use_once_clear:n #1
2454 <latexrelease> {
2455 <latexrelease> \__hook_tl_gclear:c { __hook-#1 }
2456 <latexrelease> \__hook_tl_gclear:c { __hook_next-#1 }
2457 <latexrelease> \__hook_tl_gclear:c { __hook_toplevel-#1 }
2458 <latexrelease> \prop_gclear_new:c { g__hook_#1_code_prop }
2459 <latexrelease> }
2460 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use_once_set:n and __hook_use_once_clear:n.)

_hook_if_execute_immediately_p:n To check whether the code being added should be executed immediately (that is, if the hook is a one-time hook), we check if \g__hook_⟨hook⟩_reversed_tl is I. The gymnastics around \if:w is there to allow the reversed token list to be empty.

_hook_if_execute_immediately:nTF

```

2461 \prg_new_conditional:Npnn \__hook_if_execute_immediately:n #1 { T, F, TF }
2462 {
2463 \exp_after:wN \__hook_use_none_delimit_by_s_mark:w
2464 \if:w I

```

```

2465         \if_cs_exist:w g__hook_#1_reversed_tl \cs_end:
2466         \cs:w g__hook_#1_reversed_tl \exp_after:wN \cs_end:
2467         \fi:
2468         X
2469         \s__hook_mark \prg_return_true:
2470     \else:
2471         \s__hook_mark \prg_return_false:
2472     \fi:
2473 }

```

(End of definition for __hook_if_execute_immediately:nTF.)

4.10 Querying a hook

Simpler data types, like token lists, have three possible states; they can exist and be empty, exist and be non-empty, and they may not exist, in which case emptiness doesn't apply (though `\tl_if_empty:N` returns false in this case).

Hooks are a bit more complicated: they have several other states as discussed in 4.4.2. A hook may exist or not, and either way it may or may not be empty (even a hook that doesn't exist may be non-empty) or may be disabled.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool (it may happen that a package *A* defines a hook `foo`, but it's loaded after package *B*, which adds some code to that hook. In this case it is important that the code added by package *B* is remembered until package *A* is loaded).

All other states can only be queried with internal tests as the different states are irrelevant for package code.

`\hook_if_empty_p:n` Test if a hook is empty (that is, no code was added to that hook). A $\langle\text{hook}\rangle$ being empty means that all three of its `\g__hook_<hook>_code_prop`, its `\__hook_toplevel_<hook>` and its `\__hook_next_<hook>` are empty.

`\hook_if_empty:nTF`

```

2474 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_if_empty:n}
2475 <latexrelease> {Hooks-with-args}
2476 \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2477 {
2478     \if:w
2479         T
2480         \prop_if_exist:cT { g__hook_#1_code_prop }
2481         { \prop_if_empty:cF { g__hook_#1_code_prop } { F } }
2482         \__hook_cs_if_empty:cF { __hook_toplevel~#1 } { F }
2483         \__hook_cs_if_empty:cF { __hook_next~#1 } { F }
2484         T
2485         \prg_return_true:
2486     \else:
2487         \prg_return_false:
2488     \fi:
2489 }
2490 <latexrelease> \EndIncludeInRelease
2491 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_if_empty:n}
2492 <latexrelease> {Hooks-with-args}
2493 <latexrelease> \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2494 <latexrelease> {

```

```

2495 <latexrelease> \_hook_if_structure_exist:nTF {#1}
2496 <latexrelease> {
2497 <latexrelease> \bool_lazy_and:nnTF
2498 <latexrelease> { \prop_if_empty_p:c { g__hook_#1_code_prop } }
2499 <latexrelease> {
2500 <latexrelease> \bool_lazy_and_p:nn
2501 <latexrelease> { \tl_if_empty_p:c { __hook_toplevel~#1 } }
2502 <latexrelease> { \tl_if_empty_p:c { __hook_next~#1 } }
2503 <latexrelease> }
2504 <latexrelease> { \prg_return_true: }
2505 <latexrelease> { \prg_return_false: }
2506 <latexrelease> }
2507 <latexrelease> { \prg_return_true: }
2508 <latexrelease> }
2509 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_if_empty:nTF`. This function is documented on page [232](#).)

`\_hook_if_usable_p:n` A hook is usable if the token list that stores the sorted code for that hook, `\_hook_<hook>`, exists. The property list `g__hook_<hook>_code_prop` cannot be used here because often it is necessary to add code to a hook without knowing if such hook was already declared, or even if it will ever be (for example, in case the package that defines it isn't loaded).

```

2510 \prg_new_conditional:Npnn \_hook_if_usable:n #1 { p , T , F , TF }
2511 {
2512 \cs_if_exist:cTF { __hook~#1 }
2513 { \prg_return_true: }
2514 { \prg_return_false: }
2515 }

```

(End of definition for `\_hook_if_usable:nTF`.)

`\_hook_if_structure_exist_p:n` An internal check if the hook has already its basic internal structure set up with `\_hook_init_structure:n`. This means that the hook was already used somehow (a code chunk or rule was added to it), but it still wasn't declared with `\hook_new:n`.

```

2516 \prg_new_conditional:Npnn \_hook_if_structure_exist:n #1 { p , T , F , TF }
2517 {
2518 \prop_if_exist:cTF { g__hook_#1_code_prop }
2519 { \prg_return_true: }
2520 { \prg_return_false: }
2521 }

```

(End of definition for `\_hook_if_structure_exist:nTF`.)

`\_hook_if_declared_p:n` Internal test to check if the hook was officially declared with `\hook_new:n` or a variant.

```

2522 \prg_new_conditional:Npnn \_hook_if_declared:n #1 { p , T , F , TF }
2523 {
2524 \tl_if_exist:cTF { g__hook_#1_declared_tl }
2525 { \prg_return_true: }
2526 { \prg_return_false: }
2527 }

```

(End of definition for `\_hook_if_declared:nTF`.)

`\_hook_if_reversed_p:n` An internal conditional that checks if a hook is reversed.

`\_hook_if_reversed:nTF`

```

2528 \prg_new_conditional:Npnn \_hook_if_reversed:n #1 { p , T , F , TF }
2529 {
2530   \exp_after:wN \_hook_use_none_delimit_by_s_mark:w
2531   \if:w - \cs:w g\_hook\_#1_reversed_tl \cs_end:
2532     \s\_hook_mark \prg_return_true:
2533   \else:
2534     \s\_hook_mark \prg_return_false:
2535   \fi:
2536 }

```

(End of definition for `\_hook_if_reversed:nTF`.)

`\_hook_if_generic_p:n` An internal conditional that checks if a name belongs to a generic hook. The deprecated version needs to check if #3 is empty to avoid returning true on file/before, for example.

`\_hook_if_generic:nTF`

`\_hook_if_deprecated_generic_p:n`

`\_hook_if_deprecated_generic:nTF`

```

2537 \prg_new_conditional:Npnn \_hook_if_generic:n #1 { T, TF }
2538 { \_hook_if_generic:w #1 / / / \s\_hook_mark }
2539 \cs_new:Npn \_hook_if_generic:w #1 / #2 / #3 / #4 \s\_hook_mark
2540 {
2541   \cs_if_exist:cTF { c\_hook_generic\_#1/./#3_tl }
2542   { \prg_return_true: }
2543   { \prg_return_false: }
2544 }
2545 \prg_new_conditional:Npnn \_hook_if_deprecated_generic:n #1 { T, TF }
2546 { \_hook_if_deprecated_generic:w #1 / / / \s\_hook_mark }
2547 \cs_new:Npn \_hook_if_deprecated_generic:w #1 / #2 / #3 / #4 \s\_hook_mark
2548 {
2549   \cs_if_exist:cTF { c\_hook_deprecated\_#1/./#2_tl }
2550   {
2551     \tl_if_empty:nTF {#3}
2552     { \prg_return_false: }
2553     { \prg_return_true: }
2554   }
2555   { \prg_return_false: }
2556 }

```

(End of definition for `\_hook_if_generic:nTF` and `\_hook_if_deprecated_generic:nTF`.)

`\_hook_if_cmd_hook_p:n` An internal conditional that checks if a given hook is a valid generic cmd hook.

`\_hook_if_cmd_hook:nTF`

`\_hook_if_cmd_hook_p:w`

`\_hook_if_cmd_hook:wTF`

```

2557 <latexrelease> \IncludeInRelease{2023/06/01}{\_hook_if_cmd_hook:n}
2558 <latexrelease> {Hooks-with-args}
2559 \prg_new_conditional:Npnn \_hook_if_cmd_hook:n #1 { T }
2560 { \_hook_if_cmd_hook:w #1 / / / \s\_hook_mark }
2561 \cs_new:Npn \_hook_if_cmd_hook:w #1 / #2 / #3 / #4 \s\_hook_mark
2562 {
2563   \if:w Y
2564     \str_if_eq:nnF {#1} { cmd } { N }
2565     \tl_if_exist:cF { c\_hook_generic\_#1/./#3_tl } { N }
2566     Y
2567     \prg_return_true:
2568   \else:
2569     \prg_return_false:
2570   \fi:
2571 }

```

```

2572 \<latexrelease>\EndIncludeInRelease
2573 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_if_cmd_hook:n}
2574 \<latexrelease>{Hooks-with-args}
2575 \<latexrelease>\cs_undefine:N \__hook_if_cmd_hook:nT
2576 \<latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_if_cmd_hook:nTF and __hook_if_cmd_hook:wTF.)

__hook_if_generic_reversed_p:n An internal conditional that checks if a name belongs to a generic reversed hook.
 __hook_if_generic_reversed:nTF

```

2577 \prg_new_conditional:Npnn \__hook_if_generic_reversed:n #1 { T }
2578 { \__hook_if_generic_reversed:w #1 / / / \scan_stop: }
2579 \cs_new:Npn \__hook_if_generic_reversed:w #1 / #2 / #3 / #4 \scan_stop:
2580 {
2581   \if_charcode:w - \cs:w c__hook_generic_#1/./#3_tl \cs_end:
2582   \prg_return_true:
2583   \else:
2584   \prg_return_false:
2585   \fi:
2586 }

```

(End of definition for __hook_if_generic_reversed:nTF.)

__hook_if_replacing_args:TF An internal conditional that checks if the code being added to the hook contains arguments.
 __hook_misused_if_replacing_args:nn

```

\__hook_replacing_args_true: 2587 \seq_new:N \g__hook_replacing_stack_seq
\__hook_replacing_args_false: 2588 \cs_new:Npn \__hook_misused_if_replacing_args:nn #1 #2
\__hook_replacing_args_reset: 2589 {
\g__hook_replacing_stack_seq 2590   \msg_expandable_error:nnn { latex2e } { should-not-happen }
2591   { Misused-\__hook_if_replacing_args:. }
2592 }
2593 \cs_new:Npn \__hook_if_replacing_args:TF
2594 { \__hook_misused_if_replacing_args:nn }
2595 \cs_new_protected:Npn \__hook_replacing_args_true:
2596 {
2597   \seq_gpush:No \g__hook_replacing_stack_seq
2598   { \__hook_if_replacing_args:TF }
2599   \cs_set:Npn \__hook_if_replacing_args:TF { \use_i:nn }
2600 }
2601 \cs_new_protected:Npn \__hook_replacing_args_false:
2602 {
2603   \seq_gpush:No \g__hook_replacing_stack_seq
2604   { \__hook_if_replacing_args:TF }
2605   \cs_set:Npn \__hook_if_replacing_args:TF { \use_ii:nn }
2606 }
2607 \cs_new_protected:Npn \__hook_replacing_args_reset:
2608 {
2609   \seq_gpop:NN \g__hook_replacing_stack_seq \l__hook_return_tl
2610   \cs_gset_eq:NN \__hook_if_replacing_args:TF \l__hook_return_tl
2611 }

```

(End of definition for __hook_if_replacing_args:TF and others.)

4.11 Messages

Hook errors are LaTeX kernel errors:

```
2612 \prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
And so are kernel errors (this should move elsewhere eventually).
2613 \prop_gput:Nnn \g_msg_module_type_prop { latex2e } { LaTeX }
2614 \prop_gput:Nnn \g_msg_module_name_prop { latex2e } { kernel }

2615 \msg_new:nnnn { hooks } { labels-incompatible }
2616 {
2617   Labels~'#1'~and~'#2'~are~incompatible
2618   \str_if_eq:nnF {#3} {??} { ~in-hook~'#3' } .~
2619   \int_compare:nNnTF {#4} = { 1 }
2620     { The~ code~ for~ both~ labels~ will~ be~ dropped. }
2621     { You~ may~ see~ errors~ later. }
2622 }
2623 { LaTeX~found~two~incompatible~labels~in~the~same~hook.~
2624   This~indicates~an~incompatibility~between~packages. }

2625 \msg_new:nnnn { hooks } { exists }
2626 { Hook~'#1'~ has~ already~ been~ declared. }
2627 { There~ already~ exists~ a~ hook~ declaration~ with~ this~
2628   name.\\
2629   Please~ use~ a~ different~ name~ for~ your~ hook.}

2630 <latexrelease>\IncludeInRelease{2023/06/01}{too-many-args}
2631 <latexrelease>{Hooks-with-args}

2632 \msg_new:nnnn { hooks } { too-many-args }
2633 { Too-many-arguments-for-hook~'#1'. }
2634 {
2635   You~tried~to~declare~a~hook~with~#2~arguments,~but~a~
2636   hook~can~only~have~up~to~nine.~LaTeX~will~define~this~
2637   hook~with~nine~arguments.
2638 }

2639 \msg_new:nnnn { hooks } { without-args }
2640 { Hook~'#1'~has~no~arguments. }
2641 {
2642   You~tried~to~use~\iow_char:N\\#2WithArguments~
2643   on~a~hook~that~takes~no~arguments.\\
2644   Check~the~usage~of~the~hook~or~use~\iow_char:N\\#2~instead.\\
2645   \\
2646   LaTeX~will~use~\iow_char:N\\#2.
2647 }

2648 \msg_new:nnnn { hooks } { one-time-args }
2649 { You~can't~have~arguments~in~used~one-time~hook~'#1'. }
2650 {
2651   You~tried~to~use~\iow_char:N\\#2WithArguments~
2652   on~a~one-time~hook~that~has~already~been~used.~
2653   You~have~to~add~the~code~before~the~hook~is~used,~
2654   or~add~the~code~without~arguments~using~\iow_char:N\\#2~instead.\\
2655   \\
2656   LaTeX~will~use~\iow_char:N\\#2.
2657 }
```



```

2658 \latexrelease\EndIncludeInRelease
2659 \latexrelease\IncludeInRelease{2020/10/01}{too-many-args}
2660 \latexrelease{Hooks-with-args}
2661 \latexrelease\EndIncludeInRelease

2662 \msg_new:nnnn { hooks } { hook-disabled }
2663 { Cannot-add-code-to-disabled-hook~'#1'. }
2664 {
2665   The~hook~'#1'~you~tried~to~add~code~to~was~previously~disabled~
2666   with~\iow_char:N\hook_disable_generic:n-or~
2667   \iow_char:N\DisableGenericHook,~so~
2668   it~cannot~have~code~added~to~it.
2669 }

2670 \msg_new:nnn { hooks } { empty-label }
2671 {
2672   Empty~code~label~\msg_line_context:~
2673   Using~'\_hook_currname_or_default:'~instead.
2674 }

2675 \msg_new:nnn { hooks } { empty-hook }
2676 {
2677   Empty~hook~name~\msg_line_context:.
2678 }

2679 \msg_new:nnn { hooks } { no-default-label }
2680 {
2681   Missing~(empty)~default~label~\msg_line_context:. \\\
2682   This~command~was~ignored.
2683 }

2684 \msg_new:nnnn { hooks } { unknown-rule }
2685 {
2686   Unknown~ relationship~ '#3'~
2687   between~ labels~ '#2'~ and~ '#4'~
2688   \str_if_eq:nnF {#1} {??} { ~in~hook~'#1' }. ~
2689   Perhaps~ a~ misspelling?
2690 }
2691 {
2692   The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
2693   'before'~ or~ '<',~
2694   'after'~ or~ '>',~
2695   'incompatible-warning',~
2696   'incompatible-error',~
2697   'voids'~ or~
2698   'unrelated'.
2699 }

2700 \msg_new:nnnn { hooks } { rule-too-late }
2701 {
2702   Sorting~rule~for~'#1'~hook~applied~too~late.\\
2703   Try~setting~this~rule~earlier.
2704 }
2705 {
2706   You~tried~to~set~the~ordering~of~hook~'#1'~using\\
2707   \ \ \iow_char:N\DeclareHookRule{#1}{#2}{#3}{#4}\\
2708   but~hook~'#1'~was~already~used~as~a~one~time~hook,~

```

```

2709     thus~sorting~is~\
2710     no~longer~possible.~Declare~the~rule~
2711     before~the~hook~is~used.
2712 }
2713 \msg_new:nnnn { hooks } { misused-top-level }
2714 {
2715     Illegal~use~of~\iow_char:N \AddToHook{#1}[top-level]{...}.\
2716     'top-level'~is~reserved~for~the~user's~document.
2717 }
2718 {
2719     The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
2720     be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
2721     '\__hook_currname_or_default:'~for~this~\@cls@pkg,~or~another~
2722     suitable~label.
2723 }
2724 \msg_new:nnn { hooks } { set-top-level }
2725 {
2726     You~cannot~change~the~default~label~#1~'top-level'.~Illegal \
2727     \use:nn { ~ } { ~ } \iow_char:N \#2{#3} \
2728     \msg_line_context:.
2729 }
2730 \msg_new:nnn { hooks } { extra-pop-label }
2731 {
2732     Extra~\iow_char:N \PopDefaultHookLabel. \
2733     This~command~will~be~ignored.
2734 }
2735 \msg_new:nnn { hooks } { missing-pop-label }
2736 {
2737     Missing~\iow_char:N \PopDefaultHookLabel. \
2738     The~label~'#1'~was~pushed~but~never~popped.~Something~is~wrong.
2739 }
2740 \msg_new:nnn { latex2e } { should-not-happen }
2741 {
2742     This~should~not~happen.~#1 \
2743     Please~report~at~https://github.com/latex3/latex2e.
2744 }
2745 \msg_new:nnn { hooks } { activate-disabled }
2746 {
2747     Cannot~ activate~ hook~ '#1'~ because~ it~ is~ disabled!
2748 }
2749 \msg_new:nnn { hooks } { cannot-remove }
2750 {
2751     Cannot~remove~chunk~'#2'~from~hook~'#1'~because~
2752     \__hook_if_structure_exist:nTF {#1}
2753     { it~does~not~exist~in~that~hook. }
2754     { the~hook~does~not~exist. }
2755 }
2756 \msg_new:nnn { hooks } { generic-deprecated }
2757 {
2758     Generic~hook~'#1/#2/#3'~is~deprecated. \
2759     Use~hook~'#1/#3/#2'~instead.
2760 }

```

4.12 L^AT_EX 2_ε package interface commands

\NewHook Declaring new hooks ...

```

\NewReversedHook 2761 \NewDocumentCommand \NewHook          { m }
\NewMirroredHookPair 2762 { \hook_new:n {#1} }
2763 \NewDocumentCommand \NewReversedHook      { m }
2764 { \hook_new_reversed:n {#1} }
2765 \NewDocumentCommand \NewMirroredHookPair { mm }
2766 { \hook_new_pair:nn {#1}{#2} }
```

(End of definition for \NewHook, \NewReversedHook, and \NewMirroredHookPair. These functions are documented on page 217.)

\NewHookWithArguments Declaring new hooks with arguments...

```

\NewReversedHookWithArguments 2767 <latexrelease> \IncludeInRelease{2023/06/01}{\NewHookWithArguments}
\NewMirroredHookPairWithArguments 2768 <latexrelease> {Hooks-with-args}
2769 \NewDocumentCommand \NewHookWithArguments { mm }
2770 { \hook_new_with_args:nn {#1} {#2} }
2771 \NewDocumentCommand \NewReversedHookWithArguments { mm }
2772 { \hook_new_reversed_with_args:nn {#1} {#2} }
2773 \NewDocumentCommand \NewMirroredHookPairWithArguments { mmm }
2774 { \hook_new_pair_with_args:nnn {#1} {#2} {#3} }
2775 <latexrelease> \EndIncludeInRelease
2776 <latexrelease> \IncludeInRelease{2020/10/01}{\NewHookWithArguments}
2777 <latexrelease> {Hooks-with-args}
2778 <latexrelease> \cs_new_protected:Npn \NewHookWithArguments #1 #2 { }
2779 <latexrelease> \cs_new_protected:Npn \NewReversedHookWithArguments #1 #2 { }
2780 <latexrelease> \cs_new_protected:Npn \NewMirroredHookPairWithArguments #1 #2 #3 { }
2781 <latexrelease> \EndIncludeInRelease
```

(End of definition for \NewHookWithArguments, \NewReversedHookWithArguments, and \NewMirroredHookPairWithArguments. These functions are documented on page 218.)

```

2782 <latexrelease> \IncludeInRelease{2021/06/01}{\ActivateGenericHook}
2783 <latexrelease> {Providing-hooks}
```

\ActivateGenericHook Providing new hooks ...

```

2784 \NewDocumentCommand \ActivateGenericHook { m }
2785 { \hook_activate_generic:n {#1} }
```

(End of definition for \ActivateGenericHook. This function is documented on page 219.)

\DisableGenericHook Disabling a generic hook.

```

2786 \NewDocumentCommand \DisableGenericHook { m }
2787 { \hook_disable_generic:n {#1} }
```

(End of definition for \DisableGenericHook. This function is documented on page 218.)

```

2788 <latexrelease> \EndIncludeInRelease
2789 <latexrelease> \IncludeInRelease{2020/10/01}{\ActivateGenericHook}
2790 <latexrelease> {Providing-hooks}
2791 <latexrelease> \def \ActivateGenericHook #1 { }
2792 <latexrelease> \def \DisableGenericHook #1 { }
2793 <latexrelease> \EndIncludeInRelease
```

\AddToHook

\AddToHookWithArguments

```
2794 <latexrelease> \IncludeInRelease{2023/06/01}{\AddToHookWithArguments}
2795 <latexrelease> {Hooks-with-args}
2796 \NewDocumentCommand \AddToHook { m o +m }
2797 { \hook_gput_code:nnn {#1} {#2} {#3} }
2798 \NewDocumentCommand \AddToHookWithArguments { m o +m }
2799 { \hook_gput_code_with_args:nnn {#1} {#2} {#3} }
2800 <latexrelease> \EndIncludeInRelease
2801 <latexrelease> \IncludeInRelease{2020/10/01}{\AddToHookWithArguments}
2802 <latexrelease> {Hooks-with-args}
2803 <latexrelease> \cs_new_protected:Npn \AddToHookWithArguments #1 #2 #3 { }
2804 <latexrelease> \EndIncludeInRelease
```

(End of definition for \AddToHook and \AddToHookWithArguments. These functions are documented on page 220.)

\AddToHookNext

\AddToHookNextWithArguments

```
2805 <latexrelease> \IncludeInRelease{2023/06/01}{\AddToHookNextWithArguments}
2806 <latexrelease> {Hooks-with-args}
2807 \NewDocumentCommand \AddToHookNext { m +m }
2808 { \hook_gput_next_code:nn {#1} {#2} }
2809 \NewDocumentCommand \AddToHookNextWithArguments { m +m }
2810 { \hook_gput_next_code_with_args:nn {#1} {#2} }
2811 <latexrelease> \EndIncludeInRelease
2812 <latexrelease> \IncludeInRelease{2020/10/01}{\AddToHookNextWithArguments}
2813 <latexrelease> {Hooks-with-args}
2814 <latexrelease> \cs_new_protected:Npn \AddToHookNextWithArguments #1 #2 { }
2815 <latexrelease> \EndIncludeInRelease
```

(End of definition for \AddToHookNext and \AddToHookNextWithArguments. These functions are documented on page 222.)

\ClearHookNext

```
2816 \NewDocumentCommand \ClearHookNext { m }
2817 { \hook_gclear_next_code:n {#1} }
```

(End of definition for \ClearHookNext. This function is documented on page 222.)

\RemoveFromHook

```
2818 \NewDocumentCommand \RemoveFromHook { m o }
2819 { \hook_gremove_code:nn {#1} {#2} }
```

(End of definition for \RemoveFromHook. This function is documented on page 221.)

\SetDefaultHookLabel

\PushDefaultHookLabel

\PopDefaultHookLabel

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```
2820 \NewDocumentCommand \SetDefaultHookLabel { m }
2821 { \__hook_set_default_hook_label:n {#1} }
```

The label is only automatically updated with `\@onefilewithoptions` (`\usepackage` and `\documentclass`), but some packages, like `TikZ`, define package-like interfaces, like `\usetikzlibrary` that are wrappers around `\input`, so they inherit the default label currently in force (usually `top-level`, but it may change if loaded in another package). To provide a package-like behavior also for hooks in these files, we provide high-level access to the default label stack.

```

2822 \NewDocumentCommand \PushDefaultHookLabel { m }
2823 { \__hook_curr_name_push:n {#1} }
2824 \NewDocumentCommand \PopDefaultHookLabel { }
2825 { \__hook_curr_name_pop: }

```

The current label stack holds the labels for all files but the current one (more or less like `\@currnamestack`), and the current label token list, `\g__hook_hook_curr_name_tl`, holds the label for the current file. However `\@pushfilename` happens before `\@currname` is set, so we need to look ahead to get the `\@currname` for the label. `expl3` also requires the current file in `\@pushfilename`, so here we abuse `\@expl@push@filename@aux@@` to do `\__hook_curr_name_push:n`.

```

2826 \cs_gset_protected:Npn \@expl@push@filename@aux@@ #1#2#3
2827 {
2828   \__hook_curr_name_push:n {#3}
2829   \str_gset:Nx \g_file_curr_name_str {#3}
2830   #1 #2 {#3}
2831 }

```

(End of definition for `\SetDefaultHookLabel`, `\PushDefaultHookLabel`, and `\PopDefaultHookLabel`. These functions are documented on page 225.)

`\UseHook` Avoid the overhead of `xparse` and its protection that we don't want here (since the hook should vanish without trace if empty)!

`\UseOneTimeHook`

`\UseHookWithArguments`

`\UseOneTimeHookWithArguments`

```

2832 <latexrelease>\IncludeInRelease{2023/06/01}{\UseHookWithArguments}
2833 <latexrelease> {Hooks-with-args}
2834 \cs_new:Npn \UseHook { \hook_use:n }
2835 \cs_new:Npn \UseOneTimeHook { \hook_use_once:n }
2836 \cs_new:Npn \UseHookWithArguments { \hook_use_nnw }
2837 \cs_new:Npn \UseOneTimeHookWithArguments { \hook_use_once_nnw }
2838 <latexrelease>\EndIncludeInRelease
2839 <latexrelease>\IncludeInRelease{2020/10/01}{\UseHookWithArguments}
2840 <latexrelease> {Hooks-with-args}
2841 <latexrelease>\cs_new:Npn \UseHookWithArguments #1 #2 { }
2842 <latexrelease>\cs_new:Npn \UseOneTimeHookWithArguments #1 #2 { }
2843 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\UseHook` and others. These functions are documented on page 219.)

`\ShowHook`

`\LogHook`

```

2844 \cs_new_protected:Npn \ShowHook { \hook_show:n }
2845 \cs_new_protected:Npn \LogHook { \hook_log:n }

```

(End of definition for `\ShowHook` and `\LogHook`. These functions are documented on page 228.)

`\DebugHooksOn`

`\DebugHooksOff`

```

2846 \cs_new_protected:Npn \DebugHooksOn { \hook_debug_on: }
2847 \cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }

```

(End of definition for `\DebugHooksOn` and `\DebugHooksOff`. These functions are documented on page 229.)

`\DeclareHookRule`

```

2848 \NewDocumentCommand \DeclareHookRule { m m m m }
2849 { \hook_gset_rule:nnnn {#1}{#2}{#3}{#4} }

```

(End of definition for `\DeclareHookRule`. This function is documented on page 226.)

`\DeclareDefaultHookRule` This declaration is only supported before `\begin{document}`.

```
2850 \NewDocumentCommand \DeclareDefaultHookRule { m m m }
2851 { \hook_gset_rule:nnnn {??}{#1}{#2}{#3} }
2852 \@onlypreamble\DeclareDefaultHookRule
```

(End of definition for `\DeclareDefaultHookRule`. This function is documented on page 227.)

`\ClearHookRule` A special setup rule that removes an existing relation. Basically `\_hook_rule\_gclear:nnn` plus fixing the property list for debugging.

FMi: Needs perhaps an L3 interface, or maybe it should get dropped?

```
2853 \NewDocumentCommand \ClearHookRule { m m m }
2854 { \hook_gset_rule:nnnn {#1}{#2}{unrelated}{#3} }
```

(End of definition for `\ClearHookRule`. This function is documented on page 226.)

`\IfHookEmptyTF` Here we avoid the overhead of `xparse`, since `\IfHookEmptyTF` is used in `\end` (that is, every L^AT_EX environment). As a further optimization, use `\let` rather than `\def` to avoid one expansion step.

`\IfHookEmptyT`
`\IfHookEmptyF`

```
2855 \cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF
2856 \cs_new_eq:NN \IfHookEmptyT \hook_if_empty:nT
2857 \cs_new_eq:NN \IfHookEmptyF \hook_if_empty:nF
```

(End of definition for `\IfHookEmptyTF`, `\IfHookEmptyT`, and `\IfHookEmptyF`. These functions are documented on page 228.)

`\IfHookExistsTF` Marked for removal and no longer documented in the doc section!

PhO: \IfHookExistsTF is used in `jlreq.cls`, `pxatbegshi.sty`, `pxeveryssel.sty`, `pxeveryshi.sty`, so the public name may be an alias of the internal conditional for a while. Regardless, those packages' use for `\IfHookExistsTF` is not really correct and can be changed.

```
2858 \cs_new_eq:NN \IfHookExistsTF \_hook_if_usable:nTF
```

(End of definition for `\IfHookExistsTF`.)

4.13 Deprecated that needs cleanup at some point

`\hook_disable:n` Deprecated.

```
\hook_provide:n 2859 \cs_new_protected:Npn \hook_disable:n
\hook_provide_reversed:n 2860 {
\hook_provide_pair:nn 2861 \_hook_deprecated_warn:nn
\_hook_activate_generic_reversed:n 2862 { hook_disable:n }
\_hook_activate_generic_pair:nn 2863 { hook_disable_generic:n }
2864 \hook_disable_generic:n
2865 }
2866 \cs_new_protected:Npn \hook_provide:n
2867 {
2868 \_hook_deprecated_warn:nn
2869 { hook_provide:n }
2870 { hook_activate_generic:n }
2871 \hook_activate_generic:n
2872 }
2873 \cs_new_protected:Npn \hook_provide_reversed:n
```

```

2874 {
2875   \__hook_deprecated_warn:nn
2876   { hook_provide_reversed:n }
2877   { hook_activate_generic:n }
2878   \__hook_activate_generic_reversed:n
2879 }
2880 \cs_new_protected:Npn \hook_provide_pair:nn
2881 {
2882   \__hook_deprecated_warn:nn
2883   { hook_provide_pair:nn }
2884   { hook_activate_generic:n }
2885   \__hook_activate_generic_pair:nn
2886 }
2887 \cs_new_protected:Npn \__hook_activate_generic_reversed:n #1
2888 { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { - } }
2889 \cs_new_protected:Npn \__hook_activate_generic_pair:nn #1#2
2890 { \hook_activate_generic:n {#1} \__hook_activate_generic_reversed:n {#2} }

```

(End of definition for \hook_disable:n and others.)

```

\DisableHook   Deprecated.
\ProvideHook   2891 \cs_new_protected:Npn \DisableHook
\ProvideReversedHook 2892 {
\ProvideMirroredHookPair 2893   \__hook_deprecated_warn:nn
2894   { DisableHook }
2895   { DisableGenericHook }
2896   \hook_disable_generic:n
2897 }
2898 \cs_new_protected:Npn \ProvideHook
2899 {
2900   \__hook_deprecated_warn:nn
2901   { ProvideHook }
2902   { ActivateGenericHook }
2903   \hook_activate_generic:n
2904 }
2905 \cs_new_protected:Npn \ProvideReversedHook
2906 {
2907   \__hook_deprecated_warn:nn
2908   { ProvideReversedHook }
2909   { ActivateGenericHook }
2910   \__hook_activate_generic_reversed:n
2911 }
2912 \cs_new_protected:Npn \ProvideMirroredHookPair
2913 {
2914   \__hook_deprecated_warn:nn
2915   { ProvideMirroredHookPair }
2916   { ActivateGenericHook }
2917   \__hook_activate_generic_pair:nn
2918 }

```

(End of definition for \DisableHook and others.)

```

\__hook_deprecated_warn:nn Warns about a deprecation, telling what should be used instead.
2919 \cs_new_protected:Npn \__hook_deprecated_warn:nn #1 #2
2920 { \msg_warning:nnnn { hooks } { deprecated } {#1} {#2} }

```

```

2921 \msg_new:nnn { hooks } { deprecated }
2922 {
2923   Command~\iow_char:N\|#1~is-deprecated-and-will-be-removed-in-a~
2924   future-release. \\ \\
2925   Use~\iow_char:N\|#2~instead.
2926 }

```

(End of definition for `\_hook_deprecated_warn:nn`.)

4.14 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX_{2 ϵ} names to allow for internal commands to be used outside this module. We have to unset the `@@` since we want double “at” sign in place of double underscores.

```

2927 <@@= >

```

```

\@expl@@@initialize@all@@
\@expl@@@hook@curr@name@pop@@

```

```

2928 \cs_new_eq:NN \@expl@@@initialize@all@@
2929   \__hook_initialize_all:
2930 \cs_new_eq:NN \@expl@@@hook@curr@name@pop@@
2931   \__hook_curr_name_pop:

```

(End of definition for `\@expl@@@initialize@all@@` and `\@expl@@@hook@curr@name@pop@@`.)

Rolling back here doesn’t undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```

2932 %
2933 <latexrelease> \IncludeInRelease{0000/00/00}{1thooks}
2934 <latexrelease> {The~hook~management}%
2935 <latexrelease>
2936 <latexrelease> \def \NewHook#1{}
2937 <latexrelease> \def \NewReversedHook#1{}
2938 <latexrelease> \def \NewMirroredHookPair#1#2{}
2939 <latexrelease>
2940 <latexrelease> \def \DisableGenericHook #1{}
2941 <latexrelease>
2942 <latexrelease> \long\def \AddToHookNext#1#2{}
2943 <latexrelease>
2944 <latexrelease> \def \AddToHook#1{\@gobble@AddToHook@args}
2945 <latexrelease> \providecommand \@gobble@AddToHook@args[2] [] {}
2946 <latexrelease>
2947 <latexrelease> \def \RemoveFromHook#1{\@gobble@RemoveFromHook@arg}
2948 <latexrelease> \providecommand \@gobble@RemoveFromHook@arg[1] [] {}
2949 <latexrelease>
2950 <latexrelease> \def \UseHook #1{}
2951 <latexrelease> \def \UseOneTimeHook #1{}
2952 <latexrelease> \def \ShowHook #1{}
2953 <latexrelease> \let \DebugHooksOn \@empty
2954 <latexrelease> \let \DebugHooksOff \@empty
2955 <latexrelease>
2956 <latexrelease> \def \DeclareHookRule #1#2#3#4{}
2957 <latexrelease> \def \DeclareDefaultHookRule #1#2#3{}
2958 <latexrelease> \def \ClearHookRule #1#2#3{}

```


If the hook management is not provided we make the test for existence false and the test for empty true in the hope that this is most of the time reasonable. If not a package would need to guard against running in an old kernel.

```

2959 <latexrelease>\long\def \IfHookExistsTF #1#2#3{#3}
2960 <latexrelease>\long\def \IfHookEmptyTF #1#2#3{#2}
2961 <latexrelease>
2962 <latexrelease>\EndModuleRelease
2963 <@@=hook>

2964 <latexrelease>\cs:w __hook_rollback_tidying: \cs_end:
2965 <latexrelease>\bool_lazy_and:nnT
2966 <latexrelease>    { \int_compare_p:nNn { \sourceLaTeXdate } > { 20230600 } }
2967 <latexrelease>    { \int_compare_p:nNn { \requestedLaTeXdate } < { 20230601 } }
2968 <latexrelease> {
2969 <latexrelease>     \cs_gset_protected:Npn \__hook_rollback_tidying:
2970 <latexrelease>     {
2971 <latexrelease>         \@latex@error { Rollback-code-executed-twice }
2972 <latexrelease>         {
2973 <latexrelease>             Something~went~wrong~(unless~this~was~
2974 <latexrelease>             done~on~purpose~in~a~testing~environment).
2975 <latexrelease>         }
2976 <latexrelease>         \use_none:nnnn
2977 <latexrelease>     }
2978 <latexrelease> \cs_set:Npn \__hook_tmp:w #1 #2
2979 <latexrelease> {
2980 <latexrelease>     \__hook_tl_gset:ce { __hook#1~#2 }
2981 <latexrelease>     {
2982 <latexrelease>         \exp_args:No \exp_not:o
2983 <latexrelease>         {
2984 <latexrelease>             \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
2985 <latexrelease>             { \__hook_braced_cs_parameter:n
2986 <latexrelease>                 { __hook#1~#2 } }
2987 <latexrelease>         }
2988 <latexrelease>     }
2989 <latexrelease> }
2990 <latexrelease> \seq_map_inline:Nn \g__hook_all_seq
2991 <latexrelease> {
2992 <latexrelease>     \exp_after:wN \cs_gset_nopar:Npn
2993 <latexrelease>     \cs:w g__hook_#1_code_prop \exp_args:NNo \exp_args:No
2994 <latexrelease>     \cs_end: { \cs:w g__hook_#1_code_prop \cs_end: }
2995 <latexrelease>     \__hook_tmp:w { _toplevel } {#1}
2996 <latexrelease>     \__hook_tmp:w { _next } {#1}
2997 <latexrelease> }
2998 <latexrelease> }
2999 \ExplSyntaxOff
3000 </2ekernel | latexrelease>
3001 <@@=>

```

File 09

ltxcmdhooks.dtx

1 Introduction

This file implements generic hooks for (arbitrary) commands. In theory every command `\<name>` offers now two associated hooks to which code can be added using `\AddToHook`,¹³ `\AddToHookNext`, `\AddToHookWithArguments`, and `\AddToHookNextWithArguments`.¹⁴

However, this is only true “in theory”. In practice there are a number of restrictions that makes it impossible to use such generic command hooks in a number of cases, so please read all of section 2 to understand what may prevent you from using them successfully.

The generic command hooks are:

cmd/<name>/before This hook is executed at the very start of the command, right after its arguments (if any) are parsed. The hook `<code>` runs in the command inside a call to `\UseHookWithArguments`. Any code added to this hook using `\AddToHookWithArguments` or `\AddToHookNextWithArguments` can access the command’s arguments using `#1`, `#2`, etc., up to the number of arguments of the command. If `\AddToHook` or `\AddToHookNext` are used, the arguments cannot be accessed (see the `lthooks` documentation¹⁵ on hooks with arguments).

cmd/<name>/after This hook is similar to **cmd/<name>/before**, but it is executed at the very end of the command body. This hook is implemented as a reversed hook.

The hooks are not physically present before `\begin{document}`¹⁶ (i.e., using a command in the preamble will never execute the hook) and if nobody has declared any code for them, then they are not added to the command code ever. For example, if we have the following definition

```
\newcommand\foo[2]{Code #1 for #2!}
```

then executing `\foo{A}{B}` will simply run `Code_A_for_B!` as it was always the case. However, if somebody, somewhere (e.g., in a package) adds

```
\AddToHook{cmd/foo/before}{<before code>}
```

then, after `\begin{document}` the definition of `\foo` will be:

```
\renewcommand\foo[2]{%
  \UseHookWithArguments{cmd/foo/before}{2}{#1}{#2}%
  Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

¹³In this documentation, when something is being said about `\AddToHook`, the same will be valid for `\AddToHookWithArguments`, unless that particular paragraph is highlighting the differences between both. The same is true for the other hook-related functions and their `...WithArguments` counterparts.

¹⁴In practice this is not supported for all types of commands, see section 2.4 for the restrictions that apply and what happens if one tries to use this with commands for which this is not supported.

¹⁵`texdoc lthooks-doc`

¹⁶More specifically, they are inserted in the commands after the `\begin{document}` hook, so they are also not present while `LATEX` is reading the `.aux` file.

```

\renewcommand\foo[2]{%
  Code #1 for #2!%
  \UseHookWithArguments{cmd/foo/after}{2}{#1}{#2}}

```

In other words, the mechanism is similar to what `etoolbox` offers with `\pretocmd` and `\apptocmd` with the important differences

- that code can be prepended or appended (i.e., added to the hooks) even if the command itself is not (yet) defined, because the defining package has not been loaded at this point;
- and that by using the hook management interface it is now possible to define how the code chunks added in these places are ordered, if different packages want to add code at these points.

2 Restrictions and operational details

Adding arbitrary material to commands is tricky because most of the time we do not know what the macro expects as arguments when expanding and $\text{\TeX}$ doesn't have a reliable way to see that, so some guesswork has to be employed.

We can do this in most cases when commands are defined using `\NewDocumentCommand` or `\newcommand` (with a few exceptions). For commands defined with `\def` the situation is less good. Common cases where the command hooks will not work are:

- Commands that use special catcode settings within their definition. In that case it is usually not possible to augment the definition (see 2.1).
- If a command is defined while `\ExplSyntaxOn` is in force **and** the command contains `~` characters to represent spaces, then it can't be patched to include the command hooks. In fact in some very special circumstances you might even get a low-level error rather than the information that the command can't be patched (see, for example, <https://github.com/latex3/latex2e/issues/1430>).
- Commands that have arguments as far as the user is concerned (e.g., `\section` or `\caption`), but are defined in a way that these arguments are not read by the user level command but only later during the processing. In that case the `after` hook doesn't work at all. The before hook only works with `\AddToHook` but not with `\AddToHookWithArguments` because the arguments haven't been read at that point where the hook is patched in. See section 2.4.
- Adding a specific generic command hook is only attempted once per command, thus after redefining a command such hooks will no longer be there and will also not being re-added, see section 2.1.1.

All this means that you have to have a good understanding of how commands are defined when you attempt to make use of such hooks and something goes wrong. What can help in that case is to turn on `\DebugHooksOn` in which case you get much more (low-level) details on why something fails and what was tried to enable the hooks.

2.1 Patching

The code here tries to find out if a command was defined with `\newcommand` or `\DeclareRobustCommand` or `\NewDocumentCommand`, and if so it *assumes* that the argument specification of the command is as expected (which is not fail-proof, if someone redefines the internals of these commands in devious ways, but is a reasonable assumption).

If the command is one of the defined types, the code here does a sandboxed expansion of the command such that it can be redefined again exactly as before, but with the hook code added.

If however the command is not a known type (it was defined with `\def`, for example), then the code uses an approach similar to `etoolbox`'s `\patchcmd` to retokenize the command with the hook code in place. This procedure, however, is more likely to fail if the catcode settings are not the same as the ones at the time of command's definition, so not always adding a hook to a command will work.

2.1.1 Timing

When `\AddToHook` (or its `expl3` equivalent) is called with a generic `cmd` hook, say, `cmd/foo/before`, for the first time (that is, no code was added to that same hook before), in the preamble of a document, it will store a patch instruction for that command until `\begin{document}`, and only then all the commands which had hooks added will be patched in one go. That means that no command in the preamble will have hooks patched into them.

At `\begin{document}` all the delayed patches will be executed, and if the command doesn't exist the code is still added to the hook, but it will not be executed. After `\begin{document}`, when `\AddToHook` is called with a generic `cmd` hook the first time, the command will be immediately patched to include the hook, and if it doesn't exist or if it can't be patched for any reason, an error is thrown; if `\AddToHook` was already used in the preamble no new patching is attempted.

This has the consequence that a command defined or redefined after `\begin{document}` only uses generic `cmd` hook code if `\AddToHook` is called for the first time after the definition is made, or if the command explicitly uses the generic hook in its definition by declaring it with `\NewHookPair` adding `\UseHook` as part of the code.¹⁷

2.2 Command copies

Once a hook has been added to a command, it will be present in any copies of that command which are made. For example, if in the preamble we have

```
\NewDocumentCommand\foo{}{}  
\AddToHook{cmd/foo/after}{1}%
```

then after `\begin{document}`, any use of `\NewCommandCopy` will include the hook in the copy, for example

```
\NewCommandCopy\baz\foo
```

would include the hook in this copied command.

¹⁷We might change this behavior in the main document slightly after gaining some usage experience.

2.3 Grouping

Command hooks are intended to be added to document commands, which are typically defined globally. As such, adding a hook is a global operation, even if the command was previously only locally-defined. Addition of material to hooks is also global.

2.4 Commands that look ahead

Some commands are defined in different “steps” and they look ahead in the input stream to find more arguments. If you try to add some code to the `cmd/<name>/after` hook of such command, it will not work, and it is not possible to detect that programmatically, so the user has to know (or find out) which commands can or cannot have hooks attached to them.

One good example is the `\section` command. You can add something to the `cmd/section/before` hook (but only with `\AddToHook` not `\AddToHookWithArguments`), but if you try to add anything to the `cmd/section/after` hook, `\section` will no longer work at all. That happens because the `\section` macro takes no argument, but instead calls a few internal `LATEX` macros to look for the optional and mandatory arguments. By adding code to the `cmd/section/after` hook, you get in the way of that scanning.

In such a case, where it is known that a specific generic command hook does not work if code is added to it, the package author can add a `\DisableGenericHook`¹⁸ declaration to prevent this from happening in user documents and thereby avoiding obscure errors.

3 Package author interface

The `cmd` hooks are, by default, available for all commands that can be patched to add the hooks. For some commands, however, the very beginning or the very end of the code is not the best place to put the hooks, for example, if the command looks ahead for arguments (see section 2.4).

If you are a package author and you want to add the hooks to your own commands in the proper position you can define the command and manually add the `\UseHookWithArguments` calls inside the command in the proper positions, and manually define the hooks with `\NewHookWithArguments` or `\NewReversedHookWithArguments`. When the hooks are explicitly defined, patching is not attempted so you can make sure your command works properly. For example, an (admittedly not really useful) command that typesets its contents in a framed box with width optionally given in parentheses:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{\parbox{#1}{#2}}}
```

If you try that definition, then add some code after it with

```
\AddToHook{cmd/fancybox/after}{<code>}
```

and then use the `\fancybox` command you will see that it will be completely broken, because the hook will get executed in the middle of parsing for optional `(...)` argument.

If, on the other hand, you want to add hooks to your command you can do something like:

¹⁸Please use `\DisableGenericHook` if at all, only on hooks that you “own”, i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

```

\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{%
    \UseHookWithArguments{cmd/fancybox/before}{2}{#1}{#2}%
    \parbox{#1}{#2}%
    \UseHookWithArguments{cmd/fancybox/after}{2}{#1}{#2}}}
\NewHookWithArguments{cmd/fancybox/before}{2}
\NewReversedHookWithArguments{cmd/fancybox/after}{2}

```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHookWithArguments` or `\NewReversedHookWithArguments`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHookWithArguments{cmd/fancybox/before}` does not need to be in the definition of `\fancybox`, but anywhere it makes sense to insert it (in this case in the internal `\@fancybox`).

Alternatively, if for whatever reason your command does not support the generic hooks provided here, you can disable a hook with `\DisableGenericHook`¹⁹, so that when someone tries to add code to it they will get an error. Or if you don't want the error, you can simply declare the hook with `\NewHook` and never use it.

The above approach is useful for really complex commands where for one or the other reason the hooks can't be placed at the very beginning and end of the command body and some hand-crafting is needed. However, in the example above the real (and in fact only) issue is the cascading argument parsing in the style developed long ago in L^AT_EX 2.09. Thus, a much simpler solution for this case is to replace it with the modern `\NewDocumentCommand` syntax and define the command as follows:

```

\DeclareDocumentCommand\fancybox{D()}{5cm}m}{\fbox{\parbox{#1}{#2}}}

```

If you do that then both hooks automatically work and are patched into the right places.

3.1 Arguments and redefining commands

The code in `ltxcmdhooks` does its best to find out how many arguments a given command has, and to insert the appropriate call to `\UseHookWithArguments`, so that the arguments seen by the hook are exactly those grabbed by the command (the hook, after all, is a macro call, so the arguments have to be placed in the right order, or they won't match).

When using the package writer interface, as discussed in section 3, to change the position of the hooks in your commands, you are also free to change how the hook code in your command sees its arguments. When a `cmd` hook is declared with `\NewHook` (or `\NewHookWithArguments` or other variations of that), it loses its “generic” nature and works as a regular hook. This means that you may choose to declare it without arguments regardless if the command takes arguments or not, or declare it with arguments, even if the command takes none.

However, this flexibility should not be abused. When using a nonstandard configuration for the hook arguments, think reasonably: a user will expect that the argument `#1` in the hook corresponds to the argument's first argument, and so on. Any other configuration is likely to cause confusion and, if used, will have to be well documented.

This flexibility, however, allows you to “correct” the arguments for the hooks. For example, L^AT_EX's `\refstepcounter` has a single argument, the name of the counter. The `cleveref` package adds an optional argument to `\refstepcounter`, making the name of

¹⁹Please use `\DisableGenericHook` if at all, only on hooks that you “own”, i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

the counter argument #2. If the author of `cleveref` wanted, for whatever reason, to add hooks to `\refstepcounter`, to preserve compatibility he could write something along the lines of:

```
\NewHookWithArguments{cmd/refstepcounter/before}{1}
\renewcommand\refstepcounter[2] [<default>] {%
  \UseHookWithArguments{cmd/refstepcounter/before}{1}{#2}%
  <code for \refstepcounter>}
```

so that the mandatory argument, which is arg #2 in the definition, would still be seen as #1 in the hook code.

Another possibility would be to place the optional argument as the second argument for the hook, so that people looking for it would be able to use it. In either case, it would have to be well documented to cause as little confusion as possible.

4 The Implementation

4.1 Execution plan

To add `before` and `after` hooks to a command we will need to peek into the definition of a command, which is always a tricky thing to do. Some cases are easy because we know how the command was defined, so we can assume how its *<parameter text>* looks like (for example a command defined with `\newcommand` may have an optional argument followed by a run of mandatory arguments), so we can just expand that command and make it grab #1, #2, etc. as arguments and define it all back with the hooks added.

Life's usually not that easy, so with some commands we can't do that (a #1 might as well be #12112 instead of the expected #6112, for example) so we need to resort to "patching" the command: read its `\meaning`, and tokenize it again with `\scantokens` and hope for the best.

So the overall plan is:

1. Check if a command is of a known type (that is, defined with `\newcommand`²⁰, `\DeclareRobustCommand`, or `\New(Expandable)DocumentCommand`), and if is, take appropriate action.
2. If the command is not a known type, we'll check if the command can be patched. Two things will prevent a command from being patched: if it was defined in a nonstandard catcode setting, or if it is an internal expl3 command with `__<module>` in its name, in which case we refuse to patch.
3. If the command was defined in nonstandard catcode settings, we will try a few standard ones to try our best to carry out the patching. If this doesn't help either, the code will give up and throw an error.

```
1 <@@=hook>
2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <[latexrelease] \NewModuleRelease{2021/06/01}{ltxcmdhooks}
5 <[latexrelease]                                     {The-hook-management-system-for-commands}
```

²⁰It's not always possible to reliably detect this case because a command defined with no optional argument is indistinguishable from a `\defed` command.

4.2 Variables

<code>\g_hook_patch_action_list_tl</code>	<p>Pairs of <code>\if<cmd>..\patch<cmd></code> to be used with <code>\robust@command@act</code> when looking for a known patching rule. This token list is exposed because we see some future applications (with very specialized packages, such as <code>etoolbox</code> that may want to extend the pairs processed. It is not meant for general use which is why it is not documented in the interface documentation above.</p> <pre> 6 \tl_new:N \g_hook_patch_action_list_tl (End of definition for \g_hook_patch_action_list_tl.) </pre>
<code>\l__hook_patch_num_args_int</code>	<p>The number of arguments in a macro being patched.</p> <pre> 7 \int_new:N \l__hook_patch_num_args_int (End of definition for \l__hook_patch_num_args_int.) </pre>
<code>\l__hook_patch_prefixes_tl</code> <code>\l__hook_param_text_tl</code> <code>\l__hook_replace_text_tl</code>	<p>The prefixes and parameters of the definition for the macro being patched.</p> <pre> 8 \tl_new:N \l__hook_patch_prefixes_tl 9 \tl_new:N \l__hook_param_text_tl 10 \tl_new:N \l__hook_replace_text_tl (End of definition for \l__hook_patch_prefixes_tl, \l__hook_param_text_tl, and \l__hook_replace_text_tl.) </pre>
<code>\c__hook_hash_tl</code> <code>\c__hook_hashes_tl</code>	<p>Two constant token lists that contain one and two parameter tokens.</p> <pre> 11 \tl_const:Nn \c__hook_hash_tl { # } 12 \tl_const:Nn \c__hook_hashes_tl { # # } (End of definition for \c__hook_hash_tl and \c__hook_hashes_tl.) </pre>
<code>__hook_exp_not:NN</code> <code>__hook_def_cmd:w</code>	<p>Two temporary macros that change depending on the macro being patched.</p> <pre> 13 \cs_new_eq:NN __hook_exp_not:NN ? 14 \cs_new_eq:NN __hook_def_cmd:w ? (End of definition for __hook_exp_not:NN and __hook_def_cmd:w.) </pre>
<code>\q__hook_recursion_tail</code> <code>\q__hook_recursion_stop</code>	<p>Internal quarks for recursion: they can't appear in any macro being patched.</p> <pre> 15 \quark_new:N \q__hook_recursion_tail 16 \quark_new:N \q__hook_recursion_stop (End of definition for \q__hook_recursion_tail and \q__hook_recursion_stop.) </pre>
<code>\g__hook_delayed_patches_prop</code>	<p>A list containing the patches delayed to <code>\begin{document}</code>, so that patching is not attempted twice.</p> <pre> 17 \prop_new:N \g__hook_delayed_patches_prop (End of definition for \g__hook_delayed_patches_prop.) </pre>
<code>__hook_patch_debug:e</code>	<p>A helper for patching debug info.</p> <pre> 18 \cs_new_protected:Npn __hook_patch_debug:e #1 19 { __hook_debug:n { \iow_term:e { [lthooks]~#1 } } } (End of definition for __hook_patch_debug:e.) </pre>

4.3 Patching or delaying

Before `\begin{document}` all patching is delayed.

`\__hook_try_put_cmd_hook:n` This function is called from within `\AddToHook`, when code is first added to a generic `cmd` hook. If it is called within in the preamble, it delays the action until `\begin{document}`; otherwise it tries to update the hook.

```

20 <latexrelease>\IncludeInRelease{2024/12/22}{\__hook_try_put_cmd_hook:n}%
21 <latexrelease>          {Don't~define~command}
22 \cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
23   { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
24 \cs_new_protected:Npn \__hook_try_put_cmd_hook:w
25   #1 / #2 / #3 / #4 \s__hook_mark #5
26   {
27     \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }

```

`\__hook_patch_cmd_or_delay:Nnn` expects the command to be patched as its first argument so we need to construct it from its name (`#2`). However, at this moment it may not exist yet, so using `\cs:w` would incorrectly turn it from “undefined” into `\relax`. We therefore use the following curious construction: we start a group and expand out of it to call `\cs:w`. If the command is now changed to `\relax` the `\group_end:` will undo that change, but the token is nevertheless there to be consumed by `\__hook_patch_cmd_or_delay:Nnn`.

```

28   \group_begin:
29     \exp_after:wN
30   \group_end:
31     \exp_after:wN
32   \__hook_patch_cmd_or_delay:Nnn
33     \cs:w #2\cs_end:
34     {#2} {#3}
35   }
36 <latexrelease>\EndIncludeInRelease
37 <latexrelease>\IncludeInRelease{2021/11/15}{\__hook_try_put_cmd_hook:n}%
38 <latexrelease>          {Standardize~generic~hook~names}
39 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
40 <latexrelease>   { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
41 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
42 <latexrelease>   #1 / #2 / #3 / #4 \s__hook_mark #5
43 <latexrelease>   {
44 <latexrelease>     \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }
45 <latexrelease>     \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3}
46 <latexrelease>   }
47 <latexrelease>\EndIncludeInRelease
48 <latexrelease>\IncludeInRelease{2021/06/01}{\__hook_try_put_cmd_hook:n}%
49 <latexrelease>          {Standardize~generic~hook~names}
50 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
51 <latexrelease>   { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
52 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
53 <latexrelease>   #1 / #2 / #3 / #4 \s__hook_mark #5
54 <latexrelease>   {
55 <latexrelease>     \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }
56 <latexrelease>     \str_case:nnTF {#3}
57 <latexrelease>       { { before } { } { after } { } }

```

```

58 \latexrelease { \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
59 \latexrelease { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
60 \latexrelease }
61 \latexrelease \EndIncludeInRelease

```

(End of definition for __hook_try_put_cmd_hook:n and __hook_try_put_cmd_hook:w.)

__hook_patch_cmd_or_delay:Nnn
__hook_cmd_begindocument_code:

In the preamble, __hook_patch_cmd_or_delay:Nnn just adds the patch instruction to a property list to be executed later.

```

62 \cs_new_protected:Npn \__hook_patch_cmd_or_delay:Nnn #1 #2 #3
63 {
64   \__hook_debug:n { \iow_term:n { ->~Add-generic-cmd-hook~for~#2~(#3). } }
65   \__hook_debug:n
66   { \iow_term:n { !~In~the~preamble:~delaying. } }
67   \prop_gput:Nnn \g__hook_delayed_patches_prop { #2 / #3 }
68   { \__hook_cmd_try_patch:nn {#2} {#3} }
69 }

```

The delayed patches are added to a property list to prevent duplication, and the code stored in the property list for each key is executed. The function __hook_patch_cmd_or_delay:Nnn is also redefined to be __hook_patch_command:Nnn so that no further delaying is attempted.

```

70 \cs_new_protected:Npn \__hook_cmd_begindocument_code:
71 {
72   \cs_gset_eq:NN \__hook_patch_cmd_or_delay:Nnn \__hook_patch_command:Nnn
73   \prop_map_function:NN \g__hook_delayed_patches_prop { \use_ii:nn }
74   \prop_gclear:N \g__hook_delayed_patches_prop
75   \cs_undefine:N \__hook_cmd_begindocument_code:
76 }
77 \g@addto@macro \@kernel@after@begindocument
78 { \__hook_cmd_begindocument_code: }

```

(End of definition for __hook_patch_cmd_or_delay:Nnn and __hook_cmd_begindocument_code:.)

__hook_cmd_try_patch:nn

At \begin{document} tries patching the command if the hook was not manually created in the meantime. If the document does not exist, no error is raised here as it may hook into a package that wasn't loaded. Hooks added to commands in the document body still raise an error if the command is not defined.

```

79 \cs_new_protected:Npn \__hook_cmd_try_patch:nn #1 #2
80 {
81   \__hook_debug:n
82   { \iow_term:e { ->~\string\begin{document}~try-cmd / #1 / #2. } }
83   \__hook_if_declared:nTF { cmd / #1 / #2 }
84   {
85     \__hook_debug:n
86     { \iow_term:n { .->~Giving-up:~hook~already~created. } }
87   }
88   {
89     \cs_if_exist:cT {#1}
90     { \exp_args:Nc \__hook_patch_command:Nnn {#1} {#1} {#2} }
91   }
92 }

```

(End of definition for __hook_cmd_try_patch:nn.)

4.4 Patching commands

`\__hook_patch_command:Nnn` `\__hook_patch_command:Nnn` will do some sanity checks on the argument to detect if it is possible to add hooks to the command, and raises an error otherwise. If the command can contain hooks, then it uses `\robust@command@act` to find out what type is the command, and patch it accordingly.

```

93 \cs_new_protected:Npn \__hook_patch_command:Nnn #1 #2 #3
94 {
95   \__hook_patch_debug:e { analyzing~'\token_to_str:N #1' }
96   \__hook_patch_debug:e { \token_to_str:N #1 = \token_to_meaning:N #1 }
97   \__hook_patch_check:NNnn \cs_if_exist:NTF #1 { undef }
98   {
99     \__hook_patch_debug:e { ++control~sequence~is~defined }
100    \__hook_patch_check:NNnn \token_if_macro:NTF #1 { macro }
101    {
102      \__hook_patch_debug:e { ++control~sequence~is~a~macro }
103      \__hook_patch_check:NNnn \__hook_if_public_command:NTF #1 { expl3 }
104      {
105        \__hook_patch_debug:e { ++macro~is~not~private }
106        \robust@command@act
107        \g_hook_patch_action_list_tl #1
108        \__hook_retokenize_patch:Nnn { #1 {#2} {#3} }
109      }
110    }
111  }
112 }

```

And here's the auxiliary used above:

```

113 \cs_new_protected:Npn \__hook_patch_check:NNnn #1 #2 #3 #4
114 {
115   #1 #2 {#4}
116   {
117     \msg_error:nnee { hooks } { cant-patch }
118     { \token_to_str:N #2 } {#3}
119   }
120 }

```

and a conditional `\__hook_if_public_command:NTF` to check if a command has `__` in its name (no other checking is performed). Primitives with `:D` in their name could be included here, but they are already discarded in the `\token_if_macro:NTF` test above.

```

121 \use:e
122 {
123   \prg_new_protected_conditional:Npnn
124   \exp_not:N \__hook_if_public_command:N #1 { TF }
125   {
126     \exp_not:N \exp_last_unbraced:Nf
127     \exp_not:N \__hook_if_public_command:w
128     { \exp_not:N \cs_to_str:N #1 }
129     \tl_to_str:n { _ _ } \s__hook_mark
130   }
131 }
132 \exp_last_unbraced:NNNNo
133 \cs_new_protected:Npn \__hook_if_public_command:w
134 #1 \tl_to_str:n { _ _ } #2 \s__hook_mark

```

```

135 {
136   \tl_if_empty:nTF {#2}
137     { \prg_return_true: }
138     { \prg_return_false: }
139 }

```

(End of definition for `\__hook_patch_command:Nnn` and others.)

4.4.1 Patching by expansion and redefinition

`\g_hook_patch_action_list_tl` This is the list of known command types and the function that patches the command hooks into them. The conditionals are taken from `\ShowCommand`, `\NewCommandCopy` and `\__kernel_cmd_if_xparse:NTF` defined in `ltxcmd`.

```

140 \tl_gset:Nn \g_hook_patch_action_list_tl
141 {
142   { \@if@DeclareRobustCommand \__hook_patch_DeclareRobustCommand:Nnn }
143   { \@if@newcommand \__hook_patch_newcommand:Nnn }
144   { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
145 }

```

(End of definition for `\g_hook_patch_action_list_tl`.)

`\__hook_patch_DeclareRobustCommand:Nnn` At this point we know that the commands can be patched by expanding then redefining. These are the cases of commands defined with `\newcommand` with an optional argument or with `\DeclareRobustCommand`.

With `\__hook_patch_DeclareRobustCommand:Nnn` we check if the command has an optional argument (with a test counter-intuitively called `\@if@newcommand`; also make sure the command doesn't take args by calling `\robust@command@chk@safe`). If so, we pass the patching action to `\__hook_patch_newcommand:Nnn`, otherwise we call the patching engine `\__hook_patch_expand_redefine:NNnn` with a `\c_false_bool` to indicate that there is no optional argument.

```

146 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
147 {
148   \exp_args:Nc \__hook_patch_DeclareRobustCommand_aux:Nnn
149     { \cs_to_str:N #1 ~ }
150 }
151 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand_aux:Nnn #1
152 {
153   \robust@command@chk@safe #1
154   { \@if@newcommand #1 }
155   { \use_i:nn }
156   { \__hook_patch_newcommand:Nnn }
157   { \__hook_patch_expand_redefine:NNnn \c_false_bool }
158   #1
159 }

```

(End of definition for `\__hook_patch_DeclareRobustCommand:Nnn`.)

`\__hook_patch_newcommand:Nnn` If the command was defined with `\newcommand` and an optional argument, call the patching engine with a `\c_true_bool` to flag the presence of an optional argument, and with `\command` to patch the actual code for `\command`.

```

160 \cs_new_protected:Npn \__hook_patch_newcommand:Nnn #1
161 {

```

```

162     \exp_args:Nnc \__hook_patch_expand_redefine:NNnn \c_true_bool
163     { \c_backslash_str \cs_to_str:N #1 }
164 }

```

(End of definition for `\__hook_patch_newcommand:Nnn`.)

`\__hook_cmd_patch_xparse:Nnn` And for commands defined by the xparse commands use this for patching:

```

165 \cs_new_protected:Npn \__hook_cmd_patch_xparse:Nnn #1
166 {
167     \exp_args:Nnc \__hook_patch_expand_redefine:NNnn \c_false_bool
168     { \cs_to_str:N #1 ~ code }
169 }

```

(End of definition for `\__hook_cmd_patch_xparse:Nnn`.)

`\__hook_patch_expand_redefine:NNnn` Now the real action begins. Here we have in `#1` a boolean indicating if the command
`\__hook_redefine_with_hooks:Nnnn` has a leading [...] delimited argument, in `#2` the command control sequence, in `#3` the
`\__hook_make_prefixes:w` name of the command (note that `#1 ≠ \csname#2\endcsname` at this point!), and in `#4`
the hook position, either `before` or `after`.

Patching with expansion+redefinition is trickier than it looks like at first glance. Suppose the simple definition:

```
\def\foo#1{#1#2}
```

When defined, its `<replacement text>` will be a token list containing:

out_param 1, mac_param #, character 2

Then, after expanding `\foo{##1}` (here `##` denotes a single `#6`) we end up with a token list with *out_param 1* replaced:

mac_param #, character 1, mac_param #, character 2

that is, the definition would be:

```
\def\foo#1{#1#2}
```

which obviously fails, because the original input in the definition was `##` but TeX reduced that to a single parameter token `#6` when carrying out the definition. That leaves no room for a clever solution with (say) `\unexpanded`, because anything that would double the second `#6`, would also (incorrectly) double the first, so there's not much to do other than a manual solution.

There are three cases we can distinguish to make things hopefully faster on simpler cases:

1. a macro with no parameters;
2. a macro with no parameter tokens in its definition;
3. a macro with parameters *and* parameter tokens.

The first case is trivial: if the macro has no parameters, we can just use `\unexpanded` around it, and if there is a parameter token in it, it is handled correctly (the macro can be treated as a `\tl` variable).

The second case requires looking at the `<replacement text>` of the macro to see if it has a parameter token in there. If it does not, then there is no worry, and the macro can be redefined normally (without `\unexpanded`).

The third case, as usual, is the devious one. Here we'll have to loop through the definition token by token, and double every parameter token, so that this case can be handled like the previous one.

```

170 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_patch_expand_redefine:NNnn}
171 <latexrelease> {cmd~hooks~with~args}
172 \cs_new_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
173 {
174   \__hook_patch_debug:e { ++~command~can~be~patched~without~rescanning }

```

We'll start by counting the number of arguments in the command by counting the number of characters in the `\cs_parameter_spec:N` of the macro, divided by two, and subtracting one if the command has an optional argument (that is, an extra `[]` in its `<parameter text>`).

```

175   \int_set:Nn \l__hook_patch_num_args_int
176   {
177     \exp_args:Nf \str_count:n { \__kernel_cs_parameter_spec:N #2 } / 2
178     \bool_if:NT #1 { -1 }
179   }

```

Now build two token lists:

`\l__hook_param_text_tl` will contain the `<parameter text>` to be used when redefining the macro. It should be identical to the `<parameter text>` used when originally defining that macro.

`\l__hook_replace_text_tl` will contain braced pairs of `\c__hook_hashes_tl<num>` to feed to the macro when expanded. This token list as well as the previous will have the first item surrounded by `[...]` in the case of an optional argument.

The use of `\c__hook_hashes_tl` here is to differentiate actual parameters in the macro from parameter tokens in the original definition of the macro. Later on, `\c__hook_hashes_tl` is either replaced by actual parameter tokens, or expanded into them.

```

180   \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
181   {

```

We'll first check if the command has any parameter token in its definition (feeding it empty arguments), and set `\__hook_exp_not:n` accordingly. `\__hook_exp_not:n` will be used later to either leave `\c__hook_hashes_tl` or expand it, and also to remember the result of `\__hook_if_has_hash:nTF` to avoid testing twice (the test can be rather slow).

```

182     \tl_set:Ne \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
183     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
184     { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
185     \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
186     { \exp_after:wN #2 \l__hook_tmpa_tl }
187     { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
188     { \cs_set_eq:NN \__hook_exp_not:n \use:n }
189     \cs_set_protected:Npn \__hook_tmp:w ##1 ##2

```

```

190     {
191       ##1 \l__hook_param_text_tl { \use:n ##2 }
192       ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
193     }

```

Here we'll conditionally add [...] around the first parameter:

```

194     \bool_if:NTF #1
195     { \__hook_tmp:w \tl_set:Ne { [ \c__hook_hash_tl 1 ] } }
196     { \__hook_tmp:w \tl_set:Ne { { \c__hook_hash_tl 1 } } }

```

Then, for every parameter from the second, just add it normally:

```

197     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
198     { \__hook_tmp:w \tl_put_right:Ne { { \c__hook_hash_tl ##1 } } }

```

Now, if the command has any parameter token in its definition (then `\__hook_exp_not:n` is `\exp_not:n`), call `\__hook_double_hashes:n` to double them, and replace every `\c__hook_hashes_tl` by `#`:

```

199     \tl_set:Ne \l__hook_replace_text_tl
200     { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
201     \tl_set:Ne \l__hook_replace_text_tl
202     {
203       \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
204       { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
205       { \exp_args:NV \exp_not:o }
206       \l__hook_replace_text_tl
207     }

```

And now, set a few auxiliaries for the case that the macro has parameters, so it won't be passed through `\unexpanded` (twice):

```

208     \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
209     \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
210   }
211   {

```

In the case the macro has no parameters, we'll treat it as a token list and things are much simpler (expansion control looks a bit complicated, but it's just a pair of `\exp_not:n` preventing another `\exp_not:n` from expanding):

```

212     \tl_clear:N \l__hook_param_text_tl
213     \tl_set_eq:NN \l__hook_replace_text_tl #2
214     \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
215     \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
216   }

```

Before redefining, we need to also get the prefixes used when defining the command. Here we ensure that the `\escapechar` is printable, otherwise a macro defined with prefixes `\protected \long` will have it `\meaning` printed as `protectedlong`, making life unnecessarily complicated. Here the `\escapechar` is changed to `/`, then we loop between pairs of `/.../` extracting the prefixes.

```

217     \group_begin:
218     \int_set:Nn \tex_escapechar:D { '/' }
219     \use:e
220     {
221     \group_end:
222     \tl_set:Ne \exp_not:N \l__hook_patch_prefixes_tl
223     { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
224     }

```

Here we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `\__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```

225 \__hook_disable:n { cmd / #3 / #4 }
226 \cs_undefine:c { c__hook_cmd / #3 / #4_parameter_tl }
227 \__hook_make_usable:nn { cmd / #3 / #4 } { \l__hook_patch_num_args_int }

```

Now call `\__hook_redefine_with_hooks:Nnnn` with the macro being redefined in #1, then `\UseHook{cmd/<name>/before}` in #2 or `\UseHook{cmd/<name>/after}` in #3 (one is always empty), and in #4 the `<replacement text>` of the macro.

```

228 \use:e
229 {
230   \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
231   \str_if_eq:nnTF {#4} { after }
232   { \use_ii_i:nn }
233   { \use:nn }
234   { {
235     \__hook_exp_not:NN \exp_not:N \UseHookWithArguments
236     { cmd / #3 / #4 } { \int_use:N \l__hook_patch_num_args_int }
237     \__hook_braced_parameter:n { cmd / #3 / #4 }
238   } }
239   { { } }
240   { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
241 }

```

Finally, update the hook code.

```

242 \__hook_update_hook_code:n { cmd / #3 / #4 }
243 }
244 <latexrelease> \EndIncludeInRelease
245 <latexrelease> \IncludeInRelease{2021/06/01}{\__hook_patch_expand_redefine:NNnn}
246 <latexrelease> {cmd-hooks-with-args}
247 <latexrelease> \cs_gset_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
248 <latexrelease> {
249 <latexrelease>   \__hook_patch_debug:e { ++~command~can~be~patched~without~rescanning }
250 <latexrelease>   \int_set:Nn \l__hook_patch_num_args_int
251 <latexrelease>   {
252 <latexrelease>     \exp_args:Nf \str_count:n { \__kernel_cs_parameter_spec:N #2 } / 2
253 <latexrelease>     \bool_if:NT #1 { -1 }
254 <latexrelease>   }
255 <latexrelease>   \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
256 <latexrelease>   {
257 <latexrelease>     \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
258 <latexrelease>     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
259 <latexrelease>     { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
260 <latexrelease>     \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
261 <latexrelease>     { \exp_after:wN #2 \l__hook_tmpa_tl }
262 <latexrelease>     { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
263 <latexrelease>     { \cs_set_eq:NN \__hook_exp_not:n \use:n }
264 <latexrelease>     \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
265 <latexrelease>     {
266 <latexrelease>       ##1 \l__hook_param_text_tl { \use:n ##2 }
267 <latexrelease>       ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
268 <latexrelease>     }

```



```

269 <latexrelease> \bool_if:NTF #1
270 <latexrelease> { \__hook_tmp:w \tl_set:Nx { [ \c__hook_hash_tl 1 ] } }
271 <latexrelease> { \__hook_tmp:w \tl_set:Nx { { \c__hook_hash_tl 1 } } }
272 <latexrelease> \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
273 <latexrelease> { \__hook_tmp:w \tl_put_right:Nx { { \c__hook_hash_tl ##1 } } }
274 <latexrelease> \tl_set:Nx \l__hook_replace_text_tl
275 <latexrelease> { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
276 <latexrelease> \tl_set:Nx \l__hook_replace_text_tl
277 <latexrelease> {
278 <latexrelease> \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
279 <latexrelease> { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
280 <latexrelease> { \exp_args:NV \exp_not:o }
281 <latexrelease> \l__hook_replace_text_tl
282 <latexrelease> }
283 <latexrelease> \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
284 <latexrelease> \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
285 <latexrelease> }
286 <latexrelease> {
287 <latexrelease> \tl_clear:N \l__hook_param_text_tl
288 <latexrelease> \tl_set_eq:NN \l__hook_replace_text_tl #2
289 <latexrelease> \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
290 <latexrelease> \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
291 <latexrelease> }
292 <latexrelease> \group_begin:
293 <latexrelease> \int_set:Nn \tex_escapechar:D { ‘\’ }
294 <latexrelease> \use:x
295 <latexrelease> {
296 <latexrelease> \group_end:
297 <latexrelease> \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
298 <latexrelease> { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
299 <latexrelease> }
300 <latexrelease> \use:x
301 <latexrelease> {
302 <latexrelease> \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
303 <latexrelease> \str_if_eq:nnTF {#4} { after }
304 <latexrelease> { \use_ii_i:nn }
305 <latexrelease> { \use:nn }
306 <latexrelease> { { \__hook_exp_not:NN \exp_not:N \UseHook { cmd / #3 / #4 } } }
307 <latexrelease> { { } }
308 <latexrelease> { \__hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
309 <latexrelease> }
310 <latexrelease> }
311 <latexrelease> \EndIncludeInRelease

```

Now that all the needed tools are ready, without further ado we'll redefine the command. The definition uses the prefixes gathered in `\l__hook_patch_prefixes_tl`, a primitive `\__hook_def_cmd:w` (which is `\tex_gdef:D` or `\tex_xdef:D`) to avoid adding extra prefixes, and the `<parameter text>` from `\l__hook_param_text_tl`.

Then finally, in the body of the definition, we insert #2, which is `cmd/#1/before` or empty, #4 which is the `<replacement text>`, and #3 which is `cmd/#1/after` or empty.

```

312 \cs_new_protected:Npn \__hook_redefine_with_hooks:Nnnn #1 #2 #3 #4
313 {
314 \l__hook_patch_prefixes_tl
315 \exp_after:wN \__hook_def_cmd:w

```

```

316     \exp_after:wN #1 \l__hook_param_text_tl
317     { #2 #4 #3 }
318 }

```

Here's the auxiliary that makes the prefix control sequences for the redefinition. Each item has to be `\tl_trim_spaces:n`'d because the last item (and not any other) has a trailing space.

```

319 \cs_new:Npn \__hook_make_prefixes:w / #1 /
320 {
321   \tl_if_empty:nF {#1}
322   {
323     \exp_not:c { tex_ \tl_trim_spaces:n {#1} :D }
324     \__hook_make_prefixes:w /
325   }
326 }

```

(End of definition for `\__hook_patch_expand_redefine:NNnn`, `\__hook_redefine_with_hooks:Nnnn`, and `\__hook_make_prefixes:w`.)

Here are some auxiliaries for the contraption above.

`\__hook_if_has_hash_p:n` `\__hook_if_has_hash:nTF` searches the token list #1 for a catcode 6 token, and if any is found, it returns `true`, and `false` otherwise. The searching doesn't care about preserving groups or spaces: we can ignore those safely (braces are removed) so that searching is as fast as possible.

```

327 \prg_new_conditional:Npnn \__hook_if_has_hash:n #1 { TF }
328 { \__hook_if_has_hash:w #1 ## \s_hook_mark }
329 \cs_new:Npn \__hook_if_has_hash:w #1
330 {
331   \tl_if_single_token:nTF {#1}
332   {
333     \token_if_eq_catcode:NNTF ## #1
334     { \__hook_if_has_hash_check:w }
335     { \__hook_if_has_hash:w }
336   }
337   { \__hook_if_has_hash:w #1 }
338 }
339 \cs_new:Npn \__hook_if_has_hash_check:w #1 \s_hook_mark
340 { \tl_if_empty:nTF {#1} { \prg_return_false: } { \prg_return_true: } }

```

(End of definition for `\__hook_if_has_hash:nTF`, `\__hook_if_has_hash:w`, and `\__hook_if_has_hash_check:w`.)

`\__hook_double_hashes:n` `\__hook_double_hashes:w` `\__hook_double_hashes_output:N` `\__hook_double_hashes_output_aux:N` `\__hook_double_hashes_stop:w` `\__hook_double_hashes_group:n` `\__hook_double_hashes_space:w` `\__hook_double_hashes:n` loops through the token list #1 and duplicates any catcode 6 token, and expands tokens `\ifx`-equal to `\c_hook_hashes_tl`, and leaves all other tokens `\notexpanded` with `\exp_not:N`. Unfortunately pairs of explicit catcode 1 and catcode 2 character tokens are normalized to `{_1` and `}_1` because it's not feasible to expandably detect the character code (*maybe* it could be done using something along the lines of <https://tex.stackexchange.com/a/527538>, but it's far too much work for close to zero benefit).

`\__hook_double_hashes:w` is the tail-recursive loop macro, that tests which of the three types of item is in the head of the token list.

```

341 \cs_new:Npn \__hook_double_hashes:n #1
342 { \__hook_double_hashes:w #1 \q_hook_recursion_tail \q_hook_recursion_stop }
343 \cs_new:Npn \__hook_double_hashes:w #1 \q_hook_recursion_stop

```

```

344 {
345   \tl_if_head_is_N_type:nTF {#1}
346   { \__hook_double_hashes_output:N }
347   {
348     \tl_if_head_is_group:nTF {#1}
349     { \__hook_double_hashes_group:n }
350     { \__hook_double_hashes_space:w }
351   }
352   #1 \q__hook_recursion_stop
353 }

\__hook_double_hashes_output:N checks for the end of the token list, then checks
if the token is \c__hook_hashes_tl, and if so just leaves it.
354 \cs_new:Npn \__hook_double_hashes_output:N #1
355 {
356   \if_meaning:w \q__hook_recursion_tail #1
357   \__hook_double_hashes_stop:w
358   \fi:
359   \if:w ?
360     \if_meaning:w \c__hook_hash_tl #1 ! \fi:
361     \if_meaning:w \c__hook_hashes_tl #1 ! \fi:
362     ?
363   \else:
364     \use_i:nnnn
365     \fi:
366     \use:n
367     {
368       \__hook_double_hashes_output_aux:N #1
369       \use_none:n
370       { \exp_not:n { #1 #1 } }
371     }
372     \exp_not:N #1
373     \__hook_double_hashes:w
374   }

If #1 is not \c__hook_hashes_tl, then check if its catcode is 6, and if so, leave it doubled
in \exp_not:n and consume the following \exp_not:N #1.
375 \cs_new:Npn \__hook_double_hashes_output_aux:N #1
376 {
377   \if_catcode:w ## \exp_not:N #1
378   \exp_after:wN \use_ii:nnnn
379   \fi:
380 }
381 \sys_if_engine_luatex:T
382 {
383   \cs_gset:Npn \__hook_double_hashes_output_aux:N #1
384   {

```

(this `\use_i:nnnn` uses `\fi:` and consumes `\use:n`, the whole `\if_catcode:w` block, and the `\exp_not:N`, leaving just `#1` which is `\c__hook_hashes_tl`.)

If both previous tests returned `false`, then leave the token unexpanded and resume the loop.

LuaTeX has a primitive equivalent to `#`, which is replaced in a dedicated path. That code is not needed for other engines, so we avoid the hit on performance by using an auxiliary.

```

385     \if_catcode:w ## \exp_not:N #1
386     \exp_after:wN \use_ii:nnnn
387   \else:
388     \if_meaning:w \tex_alignmark:D \tex_alignmark:D #1
389     \exp_after:wN \exp_after:wN \exp_after:wN \use_ii:nnnn
390   \fi:
391 \fi:
392 }
393 }
394 \cs_new:Npn \__hook_double_hashes_stop:w #1 \q__hook_recursion_stop { \fi: }
    Dealing with spaces and grouped tokens is trivial:
395 \cs_new:Npn \__hook_double_hashes_group:n #1
396   { { \__hook_double_hashes:n {#1} } \__hook_double_hashes:w }
397 \exp_last_unbraced:NNo
398 \cs_new:Npn \__hook_double_hashes_space:w \c_space_tl
399   { ~ \__hook_double_hashes:w }

```

(End of definition for `\__hook_double_hashes:n` and others.)

4.4.2 Patching by retokenization

At this point we’ve drained the possibilities of patching a command by expansion-and-redefinition, so we have to resort to patching by retokenizing the command. Patching by retokenization is done by getting the `\meaning` of the command, doing the necessary manipulations on the generated string, and the retokenizing that again by using `\scantokens`.

Patching by retokenization is definitely a riskier business, because it relies that the tokens printed by `\meaning` produce the exact same tokens as the ones in the original definition. That is, the catcode régime must be exactly(ish) the same, and there is no way of telling except by trial and error.

`\__hook_retokenize_patch:Nnn` This is the macro that will control the whole process. First we’ll try out one final, rather trivial case, of a command with no arguments; that is, a token list. This case can be patched with the expand-and-redefine routine but it has to be the very last case tested for, because most (all?) robust commands start with a top-level macro with no arguments, so testing this first would short-circuit `\robust@command@act` and the top-level macros would be incorrectly patched. In that case, we just check if the `\cs_parameter_spec:N` is empty, and call `\__hook_patch_expand_redefine:NNnn`.

```

400 \cs_new_protected:Npn \__hook_retokenize_patch:Nnn #1 #2 #3
401   {
402     \str_if_eq:eeTF { \__kernel_cs_parameter_spec:N #1 } { }
403     { \__hook_patch_expand_redefine:NNnn \c_false_bool #1 {#2} {#3} }
404     {
405       \__hook_patch_debug:e { ..~command~can~only~be~patched~by~rescanning }

```

Otherwise, we start the actual patching by retokenization job. The code calls `\__hook_try_patch_with_catcodes:Nnnnw` with a different catcode setting:

- The current catcode setting;
- Switching the catcode of `@`;
- Switching the `expl3` syntax on or off;

- Both of the above.

If patching succeeds, `\__hook_try_patch_with_catcodes:Nnnnw` has the side-effect of patching the macro #1 (which may be an internal from the command whose name is #2).

```

406     \tl_set:Ne \l__hook_tmpa_tl
407     {
408         \int_compare:nNnTF { \char_value_catcode:n {'\@ } } = { 12 }
409         { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
410     }
411     \tl_set:Ne \l__hook_tmpb_tl
412     {
413         \bool_if:NTF \l__kernel_expl_bool
414         { \ExplSyntaxOff }
415         { \ExplSyntaxOn \char_set_catcode_space:n { 32 }}
416     }
417     \use:e
418     {
419         \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
420         \exp_not:n { #1 {#2} {#3} }
421         { \prg_do_nothing: }
422         { \exp_not:V \l__hook_tmpa_tl } % @
423         { \exp_not:V \l__hook_tmpb_tl } % _:
424         {
425             \exp_not:V \l__hook_tmpa_tl    % @
426             \exp_not:V \l__hook_tmpb_tl    % _:
427         }
428     }
429     \q_recursion_tail \q_recursion_stop

```

If no catcode setting succeeds, give up and raise an error. The command isn't changed in any way in that case.

```

430     {
431         \msg_error:nnee { hooks } { cant-patch }
432         { \c_backslash_str #2 } { retok }
433     }
434 }
435 }

```

(End of definition for `\__hook_retokenize_patch:Nnn`.)

`\__hook_try_patch_with_catcodes:Nnnnw`

This function is a simple wrapper around `\__hook_cmd_if_scanable:NnTF` and `\__hook_patch_retokenize:Nnnn` if the former returns `<true>`, plus some debug messages.

```

436 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
437 <latexrelease> {cmd-hooks-with-args}
438 \cs_new_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
439 {
440     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
441     \__hook_patch_debug:e { ++~trying~to~patch~by~retokenization }
442     \__hook_cmd_if_scanable:NnTF {#1} {#4}
443     {
444         \__hook_patch_debug:e { ++~macro~can~be~retokenized~cleanly }
445         \__hook_patch_debug:e { ==~retokenizing~macro~now }
446         \__hook_patch_retokenize:Nnnn #1 { cmd / #2 / #3 } {#3} {#4}

```

```

447 \use_i_delimit_by_q_recursion_stop:nw \use_none:n
448 }
449 {
450 \__hook_patch_debug:e { ---macro-cannot-be-retokenized-cleanly }
451 \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
452 }
453 }
454 \latexrelease\EndIncludeInRelease
455 \latexrelease\IncludeInRelease{2021/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
456 \latexrelease{cmd-hooks-with-args}
457 \latexrelease\cs_gset_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
458 \latexrelease{
459 \latexrelease\quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
460 \latexrelease\__hook_patch_debug:e { +-~trying~to~patch~by~retokenization }
461 \latexrelease\__hook_cmd_if_scanable:NnTF {#1} {#4}
462 \latexrelease{
463 \latexrelease\__hook_patch_debug:e { +-~macro-can-be-retokenized-cleanly }
464 \latexrelease\__hook_patch_debug:e { ==~retokenizing-macro-now }
465 \latexrelease\__hook_patch_retokenize:Nnnn #1 {#2} {#3} {#4}
466 \latexrelease\use_i_delimit_by_q_recursion_stop:nw \use_none:n
467 \latexrelease}
468 \latexrelease{
469 \latexrelease\__hook_patch_debug:e { ---macro-cannot-be-retokenized-cleanly }
470 \latexrelease\__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
471 \latexrelease}
472 \latexrelease}
473 \latexrelease\EndIncludeInRelease

```

(End of definition for __hook_try_patch_with_catcodes:Nnnnw.)

\kerneltmpDoNotUse This is an oddity required to be safe (as safe as reasonably possible) when patching the command. The entirety of

$\langle\text{prefixes}\rangle$ \def $\langle\text{cs}\rangle$ $\langle\text{parameter text}\rangle$ { $\langle\text{replacement text}\rangle$ }

will go through \scantokens. The $\langle\text{parameter text}\rangle$ and $\langle\text{replacement text}\rangle$ are what we are trying to retokenize, so not much worry there. The other items, however, should “just work”, so some care is needed to not use too fancy catcode settings. Therefore we can’t use an expl3-named macro for $\langle\text{cs}\rangle$, nor the expl3 versions of \def or the $\langle\text{prefixes}\rangle$. That is why the definitions that will eventually go into \scantokens will use the oddly (but hopefully clearly)-named \kerneltmpDoNotUse:

```

474 \cs_new_eq:NN \kerneltmpDoNotUse !

```

PhO: Maybe this can be avoided by running the $\langle\text{parameter text}\rangle$ and the $\langle\text{replacement text}\rangle$ separately through \scantokens and then putting everything together at the end.

(End of definition for \kerneltmpDoNotUse.)

__hook_patch_required_catcodes: Here are the catcode settings that are *mandatory* when retokenizing commands. These are the minimum necessary settings to perform the definitions: they identify control sequences, which must be escaped with _0, delimit the definition with {_1 and }_2, and mark parameters with #_6. Everything else may be changed, but not these.

```

475 \cs_new_protected:Npn \__hook_patch_required_catcodes:
476 {

```

```

477 \char_set_catcode_escape:N \\\
478 \char_set_catcode_group_begin:N \{
479 \char_set_catcode_group_end:N \}
480 \char_set_catcode_parameter:N \#
481 % \int_set:Nn \tex_endlinechar:D { -1 }
482 % \int_set:Nn \tex_newlinechar:D { -1 }
483 }

```

PhO: etoolbox sets the \endlinechar and \newlinechar when patching, but as far as I tested these didn't make much of a difference, so I left them out for now. Maybe \newlinechar=-1 avoids a space token being added after the definition.

PhO: If the patching is split by ⟨parameter text⟩ and ⟨replacement text⟩, then only # will have to stay in that list.

PhO: Actually now that we patch \UseHook{cmd/foo/before}, all the tokens there need to have the right catcodes, so this list now includes all lowercase letters, U and H, the slash, and whatever characters in the command name... sigh...

(End of definition for __hook_patch_required_catcodes:.)

__hook_cmd_if_scanable:NnTF Here we'll do a quick test if the command being patched can in fact be retokenized with the specific catcode setting without changing in meaning. The test is straightforward:

1. apply \meaning to the command;
2. split the ⟨prefixes⟩, ⟨parameter text⟩ and ⟨replacement text⟩ and arrange them as

```

⟨prefixes⟩\def\kerneltmpDoNotUse⟨parameter text⟩{⟨replacement text⟩}

```

3. rescan that with the given catcode settings, and do the definition; then finally
4. compare \kerneltmpDoNotUse with the original command.

If both are \ifx-equal, the command can be safely patched.

```

484 \prg_new_protected_conditional:Npnn \__hook_cmd_if_scanable:Nn #1 #2 { TF }
485 {
486   \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
487   \cs_set_eq:NN \__hook_tmp:w \scan_stop:
488   \use:e
489   {
490     \cs_set:Npn \__hook_tmp:w
491       ##1 \tl_to_str:n { macro: } ##2 -> ##3 \s__hook_mark
492     { ##1 \def \kerneltmpDoNotUse ##2 {##3} }
493     \tl_set:Ne \exp_not:N \l__hook_tmpa_tl
494     { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
495   }
496   \tl_rescan:nV { #2 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
497   \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
498   { \prg_return_true: }
499   { \prg_return_false: }
500 }

```

(End of definition for __hook_cmd_if_scanable:NnTF.)

`\__hook_guess_arg_count:NN` Looks at the parameter text of a macro, and counts the parameters by looking at the
`\__hook_guess_arg_count:wN` number after a #, and checking if they are sequential. This macro assumes that all
`\__hook_guess_arg_count:nw` parameters are marked with hashes, and not other characters, and that there is no “trick
parameter”.

```

501 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_guess_arg_count:NN}
502 <latexrelease>{cmd~hooks~with~args}
503 \cs_new_protected:Npn \__hook_guess_arg_count:NN #1
504 {
505   \exp_after:wN \__hook_guess_arg_count:wN
506   \token_to_meaning:N #1 \s__hook_mark
507 }
508 \exp_last_unbraced:NNNN
509 \cs_new_protected:Npe \__hook_guess_arg_count:wN
510 #1 { \tl_to_str:n { macro: } } #2 \s__hook_mark #3
511 {
512   \int_set:Nn #3
513   {
514     \exp_not:N \__hook_guess_arg_count:nw { 0 } #2
515     \c_hash_str 0 \s__hook_mark
516   }
517 }
518 \use:e
519 { \cs_new:Npn \exp_not:N \__hook_guess_arg_count:nw #1 #2 \c_hash_str #3 }
520 {
521   \int_compare:nNnTF { #1 + 1 } = {#3}
522   { \__hook_guess_arg_count:nw {#3} }
523   { #1 \__hook_use_none_delimit_by_s_mark:w }
524 }
525 <latexrelease>\EndIncludeInRelease
526 <latexrelease>\IncludeInRelease{2021/06/01}{\__hook_guess_arg_count:NN}
527 <latexrelease>{cmd~hooks~with~args}
528 <latexrelease>\cs_undefine:N \__hook_guess_arg_count:NN
529 <latexrelease>\EndIncludeInRelease

(End of definition for \__hook_guess_arg_count:NN, \__hook_guess_arg_count:wN, and
 \__hook_guess_arg_count:nw.)

```

`\__hook_patch_retokenize:Nnnn` Then, if `\__hook_cmd_if_scanable:NnTF` returned true, we can go on and patch the
command.

```

530 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_patch_retokenize:Nnnn}
531 <latexrelease>{cmd~hooks~with~args}
532 \cs_new_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
533 {

```

Here, when patching by retokenization, we can only guess the number of arguments of
the macro.

```

534   \__hook_guess_arg_count:NN #1 \l__hook_patch_num_args_int

```

Then we redefine the hook to have the right number of arguments. Disabling the
hook, undefining the `parameter` token list then calling `\__hook_make_usable:nn` are
enough to redefine the hook to the extent we want. Code stored in the hook and other
metadata about it are not lost in the process.

```

535   \__hook_disable:n {#2}
536   \cs_undefine:c { c__hook_#2_parameter_tl }

```



```

537 \__hook_make_usable:nn {#2} { \l__hook_patch_num_args_int }
538 \tl_set:Ne \l__hook_tmpa_tl
539 { \exp_args:Ne \tl_to_str:n { \__hook_braced_parameter:n {#2} } }
540 \use:e
541 {
542     \str_replace_all:Nnn \exp_not:N \l__hook_tmpa_tl
543     { ## } { \c_hash_str }
544 }

```

Then, make some things `\relax` to avoid lots of `\noexpand` below.

```

545 \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
546 \cs_set_eq:NN \__hook_tmp:w \scan_stop:
547 \use:e
548 {

```

Now we'll define `\__hook_tmp:w` such that it splits the `\meaning` of the macro (`#1`) into its three parts:

```

####1. <prefixes>

####2. <parameter text>

####3. <replacement text>

```

and arrange that a complete definition, then place the `before` or `after` hooks around the `<replacement text>`: accordingly.

```

549 \cs_set:Npn \__hook_tmp:w
550   ##1 \tl_to_str:n { macro: } ##2 -> ##3 \s__hook_mark
551 {
552   ##1 \def \kerneltmpDoNotUse ##2
553   {
554     \str_if_eq:nnT {#3} { before }
555     {
556       \token_to_str:N \UseHookWithArguments {#2}
557       { \int_use:N \l__hook_patch_num_args_int }
558       \l__hook_tmpa_tl
559     }
560     ##3
561     \str_if_eq:nnT {#3} { after }
562     {
563       \token_to_str:N \UseHookWithArguments {#2}
564       { \int_use:N \l__hook_patch_num_args_int }
565       \l__hook_tmpa_tl
566     }
567   }
568 }

```

Now we just have to get the `\meaning` of the command being patched and pass it through the meat grinder above.

```

569 \tl_set:Ne \exp_not:N \l__hook_tmpa_tl
570 { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
571 }

```

Now rescan with the given catcode settings (overridden by the `\__hook_patch_required_catcodes:`), and implicitly (by using the rescanned token list) carry out the definition from above.

```

572 \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl

```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```
573 \cs_gset_eq:NN #1 \kerneltmpDoNotUse
```

Finally, update the hook code.

```
574 \__hook_update_hook_code:n {#2}
575 }
576 <latexrelease> \EndIncludeInRelease
577 <latexrelease> \IncludeInRelease{2021/06/01}{\__hook_patch_retokenize:Nnnn}
578 <latexrelease> {cmd-hooks-with-args}
579 <latexrelease> \cs_gset_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
580 <latexrelease> {
581 <latexrelease> \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
582 <latexrelease> \cs_set_eq:NN \__hook_tmp:w \scan_stop:
583 <latexrelease> \use:x
584 <latexrelease> {
585 <latexrelease> \cs_set:Npn \__hook_tmp:w
586 <latexrelease> #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
587 <latexrelease> {
588 <latexrelease> #####1 \def \kerneltmpDoNotUse #####2
589 <latexrelease> {
590 <latexrelease> \str_if_eq:nnT {#3} { before }
591 <latexrelease> { \token_to_str:N \UseHook { cmd / #2 / #3 } }
592 <latexrelease> #####3
593 <latexrelease> \str_if_eq:nnT {#3} { after }
594 <latexrelease> { \token_to_str:N \UseHook { cmd / #2 / #3 } }
595 <latexrelease> }
596 <latexrelease> }
597 <latexrelease> \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
598 <latexrelease> { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
599 <latexrelease> }
600 <latexrelease> \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
601 <latexrelease> \cs_gset_eq:NN #1 \kerneltmpDoNotUse
602 <latexrelease> }
603 <latexrelease> \EndIncludeInRelease
```

(End of definition for __hook_patch_retokenize:Nnnn.)

4.5 Messages

```
604 <latexrelease> \IncludeInRelease{2023/06/01}{wrong-cmd-hook}%
605 <latexrelease> {Standardize-generic-hook-names}
606 <latexrelease> \EndIncludeInRelease
607 <latexrelease> \IncludeInRelease{2021/06/01}{wrong-cmd-hook}%
608 <latexrelease> {Standardize-generic-hook-names}
609 <latexrelease> \msg_new:nnnn { hooks } { wrong-cmd-hook }
610 <latexrelease> {
611 <latexrelease> Generic-hook~‘cmd/#1/#2’~is~invalid.
612 <latexrelease> % The~hook~should~be~‘cmd/#1/before’~or~‘cmd/#1/after’.
613 <latexrelease> }
614 <latexrelease> {
615 <latexrelease> You~tried~to~add~a~generic~hook~to~command~\iow_char:N \#1,~but~‘#2’~
616 <latexrelease> is~an~invalid~component.~Only~‘before’~or~‘after’~are~allowed.
617 <latexrelease> }
```

```

618 <latexrelease>\EndIncludeInRelease
619 \msg_new:nnnn { hooks } { cant-patch }
620 {
621   Generic-hooks~cannot-be-added-to~'#1'.
622 }
623 {
624   You~tried~to~add~a~hook~to~'#1',~but~LaTeX~was~unable~to~
625   patch~the~command~because~it~\__hook_unpatchable_cases:n {#2}.
626 }
627 \cs_new:Npn \__hook_unpatchable_cases:n #1
628 {
629   \str_case:nn {#1}
630   {
631     { undef } { doesn't~exist }
632     { macro } { is~not~a~macro }
633     { expl3 } { is~a~private~expl3~macro }
634     { retok } { can't~be~retokenized~cleanly }
635   }
636 }
637 <latexrelease>\IncludeInRelease{0000/00/00}{ltxcmdhooks}%
638 <latexrelease>           {The~hook~management~system~for~commands}
639 <latexrelease>

```

The command `\__hook_cmd_begindocument_code:` is used in an internal hook, so we need to make sure it has a harmless definition after rollback as that will not remove it from the kernel hook.

```

640 <latexrelease>\cs_set_eq:NN \__hook_cmd_begindocument_code: \prg_do_nothing:
641 <latexrelease>
642 <latexrelease>\EndModuleRelease
643 \ExplSyntaxOff
644 </2ekernel | latexrelease>
645 <@@=>

```

File 10

ltsockets.dtx

Abstract

This code implements sockets which are places in the code into which predeclared chunks of code (plugs) can be placed. Both the sockets and the plugs are “named” and each socket is assigned exactly one plug at any given time.

1 Introduction

A L^AT_EX source file is transformed into a typeset document by executing code for each command or environment in the document source. Through various steps this code transforms the input and eventually generates typeset output appearing in a “galley” from which individual pages are cut off in an asynchronous way. This page generating process is normally not directly associated with commands in the input²¹ but is triggered whenever the galley has received enough material to form another page (giving current settings).

As part of this transformation input data may get stored in some form and later reused, for example, as part of the output routine processing.

2 Configuration of the transformation process

There are three different major methods offered by L^AT_EX to configure the transformation process:

- through the template mechanism,
- through the hook mechanism, or
- through sockets and plugs.

They offer different possibilities (with different features and limitations) and are intended for specific use cases, though it is possible to combine them.

2.1 The template mechanism

The template mechanism is intended for more complex document-level elements (e.g., headings such as `\section` or environments like `itemize`). The template code implements the overall processing logic for such an element and offers a set of parameters to influence the final result.

The document element is then implemented by (a) selecting a suitable template (there may be more than one available for the kind of document element) and (b) by setting its parameters to desired values. This then forms a so-called instance which is executed when the document element is found in the source.

By altering the parameter values (in a document class or in the document preamble) or, if more drastic layout changes are desired, by selecting a different template and

²¹Excepts for directives such as `\newpage`.

then adjusting its parameters, a wide variety of layouts can be realized through simple configuration setups without the need to develop new code.

The target audience of this method are therefore document class developers or users who wish to alter an existing layout (implemented by a document class) in certain (minor) ways.

The template mechanism is currently documented as part of the `xtemplate` package and one more elaborate implementation can be found as part of the `latex-lab` code for lists (to be documented further).

2.2 The hook mechanism

Hooks are places in the kernel code (or in packages) that offer packages the possibility to inject additional code at specific points in the processing in a controlled way without the need to replace the existing code block (and thereby overwriting modifications/extensions made by other packages). The target audience is therefore mainly package developers, even though some hooks can be useful for document authors.

Obviously, what can reasonably be added into a hook depends on the individual hook (hopefully documented as part of the hook documentation), but in general the idea behind hooks is that more than one package could add code into the hook at the same time. Perhaps the most famous hook (that L^AT_EX had for a very long time) is `begindocument` into which many packages add code to through `\AtBeginDocument{<code>}` (which is nowadays implemented as a shorthand for `\AddToHook{begindocument}{<code>}`). To resolve possible conflicts between injections by different packages there is a rule mechanism by which code chunks in a hook can be ordered in a certain way and by which incompatible packages can be detected if a resolution is impossible.

In contrast to template code, there is no standard configuration method through parameters for hooks, i.e., the code added to a hook “is” the configuration. If it wants to provide for configuration through parameters it has to also provide its own method to set such parameters in some way. However, in that case it is likely that using a hook is not the right approach and the developer better calls a template instance instead which then offers configuration through a key/value interface.

In most cases, hooks do not take any arguments as input. Instead, the data that they can (and are allowed to) access depends on the surrounding context.

For example, the various hooks available during the page shipout process in L^AT_EX’s output routine can (and have to) access the accumulated page material stored in a box named `\ShipoutBox`. This way, code added to, say, the `shipout/before` hook could access the page content, alter it, and then write it back into `\ShipoutBox` and any other code added to this hook could then operate on the modified content. Of course, for such a scheme to work the code prior to executing the hook would need to setup up data in appropriate places and the hook documentation would need to document what kind of storage can be accessed (and possibly altered) by the hook.

There are also hooks that take arguments (typically portions of document data) and in that case the hook code can access these arguments through `#1`, `#2`, etc.

The hook mechanism is documented in `lthooks-doc.pdf`.

2.3 The socket mechanism

In some cases there is code that implements a certain programming logic (for example, combining footnotes, floats, and the text for the current page to be shipped out) and

if this logic should change (e.g., footnotes to be placed above bottom floats instead of below) then this whole code block needs to be replaced with different code.

In theory, this could be implemented with templates, i.e., the code simply calls some instance that implements the logic and that instance is altered by selecting a different templates and/or adjusting their parameters. However, in many cases customization through parameters is overkill in such a case (or otherwise awkward, because parameterization is better done on a higher level instead of individually for small blocks of code) and using the template mechanism just to replace one block of code with a different one results in a fairly high performance hit. It is therefore usually not a good choice.

In theory, it would also be possible to use a hook, but again that is basically a misuse of the concept, because in this use case there should never be more than one block of code inside the hook; thus, to alter the processing logic one would need to set up rules that replace code rather than (as intended) execute all code added to the hook.

For this reason L^AT_EX now offers a third mechanism: “sockets” into which one can place exactly one code block — a “plug”.

In a nutshell: instead of having a fixed code block somewhere as part of the code, implementing a certain programming logic there is a reference to a named socket at this point. This is done by first declaring the named socket with:

```
\NewSocket{<socket-name>}{<number-of-inputs>}
```

This is then referenced at the point where the replaceable code block should be executed with:

```
\UseSocket{<socket-name>}
```

or, if the socket should take a number of inputs (additional arguments beside the name) with

```
\UseSocket{<socket-name>}{<arg1>}...{<argnumber-of-inputs>}}
```

In addition, several code blocks (a.k.a. plugs) implementing different logic for this socket are set up, each with a declaration of the form:

```
\NewSocketPlug{<socket-name>}{<socket-plug-name>}{<code>}
```

Finally, one of them is assigned to the socket:

```
\AssignSocketPlug{<socket-name>}{<socket-plug-name>}
```

If the programming logic should change, then all that is necessary is to make a new assignment with `\AssignSocketPlug` to a different `{<socket-plug-name>}`. This assignment obeys scope so that an environment can alter a socket without the need to restore the previous setting manually.

If the socket takes inputs, then those need to be provided to `\UseSocket` and in that case they can be referenced in the `<code>` argument of `\NewSocketPlug` with `#1`, `#2`, etc.

In most cases a named socket is used only in a single place, but there is, of course, nothing wrong with using it in several places, as long as the code in all places is supposed to change in the same way.

2.3.1 Examples

We start by declaring a new socket named `foo` that expects two inputs:

```
\NewSocket{foo}{2}
```

Such a declaration has to be unique across the whole L^AT_EX run. Thus, if another package attempts to use the same name (regardless of the number of inputs) it will generate an error:

```
\NewSocket{foo}{2}
\NewSocket{foo}{1}
```

Both declarations would therefore produce:

```
! LaTeX socket Error: Socket 'foo' already declared!
```

You also get an error if you attempt to declare some socket plug and the socket name is not yet declared, e.g.,

```
\NewSocketPlug{baz}{undeclared}{some code}
```

generates

```
! LaTeX socket Error: Socket 'baz' undeclared!
```

Setting up plugs for the socket is done like this:

```
\NewSocketPlug{foo}{plug-A}
  {\begin{quote}\itshape foo-A: #1!#2\end{quote}}
\NewSocketPlug{foo}{plug-B}
  {\begin{quote}\sffamily foo-B: #2\textsuperscript{2}\end{quote}}
```

This will set up the plugs `plug-A` and `plug-B` for this socket.

We still have to assign one or the other to the socket, thus without doing that the line

```
\UseSocket{foo}{hello}{world}
```

produces nothing because the default plug for sockets with 2 inputs is `noop` (which grabs the additional arguments and throws them away).²²

So let's do the assignment

```
\AssignSocketPlug{foo}{plug-A}
```

and then

```
\UseSocket{foo}{hello}{world}
```

will properly typeset

foo-A: hello!world

and after

```
\AssignSocketPlug{foo}{plug-B}
```

²²If socket `foo` would have been a socket with one input, then the default plug would be `identity`, in which case the socket input would remain without braces and gets typeset!

and another call to

```
\UseSocket{foo}{hello}{world}
```

we get

```
foo-B: world2
```

If we attempt to assign a plug that was not defined, e.g.,

```
\AssignSocketPlug{foo}{plug-C}
```

then we get an error during the assignment

```
! LaTeX socket Error: Plug 'plug-C' for socket 'foo' undeclared!
```

and the previous assignment remains in place.

To see what is known about a socket and its plugs you can use `\ShowSocket` or `\LogSocket` which displays information similar to this on the terminal or in the transcript file:

```
Socket foo:
  number of inputs = 2
  available plugs = noop, plug-A, plug-B
  current plug = plug-B
  definition = \long macro:#1#2->\begin {quote}\sffamily
foo-B: #2\textsuperscript {2}\end {quote}
```

2.3.2 Details and semantics

In this section we collect some normative statements.

- From a functional point of view sockets are like simple `TeX` macros, i.e., they expect 0 to 9 mandatory arguments (the socket inputs) and get replaced by their “expansion”
- A socket is “named” and the name consists of ASCII letters `[a-z]`, `[A-Z]`, `[0-9]`, `[-/@]` only
- Socket names have to be unique, i.e., there can be only one socket named `<name>`. This is ensured by declaring each socket with `\NewSocket`.

However, there is no requirement that sockets and hook names have to be different. In fact, if a certain action that could otherwise be specified as hook code has to be executed always last (or first) one could ensure this by placing a socket (single action) after a hook (or vice versa) and using the same name to indicate the relationship, e.g.,

```
\UseHook{foo}          % different package can add code here
\UseSocket{foo}         % only one package can assign a plug
```

This avoids the need to order the hook code to ensure that something is always last.

- Best practice naming conventions are ... *to be documented*

- A socket has documented inputs which are
 - the positional arguments (if any) with a description of what they contain when used
 - implicit data (registers and other 2e/expl3 data stores) that the socket is allowed to make use of, with a documented description of what they contain (if relevant for the task at hand—no need to describe the whole L^AT_EX universe)
 - information about the state of the T_EX engine (again when relevant), e.g. is called in mmode or vmode or in the output routine or ...
 - ... anything missing?
- A socket has documented results/outputs which can be
 - what kind of data it should write to the current list (if that is part of its task)
 - what kind of registers and other 2e/expl3 data stores it should modify and in what way
 - what kind of state changes it should do (if any)
 - ... *anything else*?
- At any time a socket has one block of code (a plug :-)) associated with it. Such code is itself named and the association is done by linking the socket name to the code name (putting a plug into the socket).
- The name of a plug consists of ASCII letters [a-z], [A-Z], [0-9], [-/@] only.
- Socket plug names have to be unique within on a per socket basis, but it is perfectly allowed (and sensible in some cases) to use the same plug name with different sockets (where based on the sockets' purposes, different actions may be associated with the plug name). For example `noop` is a plug name declared for every socket, yet its action “grab the socket inputs and throw them away” obviously differs depending on how many inputs the socket has.
- When declaring a plug it is stated for which socket it is meant (i.e., its code can only be used with that socket). This means that the same plug name can be used with different sockets referring to different code in each case.
- Configuration of a socket can only be done by linking different code to it. Nevertheless the code linked to it can provide its own means of configuration (but this is outside of the spec).
- Technically execution of a socket (`\UseSocket`) involves
 - doing any house keeping (like writing debugging info, ...);
 - looking up the current code association (what plug is in the socket);
 - executing this code which will pick up the mandatory arguments (happens at this point, not before), i.e., it is like calling a csname defined with


```
\def\foo#1#2...{\...#1...#2...}
```
 - do some further house keeping (if needed).
- A socket is typically only used in one place in code, but this is not a requirement, i.e., if the same operation with the same inputs need to be carried out in several places the same named socket can be used.

2.4 Socket and plug names

The $\langle socket-name \rangle$ and $\langle socket-plug-name \rangle$ are always expanded using T_EX's `\csname... \endcsname`, as such, (expandable) commands and Unicode characters are allowed in $\langle hook \rangle$ and $\langle label \rangle$ arguments.

2.4.1 Command syntax

We give both the L^AT_EX 2_ε and the L3 programming layer command names.

<code>\NewSocket</code>	<code>\NewSocket</code>	$\{\langle socket-name \rangle\}$	$\{\langle number-of-inputs \rangle\}$
<code>\socket_new:nn</code>	<code>\socket_new:nn</code>	$\{\langle socket-name \rangle\}$	$\{\langle number-of-inputs \rangle\}$

Declares a new socket with name $\langle socket-name \rangle$ having $\langle number-of-inputs \rangle$ inputs. There is automatically a plug `noop` declared for it, which does nothing, i.e., it gobbles the socket inputs (if any). This is made the default plug except for sockets with one input which additionally define the plug `identity` and assign that as their default.

This `identity` plug simply returns the socket input without its outer braces. The use case for this plug are situations like this:

`\UseSocket{taggsupport/footnote}{\code}`

If tagging is not active and the socket contains the plug `identity` then this returns $\langle code \rangle$ without the outer braces and to activate tagging all that is necessary is to change the plug to say `tagpdf` so that it surrounds $\langle code \rangle$ by some tagging magic. This is the most common use case for sockets with one input, which is why they have this special default.

The socket documentation should describe its purpose, its inputs and the expected results as discussed above.

The declaration is only allowed at top-level, i.e., not inside a group.

<code>\NewSocketPlug</code>	<code>\NewSocketPlug</code>	$\{\langle socket-name \rangle\}$	$\{\langle socket-plug-name \rangle\}$	$\{\langle code \rangle\}$
<code>\socket_new_plug:nnn</code>	<code>\socket_new_plug:nnn</code>	$\{\langle socket-name \rangle\}$	$\{\langle socket-plug-name \rangle\}$	$\{\langle code \rangle\}$
<code>\socket_set_plug:nnn</code>	<code>\socket_set_plug:nnn</code>	$\{\langle socket-name \rangle\}$	$\{\langle socket-plug-name \rangle\}$	$\{\langle code \rangle\}$

Declares a new plug for socket $\langle socket-name \rangle$ that runs $\langle code \rangle$ when executing. It complains if the plug was already declared previously.

The form `\socket_set_plug:nnn` changes an existing plug. As this should normally not be necessary, we currently have only an L3 layer name for the few cases it might be useful.

The declarations can be made inside a group and obey scope, i.e., they vanish if the group ends.

<code>\AssignSocketPlug</code>	<code>\AssignSocketPlug</code>	$\{\langle socket-name \rangle\}$	$\{\langle socket-plug-name \rangle\}$
<code>\socket_assign_plug:nn</code>	<code>\socket_assign_plug:nn</code>	$\{\langle socket-name \rangle\}$	$\{\langle socket-plug-name \rangle\}$

Assigns the plug $\langle socket-plug-name \rangle$ to the socket $\langle socket-name \rangle$. It errors if either socket or plug is not defined.

The assignment is local, i.e., it obeys scope.

<hr/>	<hr/>
<code>\UseSocket</code>	<code>\UseSocket {<socket-name>}</code>
<code>\socket_use:nw</code>	<code>\socket_use:nnn {<socket-name>} {<socket-arg₁>} {<socket-arg₂>}</code>
<code>\socket_use:n</code>	Executes the socket <code><socket-name></code> by retrieving the <code><code></code> of the current plug assigned to the socket. This is the only command that would appear inside macro code in packages.
<code>\socket_use:nn</code>	For performance reasons there is no explicit check that the socket was declared!
<code>\socket_use:nnn</code>	The different L3 programming layer commands are really doing the same thing: they grab as many arguments as defined as inputs for the socket and then pass them to the plug. The different names are only there to make the code more readable, i.e., to indicate how many arguments are grabbed in total (note that no runtime check is made to verify that this is actually true). We only provide them for sockets with up to 3 inputs (most likely those with zero or one input would have been sufficient). If you happen to have a socket with more inputs, use <code>\socket_use:nw</code> .
<code>\socket_use:nnnn</code>	

<hr/>	<hr/>
<code>\socket_use_expandable:nw</code>	<code>* \socket_use_expandable:n {<socket-name>}</code>
<code>\socket_use_expandable:n</code>	<code>*</code>

Fully expandable variant of `\socket_use:n`. This can be used in macro code to retrieve code from sockets which need to appear in an expandable context.

This usually requires the plug to only contain expandable code and should therefore only be used for sockets which are clearly documented to be used in an expandable context. This command does not print any debugging info when `\DebugSocketsOn` is active and should therefore be avoided whenever possible.

For performance reasons there is no explicit check that the socket was declared!

<hr/>	<hr/>
<code>\ShowSocket</code>	<code>\ShowSocket {<socket-name>}</code>
<code>\LogSocket</code>	<code>\socket_show:n {<socket-name>}</code>
<code>\socket_show:n</code>	Displays information about the socket <code><socket-name></code> and its state then stops and waits for further instructions — at the moment some what rudimentary.
<code>\socket_log:n</code>	<code>\LogSocket</code> and <code>\socket_log:n</code> only differ in that they don't stop.

It is sometimes necessary/helpful to know if a particular socket or plug exists (or is assigned to a certain socket) and based on that take different actions.

<hr/>	<hr/>
<code>\IfSocketExistsTF</code>	<code>* \IfSocketExistsTF {<socket-name>} {<true code>} {<false code>}</code>
<code>\socket_if_exist:nTF</code>	<code>*</code> If socket <code><socket-name></code> exists then execute <code><true code></code> otherwise <code><false code></code> . Variants with only T or F are also available.

<hr/>	<hr/>
<code>\IfSocketPlugExistsTF</code>	<code>* \IfSocketPlugExistsTF {<socket-name>} {<plug-name>}</code>
<code>\socket_if_plug_exist:nnTF</code>	<code>* {<true code>} {<false code>}</code>
	If plug <code><plug-name></code> for socket <code><socket-name></code> exists then execute <code><true code></code> otherwise <code><false code></code> . Variants with only T or F are also available.

<hr/>	<hr/>
<code>\IfSocketPlugAssignedTF</code>	<code>* \IfSocketPlugAssignedTF {<socket-name>} {<plug-name>}</code>
<code>\socket_if_plug_assigned:nnTF</code>	<code>* {<true code>} {<false code>}</code>
	If plug <code><plug-name></code> is assigned to socket <code><socket-name></code> then execute <code><true code></code> otherwise <code><false code></code> . Variants with only T or F are also available.

<hr/> <code>\socket_get_plug:nN</code> <hr/>	<code>\socket_get_plug:nN {<socket-name>} {<tl var>}</code>
	This stores into <code><tl var></code> the name of the plug currently assigned to the socket <code><socket-name></code> . If the socket is not declared an error is issued.
<hr/> <code>\DebugSocketsOn</code> <code>\DebugSocketsOff</code> <code>\socket_debug_on:</code> <code>\socket_debug_off:</code> <hr/>	<code>\DebugSocketsOn ... \DebugSocketsOff</code> Turns debugging of sockets on or off.

2.4.2 Rationale for error handling

The errors during the declarations are produced to help with typos—after all, such declarations might be part of a document preamble (not that likely, but possible). However, `\UseSocket` is not doing much checking, e.g.,

```
\UseSocket{mispelled-socket}{hello}{world}
```

will generate a rather low-level error and then typesets “helloworld” because there is no dedicated runtime check if `mispelled-socket` is a known socket.

The reason is that if the misspelling is in the code, then this is a programming error in the package and for speed reasons L^AT_EX does not repeatedly make runtime checks for coding errors unless they can or are likely to be user introduced.

3 The Implementation

The implementation of the socket mechanism should be (partially) redone and we should probably store the different code chunks in a property list so that we can have a decent `\ShowSocket` command that shows the available alternatives.

TODO: *implement?*

```

1 <*2ekernel | latexrelease>
2 \ExplSyntaxOn
3 <@@=socket>
4 <[latexrelease] \NewModuleRelease{2023/11/01}{ltsockets}>
5 <[latexrelease]                               {The~socket~management~system}>

```

3.1 Debugging the socket structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

```

\g__socket_debug_bool Holds the current debugging state.
    6 \bool_new:N \g__socket_debug_bool
(End of definition for \g__socket_debug_bool.)

```

\socket_debug_on: Turns debugging on and off by redefining `\__socket_debug:n` and `\__socket_debug_term:n`. By default they do nothing.

\socket_debug_off:

```

\__socket_debug:n      7 \cs_new_eq:NN \__socket_debug:n \use_none:n
\__socket_debug_term:n 8 \cs_new_eq:NN \__socket_debug_term:n \use_none:n
\__socket_debug_gset:
9 \cs_new_protected:Npn \socket_debug_on:
10 {
11   \bool_gset_true:N \g__socket_debug_bool
12   \__socket_debug_gset:
13 }
14 \cs_new_protected:Npn \socket_debug_off:
15 {
16   \bool_gset_false:N \g__socket_debug_bool
17   \__socket_debug_gset:
18 }
19 \cs_new_protected:Npn \__socket_debug_gset:
20 {
21   \cs_gset_protected:Npe \__socket_debug:n ##1
22   { \bool_if:NT \g__socket_debug_bool {##1} }
23   \cs_gset_protected:Npe \__socket_debug_term:n ##1
24   { \bool_if:NT \g__socket_debug_bool
25
26     { \iow_term:e { ^^J [Sockets]~ ==>~ ##1 \exp_not:N\on@line} } }

```

(End of definition for `\socket_debug_on:` and others. These functions are documented on page 361.)

3.2 The L3 layer commands

\socket_new:nn Declaring a socket creates a str to hold the name (a pointer) to the code that should be used when the socket is executed, and an integer to hold the number of inputs of that socket. Initially, an “empty” code chunk is created and assigned so the socket does nothing by default other than swallowing its inputs (if any).

```

27 \cs_new_protected:Npn \socket_new:nn #1 #2 {
28   \socket_if_exist:nTF {#1}
29   {
30     \msg_error:nnn { socket } { already-declared } {#1}
31   }
32   {

```

We only support declarations on top-level.

```

33   \int_if_zero:nTF \tex_currentgrouplevel:D
34   {
35     \str_new:c { l__socket_#1_plug_str }
36     \seq_new:c { l__socket_#1_plugs_seq }
37     \int_const:cn { c__socket_#1_args_int } {#2}
38     \socket_new_plug:nnn {#1} { noop } {}
39     \int_compare:nNnTF {#2} = 1
40     {
41       \socket_new_plug:nnn {#1} { identity } {##1}
42       \socket_assign_plug:nn {#1} { identity }
43     }
44     { \socket_assign_plug:nn {#1} { noop } }
45     \__socket_debug_term:n

```

```

46         { Socket~ '#1'~ declared~ with~ #2~ input(s) }
47     }
48     {
49         \msg_error:nn { socket } { not-top-level }
50     }
51 }
52 }

```

(End of definition for `\socket_new:nn`. This function is documented on page 359.)

`\socket_if_exist_p:n` Conditional testing the existence of a socket. The argument is fully expanded as part of the csname generation.

`\socket_if_exist:nTF`

```

53 \prg_new_conditional:Npnn \socket_if_exist:n #1 { p , T , F , TF }
54 { \str_if_exist:cTF { l__socket_#1_plug_str }
55   \prg_return_true:
56   \prg_return_false:
57 }

```

(End of definition for `\socket_if_exist:nTF`. This function is documented on page 360.)

`\socket_log:n` Show the current state of the socket — for now this is just a quick draft and should be redone and extended.

`\socket_show:n`

```

58 \cs_new_protected:Npn \socket_log:n #1 {
59   \typeout{ Socket~ #1:}
60   \socket_if_exist:nTF {#1}
61   {
62     \typeout{ @spaces number~ of~ inputs~ =~
63               \int_use:c { c__socket_#1_args_int } }
64     \typeout{ @spaces available~plugs~ =~
65               \seq_use:cn { l__socket_#1_plugs_seq }{,~}{,~}{,~} }
66     \typeout{ @spaces current~ plug~ =~
67               \str_use:c { l__socket_#1_plug_str } }
68     \typeout{ @spaces definition~ =~
69               \cs_meaning:c
70               { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w } }
71     \typeout{}
72   }
73   {

```

If we are showing a socket it is not an error if it doesn't exist.

```

74     \typeout { Socket~ is~ not~ declared! }
75   }
76 }

```

And here the version that stops:

```

77 \cs_new_protected:Npn \socket_show:n #1 {\socket_log:n {#1} \errmessage{}}

```

(End of definition for `\socket_log:n` and `\socket_show:n`. These functions are documented on page 360.)

`\socket_new_plug:nnn` Declaring a code for a socket is just making a definition, taking the number of arguments from the saved int.

`\socket_set_plug:nnn`

```

78 \cs_new_protected:Npn \socket_new_plug:nnn #1#2#3 {
79   \socket_if_exist:nTF {#1}
80   {

```

```

81     \socket_if_plug_exist:nnTF {#1} {#2}
82     {
83         \msg_error:nnnn { socket } { plug-already-declared } {#1} {#2}
84     }
85     {
86         \cs_generate_from_arg_count:cNnn
87         { __socket_#1_plug_#2:w }
88         \cs_new:Npn
89         { \int_use:c { c__socket_#1_args_int } }
90         {#3}

```

This is a new declaration so we add the name to a seq for the debugging info.

```

91         \seq_put_right:cn { l__socket_#1_plugs_seq } {#2}
92         \__socket_debug_term:n
93         { Plug~ '2'~ for~ socket~ '1'~ declared }
94     }
95 }
96 { \msg_error:nnn { socket } { undeclared } {#1} }
97 }

```

Changing the plug of an existing socket is rather similar, except that we don't have to deal with adding it to the debugging sequence.

```

98 \cs_new_protected:Npn \socket_set_plug:nnn #1#2#3 {
99     \socket_if_exist:nTF {#1}
100     {
101         \socket_if_plug_exist:nnTF {#1} {#2}
102         {
103             \cs_generate_from_arg_count:cNnn
104             { __socket_#1_plug_#2:w }
105             \cs_set:Npn
106             { \int_use:c { c__socket_#1_args_int } }
107             {#3}
108             \__socket_debug_term:n
109             { Plug~ '2'~ for~ socket~ '1'~ changed }
110         }
111         {
112             \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
113         }
114     }
115     { \msg_error:nnn { socket } { undeclared } {#1} }
116 }

```

(End of definition for \socket_new_plug:nnn and \socket_set_plug:nnn. These functions are documented on page 359.)

`\socket_if_plug_exist_p:nn` Conditional testing the existence of a plug. Both arguments are fully expanded as part of the csname generation.

`\socket_if_plug_exist:nnTF`

```

117 \prg_new_conditional:Npnn \socket_if_plug_exist:nn #1#2 { p , T , F , TF }
118 { \cs_if_exist:cTF { __socket_#1_plug_#2:w }
119     \prg_return_true:
120     \prg_return_false:
121 }

```

(End of definition for \socket_if_plug_exist:nnTF. This function is documented on page 360.)

\socket_assign_plug:nn Assigning a plug to a socket just changes the name in the socket string. The assignment is local to the current group.

```

122 \cs_new_protected:Npn \socket_assign_plug:nn #1 #2 {
123   \socket_if_exist:nTF {#1}
124   {
125     \socket_if_plug_exist:nnTF {#1} {#2}
126     {
127       \__socket_debug_term:n
128       { Replacing~ plug~ '\str_use:c { l__socket_#1_plug_str }'~
129         with~ '#2'~ in~ socket~ '#1' }
130       \str_set:cn { l__socket_#1_plug_str } {#2}
131     }
132     {
133       \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
134     }
135   }
136   { \msg_error:nnn { socket } { undeclared } {#1} }
137 }

```

(End of definition for \socket_assign_plug:nn. This function is documented on page 359.)

\socket_if_plug_assigned:nnTF Conditional testing the assignment of a plug. Both arguments are fully expanded.

```

138 \prg_new_conditional:Npnn \socket_if_plug_assigned:nn #1#2 { p , T , F , TF }
139 { \exp_args:Ne
140   \str_if_eq:nvTF {#2} { l__socket_#1_plug_str }
141   \prg_return_true:
142   \prg_return_false:
143 }

```

(End of definition for \socket_if_plug_assigned:nnTF. This function is documented on page 360.)

\socket_get_plug:nN Retrieving the currently assigned plug.

```

144 \cs_new_protected:Npn \socket_get_plug:nN #1#2
145 {
146   \socket_if_exist:nTF {#1}
147   {
148     \tl_set_eq:Nc #2 { l__socket_#1_plug_str }
149   }
150   {
151     \msg_error:nnn { socket } { undeclared } {#1}
152   }
153 }

```

(End of definition for \socket_get_plug:nN. This function is documented on page 361.)

\socket_use:nw And using it is more or less a \use:c so very lightweight. We do not add a runtime check for speed reasons!
\socket_use:n This command is named \socket_use:nw because we don't know how many inputs the socket has until we have looked at the socket name (in argument #1). But, of course,
\socket_use:nn the developer knows so we also offer a few aliases \socket_use:nn, etc. so that one can
\socket_use:nnnn indicate the correct number of arguments (socket inputs plus one) in the L3 layer code.

```

154 \cs_new_protected:Npn \socket_use:nw #1 {
155   \__socket_debug_term:n
156   { Socket~ '#1'~ containing~ plug~

```



```

157     '\str_use:c { l__socket_#1_plug_str }'~ used }
158     \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
159 }

```

To make code a bit more readable we also define functions that indicate how many arguments are picked up. However, this is just for code documentation: internally they all do the same and the number of arguments isn't checked by default.

```

160 \cs_new_eq:NN \socket_use:n \socket_use:nw % socket with no inputs
161 \cs_new_eq:NN \socket_use:nn \socket_use:nw % socket with one input
162 \cs_new_eq:NN \socket_use:nnn \socket_use:nw % socket with two inputs
163 \cs_new_eq:NN \socket_use:nnnn \socket_use:nw % socket with three inputs

```

The above commands could be changed to check how many inputs the socket is declared with (for example, when checking is in force).

TODO: *Implement?*

(End of definition for `\socket_use:nw` and others. These functions are documented on page 360.)

`\socket_use_expandable:nw`
`\socket_use_expandable:n`

The same as the non-expandable code, except for the missing debug output.

```

164 \cs_new:Npn \socket_use_expandable:nw #1 {
165     \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
166 }
167 \cs_new_eq:NN \socket_use_expandable:n \socket_use_expandable:nw % socket with no inputs

```

(End of definition for `\socket_use_expandable:nw` and `\socket_use_expandable:n`. These functions are documented on page 360.)

3.3 Error messages

```

168 \msg_new:nnnn { socket } { already-declared }
169 { Socket~ '#1'~ already~ declared! }
170 { A~ socket~ can~ only~ be~ declared~ once.~ The~ name~ '#1'~ is~
171     already~ taken.~ Use~ \ShowSocket{#1}~ to~ see~ its~ definition. }
172
173 \msg_new:nnnn { socket } { undeclared }
174 { Socket~ '#1'~ undeclared! }
175 { You~ tried~ to~ use~ a~ socket~ that~ was~ not~ declared~ before. }
176
177 \msg_new:nnnn { socket } { not-top-level }
178 { Sockets~ can~ only~ be~ declared~ at~ top-level! }
179 { It~ is~ not~ allowed~ to~ declare~ sockets~ inside~ a~
180     group.~ Move~ the~ declaration~ to~ the~ top-level. }
181
182 \msg_new:nnnn { socket } { plug-already-declared }
183 { Plug~ '#2'~ for~ socket~ '#1'~ already~ declared! }
184 { You~ can't~ change~ an~ existing~ plug~ with~ \NewSocketPlug~ and~ it~
185     is~ normally~ not~ sensible~ to~ do~ so.~ Use~ the~ L3~ programming~
186     layer~ function~ \socket_set_plug:nnn~ if~ you~ really~ have~ to. }
187
188 \msg_new:nnnn { socket } { plug-undeclared }
189 { Plug~ '#2'~ for~ socket~ '#1'~ undeclared! }
190 { The~ plug~ name~ is~ unknown.~ Is~ the~ name~ misspelled~ or~ did~ you~
191     intend~ to~ assign~ it~ to~ a~ different~ socket? }
192
193 \prop_gput:Nnn \g_msg_module_type_prop { socket } { LaTeX }

```

3.4 The L^AT_EX 2_ε interface commands

`\NewSocket` `\NewSocketPlug` `\ShowSocket` `\LogSocket` `\AssignSocketPlug` `\UseSocket` `\DebugSocketsOn` `\DebugSocketsOff`

As we expect that there are existing L^AT_EX 2_ε packages that may want to make use of the socket mechanism, we provide 2e names for most of the commands.

```

192 \cs_new_eq:NN \NewSocket      \socket_new:nn
193 \cs_new_eq:NN \ShowSocket    \socket_show:n
194 \cs_new_eq:NN \LogSocket     \socket_log:n
195 \cs_new_eq:NN \NewSocketPlug \socket_new_plug:nnn
196 \cs_new_eq:NN \AssignSocketPlug \socket_assign_plug:nn
197 \cs_new_eq:NN \UseSocket     \socket_use:nw
198 \cs_new_eq:NN \DebugSocketsOn \socket_debug_on:
199 \cs_new_eq:NN \DebugSocketsOff \socket_debug_off:

```

(End of definition for `\NewSocket` and others. These functions are documented on page 359.)

```

\IfSocketExistsTF
\IfSocketExistsT
\IfSocketExistsF
\IfSocketPlugExistsTF
\IfSocketPlugExistsT
\IfSocketPlugExistsF
\IfSocketPlugAssignedTF
\IfSocketPlugAssignedT
\IfSocketPlugAssignedF

```

A bunch of conditionals:

```

200 \cs_new_eq:NN \IfSocketExistsTF \socket_if_exist:nTF
201 \cs_new_eq:NN \IfSocketExistsT  \socket_if_exist:nT
202 \cs_new_eq:NN \IfSocketExistsF  \socket_if_exist:nF
203 \cs_new_eq:NN \IfSocketPlugExistsTF \socket_if_plug_exist:nnTF
204 \cs_new_eq:NN \IfSocketPlugExistsT  \socket_if_plug_exist:nnT
205 \cs_new_eq:NN \IfSocketPlugExistsF  \socket_if_plug_exist:nnF
206 \cs_new_eq:NN \IfSocketPlugAssignedTF \socket_if_plug_assigned:nnTF
207 \cs_new_eq:NN \IfSocketPlugAssignedT  \socket_if_plug_assigned:nnT
208 \cs_new_eq:NN \IfSocketPlugAssignedF  \socket_if_plug_assigned:nnF

```

(End of definition for `\IfSocketExistsTF` and others. These functions are documented on page 360.)

```

209 %
210 <latexrelease>\IncludeInRelease{0000/00/00}{ltsockets}
211 <latexrelease>          {The~socket~management~(undo)}%
212 <latexrelease>
213 <latexrelease>\let \NewSocket \@undefined
214 <latexrelease>\let \ShowSocket \@undefined
215 <latexrelease>\let \LogSocket \@undefined
216 <latexrelease>
217 <latexrelease>\let \NewSocketPlug \@undefined
218 <latexrelease>\let \AssignSocketPlug \@undefined
219 <latexrelease>\let \UseSocket \@undefined
220 <latexrelease>
221 <latexrelease>\let \DebugSocketsOn \@undefined
222 <latexrelease>\let \DebugSocketsOff \@undefined
223 <latexrelease>
224 <latexrelease>\let \IfSocketExistsTF \@undefined
225 <latexrelease>\let \IfSocketExistsT \@undefined
226 <latexrelease>\let \IfSocketExistsF \@undefined
227 <latexrelease>\let \IfSocketPlugExistsTF \@undefined
228 <latexrelease>\let \IfSocketPlugExistsT \@undefined
229 <latexrelease>\let \IfSocketPlugExistsF \@undefined
230 <latexrelease>\let \IfSocketPlugAssignedTF \@undefined
231 <latexrelease>\let \IfSocketPlugAssignedT \@undefined
232 <latexrelease>\let \IfSocketPlugAssignedF \@undefined
233 <latexrelease>
234 <latexrelease>\EndModuleRelease

```

```
235 \ExplSyntaxOff
236 </2ekernel | latexrelease>
      Reset module prefix:
237 <@@@=>
```

File 11

ltemplates.dtx

1 Introduction

There are three broad “layers” between putting down ideas into a source file and ending up with a typeset document. These layers of document writing are

1. authoring of the text with mark-up;
2. document layout design;
3. implementation (with T_EX programming) of the design.

We write the text as an author, and we see the visual output of the design after the document is generated; the T_EX implementation in the middle is the glue between the two.

L^AT_EX’s greatest success has been to standardize a system of mark-up that balances the trade-off between ease of reading and ease of writing to suit almost all forms of technical writing. It’s other original strength was a good background in typographical design; while the standard L^AT_EX 2_ε classes look somewhat dated now in terms of their visual design, their typography is generally sound (barring the occasional minor faults).

However, L^AT_EX 2_ε has always lacked a standard approach to customising the visual design of a document. Changing the looks of the standard classes involved either:

- Creating a new version of the implementation code of the class and editing it.
- Loading one of the many packages to customise certain elements of the standard classes.
- Loading a completely different document class, such as KOMA-Script or memoir, that allows easy customization.

All three of these approaches have their drawbacks and learning curves.

The idea behind ltemplates is to cleanly separate the three layers introduced at the beginning of this section, so that document authors who are not programmers can easily change the design of their documents. ltemplates also makes it easier for L^AT_EX programmers to provide their own customizations on top of a pre-existing class.

2 What is a document?

Besides the textual content of the words themselves, the source file of a document contains mark-up elements that add structure to the document. These elements include sectional divisions, figure/table captions, lists of various sorts, theorems/proofs, and so on. The list will be different for every document that can be written.

Each element can be represented logically without worrying about the formatting, with mark-up such as `\section`, `\caption`, `\begin{enumerate}` and so on. The output of each one of these document elements will be a typeset representation of the information marked up, and the visual arrangement and design of these elements can vary widely in producing a variety of desired outcomes.

For each type of document element, there may be design variations that contain the same sort of information but present it in slightly different ways. For example, the difference between a numbered and an unnumbered section, `\section` and `\section*`, or the difference between an itemized list or an enumerated list.

There are three distinct layers in the definition of “a document” at this level

1. semantic elements such as the ideas of sections and lists;
2. a set of design solutions for representing these elements visually;
3. specific variations for these designs that represent the elements in the document.

In the parlance of the template system, these are called types, templates, and instances, and they are discussed below in sections 4, 5, and 7, respectively.

3 Types, templates, and instances

By formally declaring documents to be composed of mark-up elements grouped into types, which are interpreted and typeset with a set of templates, each of which has one or more instances with which to compose each and every semantic unit of the text, we can cleanly separate the components of document construction.

All of the structures provided by the template system are global, and do not respect $\TeX$ grouping.

4 Template types

An *template type* (sometimes just “type”) is an abstract idea of a document element that takes a fixed number of arguments corresponding to the information from the document author that it is representing. A sectioning type, for example, might take three inputs: “title”, “short title”, and “label”.

Any given document class will define which types are to be used in the document, and any template of a given type can be used to generate an instance for the type. (Of course, different templates will produce different typeset representations, but the underlying content will be the same.)

```
\NewTemplateType <{template type}> <{no. of args}>
```

This function defines an `<template type>` taking `<number of arguments>`, where the `<type>` is an abstraction as discussed above. For example,

```
\NewTemplateType{sectioning}{3}
```

creates a type “sectioning”, where each use of that type will need three arguments.

5 Templates

A *template* is a generalized design solution for representing the information of a specified type. Templates that do the same thing, but in different ways, are grouped together by their type and given separate names. There are two important parts to a template:

- the parameters it takes to vary the design it is producing;

Key-type	Description of input
<code>boolean</code>	<code>true</code> or <code>false</code>
<code>choice{⟨choices⟩}</code>	A list of pre-defined <code>⟨choices⟩</code>
<code>commalist</code>	A comma-separated list
<code>function{⟨N⟩}</code>	A (protected) function definition with N arguments (N from 0 to 9)
<code>instance{⟨name⟩}</code>	An instance of type <code>⟨name⟩</code>
<code>integer</code>	An integer or integer expression
<code>length</code>	A fixed length
<code>muskip</code>	A math length with shrink and stretch components
<code>real</code>	A real (floating point) value
<code>skip</code>	A length with shrink and stretch components
<code>tokenlist</code>	A token list: any text or commands

Table 1: Key-types for defining template interfaces with `\DeclareTemplateInterface`.

- the implementation of the design.

As a document author or designer does not care about the implementation but rather only the interface to the template, these two aspects of the template definition are split into two independent declarations, `\DeclareTemplateInterface` and `\DeclareTemplateCode`.

<code>\DeclareTemplateInterface</code>	<code>\DeclareTemplateInterface</code> <code>{⟨type⟩}{⟨template⟩}{⟨no. of args⟩}</code> <code>{⟨key list⟩}</code>
--	---

A `⟨template⟩` interface is declared for a particular `⟨type⟩`, where the `⟨number of arguments⟩` must agree with the type declaration. The interface itself is defined by the `⟨key list⟩`, which is itself a key–value list taking a specialized format:

```

⟨key1⟩ ":" ⟨key type1⟩ ","
⟨key2⟩ ":" ⟨key type2⟩ ","
⟨key3⟩ ":" ⟨key type3⟩ "=" ⟨default3⟩ ","
⟨key4⟩ ":" ⟨key type4⟩ "=" ⟨default4⟩ ","
...

```

Each `⟨key⟩` name should consist of ASCII characters, with the exception of `,`, `=` and `:`. The recommended form for key names is to use lower case letters, with dashes to separate out different parts. Spaces are ignored in key names, so they can be included or missed out at will. Each `⟨key⟩` must have a `⟨key type⟩`, which defines the type of input that the `⟨key⟩` requires. A full list of key types is given in Table 1. Each key may have a `⟨default⟩` value, which will be used in by the template if the `⟨key⟩` is not set explicitly. The `⟨default⟩` should be of the correct form to be accepted by the `⟨key type⟩` of the `⟨key⟩`: this is not checked by the code. Expressions for numerical values are evaluated when the template is used, thus for example values given in terms of `em` or `ex` will be set respecting the prevailing font.

<code>\KeyValue</code>	<code>\KeyValue {⟨key name⟩}</code>
------------------------	-------------------------------------

There are occasions where the default (or value) for one key should be taken from another. The `\KeyValue` function can be used to transfer this information without needing to know the internal implementation of the key:

```
\DeclareTemplateInterface { type } { template } { no. of args }
{
  key-name-1 : key-type = value ,
  key-name-2 : key-type = \KeyValue { key-name-1 },
  ...
}
```

<code>\DeclareTemplateCode</code>	<code>\DeclareTemplateCode</code>
-----------------------------------	-----------------------------------

```
{⟨type⟩} {⟨template⟩} {⟨no. of args⟩}
{⟨key bindings⟩} {⟨code⟩}
```

The relationship between a templates keys and the internal implementation is created using the `\DeclareTemplateCode` function. As with `\DeclareTemplateInterface`, the `⟨template⟩` name is given along with the `⟨type⟩` and `⟨number of arguments⟩` required. The `⟨key bindings⟩` argument is a key-value list which specifies the relationship between each `⟨key⟩` of the template interface with an underlying `⟨variable⟩`.

```
⟨key1⟩ "=" ⟨variable1⟩,
⟨key2⟩ "=" ⟨variable2⟩,
⟨key3⟩ "=" global ⟨variable3⟩,
⟨key4⟩ "=" global ⟨variable4⟩,
⟨key5⟩ "=" name { ⟨name5⟩ },
...
```

As illustrated, the keyword `global` may be included in the listing to indicate that the `⟨variable⟩` should be assigned globally. The keyword `name` may also be used to give the variable by (cs)name rather than as a single token. If both keywords are used, `global` should come before `name`.

A full list of variable bindings is given in Table 2.

The `⟨code⟩` argument of `\DeclareTemplateCode` is used as the replacement text for the template when it is used, either directly or as an instance. This may therefore accept arguments `#1`, `#2`, etc. as detailed by the `⟨number of arguments⟩` taken by the type.

<code>\AssignTemplateKeys</code>	<code>\AssignTemplateKeys</code>
----------------------------------	----------------------------------

In the final argument of `\DeclareTemplateCode` the assignment of keys defined by the template may be delayed by including the command `\AssignTemplateKeys`. If this is *not* present, keys are assigned immediately before the template code. If an `\AssignTemplateKeys` command is present, assignment is delayed until this point. Note that the command must be *directly* present in the code, not placed within a nested command/macro.

Key-type	Description of binding
<code>boolean</code>	Boolean variable, <i>e.g.</i> <code>\l_tmpa_bool</code>
<code>choice</code>	List of choice implementations (see Section 6)
<code>commalist</code>	Comma list, <i>e.g.</i> <code>\l_tmpa_clist</code>
<code>function</code>	Function taking N arguments, <i>e.g.</i> <code>\use_i:nn</code>
<code>instance</code>	
<code>integer</code>	Integer variable, <i>e.g.</i> <code>\l_tmpa_int</code>
<code>length</code>	Dimension variable, <i>e.g.</i> <code>\l_tmpa_dim</code>
<code>muskip</code>	Muskip variable, <i>e.g.</i> <code>\l_tmpa_muskip</code>
<code>real</code>	Floating-point variable, <i>e.g.</i> <code>\l_tmpa_fp</code>
<code>skip</code>	Skip variable, <i>e.g.</i> <code>\l_tmpa_skip</code>
<code>tokenlist</code>	Token list variable, <i>e.g.</i> <code>\l_tmpa_tl</code>

Table 2: Bindings required for different key types when defining template implementations with `\DeclareTemplateCode`. Apart from `choice` and `function` all of these accept the keyword `global` to carry out a global assignment. Apart from `choice`, all of these accept the keyword `name` to allow passing command name without the leading backslash: this is useful for auto-generated names.

<code>\SetKnownTemplateKeys</code>	<code>\SetKnownTemplateKeys</code>	<code>{\type}\{template}\{keyvals}</code>
<code>\SetTemplateKeys</code>	<code>\SetTemplateKeys</code>	<code>{\type}\{template}\{keyvals}</code>
<code>\SetCurrentKnownTemplateKeys</code>	<code>\SetCurrentKnownTemplateKeys</code>	<code>\{keyvals}</code>
<code>\UnusedTemplateKeys</code>	<code>\UnusedTemplateKeys</code>	<code>% all {keyvals} unused by previous \SetKnownTemplateKeys</code>

In the final argument of `\DeclareTemplateCode` one can also overwrite (some of) the current template key value settings by using the command `\SetKnownTemplateKeys` or `\SetTemplateKeys`, i.e., they can overwrite the template default values and the values assigned by the instance.

The `\SetKnownTemplateKeys`, `\SetTemplateKeys` and `\SetCurrentKnownTemplateKeys` commands are only supported within the code of a template; using them elsewhere has unpredictable results. If they are used together with `\AssignTemplateKeys` then the latter command should come first in the template code.

The main use case for these commands is the situation where there is an argument (normally #1) to the template in which a key/value list can be specified that overwrites the normal settings. In that case one could use

`\SetKnownTemplateKeys{\type}\{template}\{#1}`

to process this key/value list inside the template.

If `\SetKnownTemplateKeys` or `\SetCurrentKnownTemplateKeys` are executed and the `\{keyvals}` argument contains keys not known to the `\{template}` they are simply ignored and stored in the tokenlist `\UnusedTemplateKeys` without generating an error. This way it is possible to apply the same key/val list specified by the user on a document-level command or environment to several templates, which is useful, if the command or environment is implemented by calling several different template instances.

`\SetCurrentKnownTemplateKeys` works like `\SetKnownTemplateKeys` but obtains the name of the current `\{type}\{template}` automatically based on the current context. It should be used before any other template code may be called (and thus change the value of the tracking of the current template).

As a variation of that, you can use this key/val list the first time, and for the next template instance use what remains in `\UnusedTemplateKeys` (i.e., the key/val list with only the keys that have not been processed previously). The final processing step could then be `\SetTemplateKeys`, which unconditionally attempts to set the `<keyvals>` received in its third argument. This command complains if any of them are unknown keys. Alternatively, you could use `\SetKnownTemplateKeys` and afterwards check whether `\UnusedTemplateKeys` is empty.²³

For example, a list, such as `enumerate`, is made up from a `blockenv`, `block`, `list`, and a `para` template and in the single user-supplied optional argument of `enumerate` key/values for any of these templates might be specified.

In fact, in the particular example of list environments, the supplied key/value list is also saved and then applied to each `\item` which is implemented through an `item` template. This way, one can specify one-off settings for all the items of a single list (on the environment level), as well as to individual items within that list (by specifying them in the optional argument of an `\item`). With `\SetKnownTemplateKeys` and `\SetTemplateKeys` working together, it is possible to provide this flexibility and still alert the user when one of their keys is misspelled.

On the other hand you may want to allow for “misspellings” without generating an error or a warning. For example, if you define a template that accepts only a few keys, you might just want to ignore anything specified in the source when you use this template in place of a different one, without the need to alter the document source. Or you might just generate a warning message, which is easy, given that the unused key/values are available in the `\UnusedTemplateKeys` variable.

`\DeclareTemplateCopy`

`\DeclareTemplateCopy`
`{<type>} {<template2>} {<template1>}`

Copies `<template1>` of `<type>` to a new name `<template2>`: the copy can then be edited independent of the original.

6 Multiple choices

The `choice` key type implements multiple choice input. At the interface level, only the list of valid choices is needed:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
{ key-name : choice { A, B, C } }
```

where the choices are given as a comma-list (which must therefore be wrapped in braces). A default value can also be given:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
{ key-name : choice { A, B, C } = A }
```

At the implementation level, each choice is associated with code, using a nested key-value list.

²³Using `\SetTemplateKeys` exposes the inner structure of the template keys when generating an error. This is something one may want to avoid as it can be confusing to the user, especially if several templates are involved. In that case use `\SetKnownTemplateKeys` and afterwards check whether `\UnusedTemplateKeys` is empty; if it is not empty then generate your own error message.

```

\DeclareTemplateCode { foo } { bar } { 0 }
{
  key-name =
  {
    A = Code-A ,
    B = Code-B ,
    C = Code-C
  }
}
{ ... }

```

The two choice lists should match, but in the implementation a special **unknown** choice is also available. This can be used to ignore values and implement an “else” branch:

```

\DeclareTemplateCode { foo } { bar } { 0 }
{
  key-name =
  {
    A      = Code-A ,
    B      = Code-B ,
    C      = Code-C ,
    unknown = Else-code
  }
}
{ ... }

```

The **unknown** entry must be the last one given, and should *not* be listed in the interface part of the template.

For keys which accept the values **true** and **false** both the boolean and choice key types can be used. As template interfaces are intended to prompt clarity at the design level, the boolean key type should be favored, with the choice type reserved for keys which take arbitrary values.

7 Instances

After a template is defined it still needs to be put to use. The parameters that it expects need to be defined before it can be used in a document. Every time a template has parameters given to it, an *instance* is created, and this is the code that ends up in the document to perform the typesetting of whatever pieces of information are input into it.

For example, a template might say “here is a section with or without a number that might be centered or left aligned and print its contents in a certain font of a certain size, with a bit of a gap before and after it” whereas an instance declares “this is a section with a number, which is centered and set in 12pt italic with a 10pt skip before and a 12pt skip after it”. Therefore, an instance is just a frozen version of a template with specific settings as chosen by the designer.

<code>\DeclareInstance</code>	<code>\DeclareInstance</code>
-------------------------------	-------------------------------

<code>{\type}</code>	<code>{\instance}</code>	<code>{\template}</code>	<code>{\parameters}</code>
----------------------	--------------------------	--------------------------	----------------------------

This function uses a `<template>` for an `<type>` to create an `<instance>`. The `<instance>` will be set up using the `<parameters>`, which will set some of the `<keys>` in the `<template>`.

As a practical example, consider a type for document sections (which might include chapters, parts, sections, *etc.*), which is called **sectioning**. One possible template for this type might be called **basic**, and one instance of this template would be a numbered section. The instance declaration might read:

```
\DeclareInstance { sectioning } { section-num } { basic }
{
  numbered      = true ,
  justification = center ,
  font          =\normalsize\itshape ,
  before-skip   = 10pt ,
  after-skip    = 12pt ,
}
```

Of course, the key names here are entirely imaginary, but illustrate the general idea of fixing some settings.

<code>\InstanceValue</code> *	<code>\InstanceValue</code>	<code>{\type}</code>	<code>{\instance}</code>	<code>{\key}</code>
-------------------------------	-----------------------------	----------------------	--------------------------	---------------------

Expands to the current value for the `<key>` stored in the `<instance>` of `<type>`. If the `<instance>` or `<key>` do not exist, the expansion is empty. The result is returned within the `\unexpanded` primitive (`\exp_not:n`).

<code>\IfInstanceExistsT</code>	<code>\IfInstanceExistsTF</code>	<code>{\type}</code>	<code>{\instance}</code>	<code>{\true code}</code>	<code>{\false code}</code>
---------------------------------	----------------------------------	----------------------	--------------------------	---------------------------	----------------------------

<code>\IfInstanceExistsF</code>	<code>\IfInstanceExistsTF</code>	Tests if the named <code><instance></code> of a <code><type></code> exists, and then inserts the appropriate code into the input stream.
---------------------------------	----------------------------------	--

<code>\DeclareInstanceCopy</code>	<code>\DeclareInstanceCopy</code>
-----------------------------------	-----------------------------------

<code>{\type}</code>	<code>{\instance2}</code>	<code>{\instance1}</code>
----------------------	---------------------------	---------------------------

Copies the `<values>` for `<instance1>` for an `<type>` to `<instance2>`.

8 Document interface

After the instances have been chosen, document commands must be declared to use those instances in the document. `\UseInstance` calls instances directly, and this command should be used internally in document-level mark-up.

<code>\UseInstance</code>	<code>\UseInstance</code>
---------------------------	---------------------------

<code>{\type}</code>	<code>{\instance}</code>	<code><arguments></code>
----------------------	--------------------------	--------------------------------

Uses an `<instance>` of the `<type>`, which will require `<arguments>` as determined by the number specified for the `<type>`. The `<instance>` must have been declared before it can be used, otherwise an error is raised.

<code>\UseTemplate</code>	<code>\UseTemplate {<type>} {<template>}</code> <code>{<settings>} <arguments></code>
---------------------------	--

Uses the `<template>` of the specified `<type>`, applying the `<settings>` and absorbing `<arguments>` as detailed by the `<type>` declaration. This in effect is the same as creating an instance using `\DeclareInstance` and immediately using it with `\UseInstance`, but without the instance having any further existence. This command is therefore useful when a template needs to be used only once.

This function can also be used as the argument to `instance` key types:

```
\DeclareInstance { type } { template } { instance }
{
  instance-key =
    \UseTemplate { type2 } { template2 } { <settings> }
}
```

9 Changing existing definitions

Template parameters may be assigned specific defaults for instances to use if the instance declaration doesn't explicit set those parameters. In some cases, the document designer will wish to edit these defaults to allow them to “cascade” to the instances. The alternative would be to set each parameter identically for each instance declaration, a tedious and error-prone process.

<code>\EditTemplateDefaults</code>	<code>\EditTemplateDefaults</code> <code>{<type>} {<template>} {<new defaults>}</code>
------------------------------------	---

Edits the `<defaults>` for a `<template>` for an `<type>`. The `<new defaults>`, given as a key–value list, replace the existing defaults for the `<template>`. This means that the change will apply to instances declared after the editing, but that instances which have already been created are unaffected.

<code>\EditInstance</code>	<code>\EditInstance</code> <code>{<type>} {<instance>} {<new values>}</code>
----------------------------	---

Edits the `<values>` for an `<instance>` for an `<type>`. The `<new values>`, given as a key–value list, replace the existing values for the `<instance>`. This function is complementary to `\EditTemplateDefaults`: `\EditInstance` changes a single instance while leaving the template untouched.

9.1 Expanding the values of keys

To allow the user to apply expansion of values when the key is set, key names can be followed by an expansion specifier. This is given by appending `:` and a single letter specifier to the key name. These letters are the normal argument specifiers for `expl3`, thus they may be one of `n` (redundant but supported), `o`, `V`, `v`, `e`, `N` (again redundant) or `c`. Expansion of a control sequence name is particularly useful when you need to refer to an internal $\text{\LaTeX} 2_{\epsilon}$ or an L3 programming layer variable, e.g.,

```
key-a:c = @itemdepth , % use \@itemdepth as the value
key-b:v = @itemdepth   % use the current value of \@itemdepth as the value
```

10 Creating equivalent keys

There are times when it is useful to have different names which map to the same underlying key interfaces, for example where a more complex higher level name is implemented by several simpler underlying keys, or where a key name needs to be changed. To support this, it is possible to define equivalent keys using `\DeclareTemplateInterface`.

```
\DeclareTemplateInterfaceEquivalents <type> <template> [<deprecation-date>] <key list>
```

Defines one or more keys to act as equivalents to the base keys of the `<template>`. As for `\DeclareTemplateInterface`, these take a specialized format:

```
<key1> ":" <key type1> "=" <definition1> ","
<key2> ":" <key type2> "=" <definition2> ","
...
```

The `<key type>` of the key being created should be one of

- **alias**: A direct equivalence between the key being created and a single parent
- **meta**: An equivalence to one or more parent keys, where additional code may be included as well as the value passed to the key being created; the value (provided in the instance declaration or in the ad hoc setting in the document) is available as `#1`.

for example

```
\DeclareTemplateInterfaceEquivalents { type } { template }
{
  alias-key-one : alias = base-key-one ,
  alias-key-two : meta =
    { parent-key-two = {#1}, base-key-three = \hspace{#1} }
}
```

Equivalent key names may be reused to overwrite existing equivalent keys but cannot be the same as base keys of the template.

Optionally, the `<deprecation date>` may be set. Before this date, a warning will be printed concerning the keys; after this date, an error will be issued. In both cases, the keys will still be defined.

10.1 Modifying existing key values

Some changes to key values are relative adjustments to the previous value, for example to add 1 cm to a margin. To support this, the syntax

```
key-a += 1cm
```

can be used, with a `+` immediately before the `=` sign. This will cause the new value to be *added* to the existing one. This naturally applies to all numeric key types, and for `tokenlist` and `commalist` means adding the new material to the end of the existing value. An error will arise if this is used for other key types, for example `function`.

11 Getting information about templates and instances

<hr/> <hr/>	<code>\ShowInstanceValues {<type>} {<instance>}</code>	Shows the <i><values></i> for an <i><instance></i> of the given <i><type></i> at the terminal.
<hr/> <hr/>	<code>\ShowTemplateCode {<type>} {<template>}</code>	Shows the <i><code></i> of a <i><template></i> for an <i><type></i> in the terminal.
<hr/> <hr/>	<code>\ShowTemplateDefaults {<type>} {<template>}</code>	Shows the <i><default></i> values of a <i><template></i> for an <i><type></i> in the terminal.
<hr/> <hr/>	<code>\ShowTemplateInterface {<type>} {<template>}</code>	Shows the <i><keys></i> and associated <i><key types></i> of a <i><template></i> for an <i><type></i> in the terminal.
<hr/> <hr/>	<code>\ShowTemplateVariables {<type>} {<template>}</code>	Shows the <i><variables></i> and associated <i><keys></i> of a <i><template></i> for an <i><type></i> in the terminal. Note that <i>code</i> and <i>choice</i> keys do not map directly to variables but to arbitrary code. For <i>choice</i> keys, each valid choice is shown as a separate entry in the list, with the key name and choice separated by a space, for example Template 'example' of type 'example' has variable mapping: > demo unknown => \def \demo {?} > demo c => \def \demo {c} > demo b => \def \demo {b} > demo a => \def \demo {a}.

would be shown for a choice key *demo* with valid choices *a*, *b* and *c*, plus code for an *unknown* branch.

12 Debugging support

<hr/> <hr/>	<code>\DebugTemplatesOn</code>	Turn on or off debugging support for use of templates. Debugging information is provided at the point of <i>use</i> of templates and instances, i.e. where normal messages should be avoided for performance reasons.
<hr/> <hr/>	<code>\DebugTemplatesOff</code>	

13 The implementation

```
1 <@@=template>
2 <*2kernel>
3 \message{templates,}
4 </2kernel>
```

```

5 <*2ekernel | latexrelease>
6 \ExplSyntaxOn
7 <latexrelease> \NewModuleRelease{2024/06/01}{lttemplates}
8 <latexrelease> {Prototype-document~commands}%

```

13.1 Extensions for expl3

A slight refinement of code from l3doc, likely should move to the core. Note that like other `compare` functions, the arguments are always fully expanded.

```

\__template_date_compare:nNnTF
\__template_date_compare_aux:nNn
\__template_date_compare_auxi:nnnNnnn
\__template_date_compare_auxii:w
\__template_date_compare_auxiii:w
9 \prg_new_conditional:Npnn \__template_date_compare:nNn #1#2#3 { TF }
10 { \use:e { \exp_not:N \__template_date_compare_aux:nNn {#1} #2 {#3} } }
11 \cs_new:Npn \__template_date_compare_aux:nNn #1#2#3
12 {
13   \use:e
14   {
15     \exp_not:N \__template_date_compare_auxi:nnnNnnn
16     \__template_date_compare_auxii:w #1 - - - \q_stop
17     #2
18     \__template_date_compare_auxii:w #3 - - - \q_stop
19   }
20 }
21 \cs_new:Npn \__template_date_compare_auxi:nnnNnnn #1#2#3#4#5#6#7
22 {
23   \int_compare:nNnTF {#1} = {#5}
24   {
25     \int_compare:nNnTF {#2} = {#6}
26     {
27       \int_compare:nNnTF {#3} #4 {#7}
28       { \prg_return_true: } { \prg_return_false: }
29     }
30     {
31       \int_compare:nNnTF {#2} #4 {#6}
32       { \prg_return_true: } { \prg_return_false: }
33     }
34   }
35   {
36     \int_compare:nNnTF {#1} #4 {#5}
37     { \prg_return_true: } { \prg_return_false: }
38   }
39 }
40 \cs_new:Npn \__template_date_compare_auxii:w #1 - #2 - #3 - #4 \q_stop
41 {
42   \tl_if_blank:nTF {#2#3}
43   { \__template_date_compare_auxiii:w #1 / / / \q_stop }
44   { { 0 #1 } { 0 #2 } { 0 #3 } }
45 }
46 \cs_new:Npn \__template_date_compare_auxiii:w #1 / #2 / #3 / #4 \q_stop
47 { { 0 #1 } { 0 #2 } { 0 #3 } }

```

(End of definition for `\__template_date_compare:nNnTF` and others.)

13.2 Variables and constants

```
\c__template_code_root_tl
\c__template_defaults_root_tl
\c__template_instances_root_tl
\c__template_keytypes_root_tl
\c__template_key_order_root_tl
\c__template_restrict_root_tl
\c__template_values_root_tl
\c__template_vars_root_tl
```

```
\c__template_msg_root_tl
```

So that literal values are kept to a minimum.

```
48 \tl_const:Nn \c__template_code_root_tl { template-code~>~ }
49 \tl_const:Nn \c__template_defaults_root_tl { template-defaults~>~ }
50 \tl_const:Nn \c__template_instances_root_tl { template-instance~>~ }
51 \tl_const:Nn \c__template_keytypes_root_tl { template-key-types~>~ }
52 \tl_const:Nn \c__template_key_order_root_tl { template-key-order~>~ }
53 \tl_const:Nn \c__template_msg_root_tl { template-message~>~ }
54 \tl_const:Nn \c__template_values_root_tl { template-values~>~ }
55 \tl_const:Nn \c__template_vars_root_tl { template-vars~>~ }
```

```
\c__template_keytypes_seq
```

A list of all keytypes.

```
56 \exp_args:NNe \seq_const_from_clist:Nn \c__template_keytypes_seq
57 {
58   \tl_to_str:n
59   {
60     boolean ,
61     choice ,
62     commalist ,
63     function ,
64     instance ,
65     integer ,
66     length ,
67     muskip ,
68     real ,
69     skip ,
70     tokenlist
71   }
72 }
```

```
\c__template_keytypes_arg_seq
```

A list of keytypes which also need additional data (an argument), used to parse the keytype correctly.

```
73 \seq_const_from_clist:Nn \c__template_keytypes_arg_seq
74 { choice , function , instance }
```

```
\g__template_type_prop
```

For storing types and the associated number of arguments.

```
75 \prop_new:N \g__template_type_prop
```

\l__template_assignments_tl

When creating an instance, the assigned values are collected here.

76 \tl_new:N \l__template_assignments_tl

\l__template_default_tl

The default value for a key is recovered here from the property list in which it is stored.

77 \tl_new:N \l__template_default_tl

\l__template_error_bool

A flag for errors to be carried forward.

78 \bool_new:N \l__template_error_bool

\l__template_global_bool

Used to indicate that assignments should be global.

79 \bool_new:N \l__template_global_bool

\l__template_key_name_tl
\l__template_keytype_tl
\l__template_keytype_arg_tl
\l__template_value_tl
\l__template_var_tl

When defining each key in a template, the name and type of the key need to be separated and stored. Any argument needed by the keytype is also stored separately.

80 \tl_new:N \l__template_key_name_tl
81 \tl_new:N \l__template_keytype_tl
82 \tl_new:N \l__template_keytype_arg_tl
83 \tl_new:N \l__template_value_tl
84 \tl_new:N \l__template_var_tl

\l__template_value_exp_str

85 \str_new:N \l__template_value_exp_str

\l__template_keytypes_prop

\l__template_key_order_seq

\l__template_value_order_seq

\l__template_vars_prop

To avoid needing too many difficult-to-follow csname assignments, various scratch token registers are used to build up data, which is then transferred

86 \prop_new:N \l__template_keytypes_prop
87 \seq_new:N \l__template_key_order_seq
88 \prop_new:N \l__template_values_prop
89 \seq_new:N \l__template_value_order_seq
90 \prop_new:N \l__template_vars_prop

\l__template_msg_prop

91 \prop_new:N \l__template_msg_prop

<code>\l__template_tmp_clist</code>	Scratch space.
<code>\l__template_tmp_dim</code>	92 <code>\clist_new:N \l__template_tmp_clist</code>
<code>\l__template_tmp_int</code>	93 <code>\dim_new:N \l__template_tmp_dim</code>
<code>\l__template_tmp_muskip</code>	94 <code>\int_new:N \l__template_tmp_int</code>
<code>\l__template_tmp_skip</code>	95 <code>\muskip_new:N \l__template_tmp_muskip</code>
<code>\l__template_tmp_tl</code>	96 <code>\skip_new:N \l__template_tmp_skip</code>
	97 <code>\tl_new:N \l__template_tmp_tl</code>

<code>\l__template_current_type_tl</code>
<code>\l__template_current_template_tl</code>

For setting the current template values.

98 `\tl_new:N \l__template_current_type_tl`
99 `\tl_new:N \l__template_current_template_tl`

`\__template_tmp:n` Scratch space.

100 `\cs_new:Npn \__template_tmp:n #1 { }`

(End of definition for __template_tmp:n.)

`\s__template_mark` Internal scan marks.

`\s__template_stop`

101 `\scan_new:N \s__template_mark`
102 `\scan_new:N \s__template_stop`

`\q__template_nil` Internal quarks.

103 `\quark_new:N \q__template_nil`

`\__template_quark_if_nil_p:n` Branching quark conditional.

`\__template_quark_if_nil:nTF`

104 `\__kernel_quark_new_conditional:Nn \__template_quark_if_nil:N { F }`

(End of definition for __template_quark_if_nil:nTF.)

13.3 Debugging support

`\__template_debug_on:`

`\__template_debug_off:`

105 `\cs_new_protected:Npn \__template_debug_on:`
106 `{`
107 `\cs_set_protected:Npn \__template_debug:n ##1 {##1}`
108 `\cs_set_protected:Npn \__template_debug_typeout:n ##1`
109 `{ \iow_term:e { [ Template ] ~ ==> ~ ##1 } }`
110 `}`
111 `\cs_new_protected:Npn \__template_debug_off:`
112 `{`
113 `\cs_set_protected:Npn \__template_debug:n ##1 { }`
114 `\cs_set_protected:Npn \__template_debug_typeout:n ##1 { }`
115 `}`

(End of definition for __template_debug_on: and __template_debug_off:.)

```

    \__template_debug:n
\__template_debug_typeout:n
116 \cs_new_protected:Npn \__template_debug:n #1 { }
117 \cs_new_protected:Npn \__template_debug_typeout:n #1 { }

(End of definition for \__template_debug:n and \__template_debug_typeout:n.)

```

13.4 Testing existence and validity

There are a number of checks needed for either the existence of a type, template or instance. There are also some for the validity of a particular call. All of these are collected up here.

`\__template_execute_if_arg_agree:nnT` A test agreement between the number of arguments for the template type and that specified when creating a template. This is not done as a separate conditional for efficiency and better error message

```

118 \cs_new_protected:Npn \__template_execute_if_arg_agree:nnT #1#2#3
119 {
120   \prop_get:NnN \g__template_type_prop {#1} \l__template_tmp_tl
121   \int_compare:nNnTF {#2} = \l__template_tmp_tl
122     {#3}
123     {
124       \msg_error:nneee { template } { argument-number-mismatch }
125       {#1} { \l__template_tmp_tl } {#2}
126     }
127 }

```

(End of definition for `\__template_execute_if_arg_agree:nnT`.)

`\__template_execute_if_code_exist:nnT` A template is only fully declared if the code has been set up, which can be checked by looking for the template function itself.

```

128 \cs_new_protected:Npn \__template_execute_if_code_exist:nnT #1#2#3
129 {
130   \cs_if_exist:cTF { \c__template_code_root_tl #1 / #2 }
131     {#3}
132     { \msg_error:nnnn { template } { no-template-code } {#1} {#2} }
133 }

```

(End of definition for `\__template_execute_if_code_exist:nnT`.)

`\__template_execute_if_keytype_exist:nT`
`\__template_execute_if_keytype_exist:VT` The test for valid keytypes looks for a function to set up the key, which is part of the “code” side of the template definition. This avoids having different lists for the two parts of the process.

```

134 \cs_new_protected:Npn \__template_execute_if_keytype_exist:nT #1#2
135 {
136   \seq_if_in:NnTF \c__template_keytypes_seq { \tl_to_str:n {#1} }
137     {#2}
138     { \msg_error:nnn { template } { unknown-keytype } {#1} }
139 }
140 \cs_generate_variant:Nn \__template_execute_if_keytype_exist:nT { V }

```

(End of definition for `\__template_execute_if_keytype_exist:nT`.)

`\_template\_execute\_if\_type\_exist:nT` To check that a particular type is valid.

```

141 \cs_new_protected:Npn \_template\_execute\_if\_type\_exist:nT #1#2
142 {
143   \prop\_if\_in:NnTF \g\_template\_type\_prop {#1}
144     {#2}
145     { \msg\_error:nnn { template } { unknown-type } {#1} }
146 }

```

(End of definition for _template_execute_if_type_exist:nT.)

`\_template\_execute\_if\_keyname\_free:nT`

`\_template\_execute\_if\_keyname\_free:VT`

```

147 \cs_new_protected:Npn \_template\_execute\_if\_keyname\_free:nT #1#2
148 {
149   \seq\_if\_in:NnTF \l\_template\_key\_order\_seq {#1}
150     { \msg\_error:nnn { template } { duplicate-key-interface } {#1} }
151     {#2}
152 }
153 \cs_generate_variant:Nn \_template\_execute\_if\_keyname\_free:nT { V }

```

(End of definition for _template_execute_if_keyname_free:nT.)

`\_template\_execute\_if\_keys\_exist:nnT`

To check that the keys for a template have been set up before trying to create any code, a simple check for the correctly-named keytype property list.

```

154 \cs_new_protected:Npn \_template\_if\_keys\_exist:nnT #1#2#3
155 {
156   \cs\_if\_exist:cTF { \c\_template\_keytypes\_root\_tl #1 / #2 }
157     {#3}
158     { \msg\_error:nnnn { template } { unknown-template } {#1} {#2} }
159 }

```

(End of definition for _template_execute_if_keys_exist:nnT.)

`\_template\_if\_key\_value:nTF`

`\_template\_if\_key\_value:VTF`

Tests for the first token in a string being `\KeyValue`.

```

160 \prg_new_conditional:Npnn \_template\_if\_key\_value:n #1 { T , F , TF }
161 {
162   \str\_if\_eq:noTF { \KeyValue } { \tl\_head:w #1 \q\_nil \q\_stop }
163     \prg\_return\_true:
164     \prg\_return\_false:
165 }
166 \prg_generate_conditional_variant:Nnn \_template\_if\_key\_value:n { V } { T , F , TF }

```

(End of definition for _template_if_key_value:nTF.)

`\_template\_if\_instance\_exist:nnTF`

Testing for an instance

```

167 \prg_new_conditional:Npnn \_template\_if\_instance\_exist:nn #1#2 { T, F, TF }
168 {
169   \cs\_if\_exist:cTF { \c\_template\_instances\_root\_tl #1 / #2 }
170     \prg\_return\_true:
171     \prg\_return\_false:
172 }

```

(End of definition for _template_if_instance_exist:nnTF.)

```

\__template_if_use_template:nTF Tests for the first token in a string being \UseTemplate.
173 \prg_new_conditional:Npnn \__template_if_use_template:n #1 { TF }
174 {
175   \str_if_eq:noTF { \UseTemplate } { \tl_head:w #1 \q_nil \q_stop }
176   \prg_return_true:
177   \prg_return_false:
178 }

```

(End of definition for __template_if_use_template:nTF.)

13.5 Saving and recovering property lists

The various property lists for templates have to be shuffled in and out of storage.

The defaults and keytypes are transferred from the scratch property lists to the “proper” lists for the template being created.

```

\__template_store_defaults:nn
\__template_store_keytypes:nn
\__template_store_values:nn
\__template_store_vars:nn
179 \cs_new_protected:Npn \__template_store_defaults:nn #1#2
180 {
181   \debug_suspend:
182   \prop_gclear_new:c { \c__template_defaults_root_tl #1 / #2 }
183   \prop_gset_eq:cN { \c__template_defaults_root_tl #1 / #2 }
184   \l__template_values_prop
185   \debug_resume:
186 }
187 \cs_new_protected:Npn \__template_store_keytypes:nn #1#2
188 {
189   \debug_suspend:
190   \prop_if_exist:cTF { \c__template_keytypes_root_tl #1 / #2 }
191   {
192     \msg_info:nnnn { template } { declare-template-interface } {#1} {#2}
193     \prop_gclear:c { \c__template_keytypes_root_tl #1 / #2 }
194   }
195   { \prop_new:c { \c__template_keytypes_root_tl #1 / #2 } }
196   \prop_gset_eq:cN { \c__template_keytypes_root_tl #1 / #2 }
197   \l__template_keytypes_prop
198   \seq_gclear_new:c { \c__template_key_order_root_tl #1 / #2 }
199   \seq_gset_eq:cN { \c__template_key_order_root_tl #1 / #2 }
200   \l__template_key_order_seq
201   \debug_resume:
202 }
203 \cs_new_protected:Npn \__template_store_values:nn #1#2
204 {
205   \debug_suspend:
206   \prop_gclear_new:c { \c__template_values_root_tl #1 / #2 }
207   \prop_set_eq:cN { \c__template_values_root_tl #1 / #2 }
208   \l__template_values_prop
209   \debug_resume:
210 }
211 \cs_new_protected:Npn \__template_store_vars:nn #1#2
212 {
213   \debug_suspend:
214   \prop_gclear_new:c { \c__template_msg_root_tl #1 / #2 }
215   \prop_gset_eq:cN { \c__template_msg_root_tl #1 / #2 }
216   \l__template_msg_prop

```

```

217 \prop_gclear_new:c { \c__template_vars_root_tl #1 / #2 }
218 \prop_gset_eq:cN { \c__template_vars_root_tl #1 / #2 }
219 \l__template_vars_prop
220 \debug_resume:
221 }

```

(End of definition for `\__template_store_defaults:nn` and others.)

```

\__template_recover_defaults:nn
\__template_recover_keytypes:nn
\__template_recover_values:nn
\__template_recover_vars:nn

```

Recovering the stored data for a template is rather less complex than storing it. All that happens is the data is transferred from the permanent to the scratch storage. However, we need to check the scratch storage does exist.

```

222 \cs_new_protected:Npn \__template_recover_defaults:nn #1#2
223 {
224   \prop_if_exist:cTF
225   { \c__template_defaults_root_tl #1 / #2 }
226   {
227     \prop_set_eq:Nc \l__template_values_prop
228     { \c__template_defaults_root_tl #1 / #2 }
229   }
230   { \prop_clear:N \l__template_values_prop }
231 }
232 \cs_new_protected:Npn \__template_recover_keytypes:nn #1#2
233 {
234   \prop_if_exist:cTF
235   { \c__template_keytypes_root_tl #1 / #2 }
236   {
237     \prop_set_eq:Nc \l__template_keytypes_prop
238     { \c__template_keytypes_root_tl #1 / #2 }
239   }
240   { \prop_clear:N \l__template_keytypes_prop }
241   \seq_if_exist:cTF { \c__template_key_order_root_tl #1 / #2 }
242   {
243     \seq_set_eq:Nc \l__template_key_order_seq
244     { \c__template_key_order_root_tl #1 / #2 }
245   }
246   { \seq_clear:N \l__template_key_order_seq }
247 }
248 \cs_new_protected:Npn \__template_recover_values:nn #1#2
249 {
250   \prop_if_exist:cTF
251   { \c__template_values_root_tl #1 / #2 }
252   {
253     \prop_set_eq:Nc \l__template_values_prop
254     { \c__template_values_root_tl #1 / #2 }
255   }
256   { \prop_clear:N \l__template_values_prop }
257 }
258 \cs_new_protected:Npn \__template_recover_vars:nn #1#2
259 {
260   \prop_if_exist:cTF
261   { \c__template_msg_root_tl #1 / #2 }
262   {
263     \prop_set_eq:Nc \l__template_msg_prop
264     { \c__template_msg_root_tl #1 / #2 }

```

```

265     }
266     { \prop_clear:N \l__template_msg_prop }
267   \prop_if_exist:cTF
268     { \c__template_vars_root_tl #1 / #2 }
269     {
270       \prop_set_eq:Nc \l__template_vars_prop
271       { \c__template_vars_root_tl #1 / #2 }
272     }
273   { \prop_clear:N \l__template_vars_prop }
274 }

```

(End of definition for `\__template_recover_defaults:nn` and others.)

13.6 Creating new template types

`\__template_define_type:nn`
`\__template_declare_type:nn`

Although the type is the “top level” of the template system, it is actually very easy to implement. All that happens is that the number of arguments required is recorded, indexed by the name of the type.

```

275 \cs_new_protected:Npn \__template_define_type:nn #1#2
276 {
277   \prop_if_in:NnTF \g__template_type_prop {#1}
278     { \msg_error:nnn { template } { type-already-defined } {#1} }
279     { \__template_declare_type:nn {#1} {#2} }
280 }
281 \cs_new_protected:Npn \__template_declare_type:nn #1#2
282 {
283   \int_set:Nn \l__template_tmp_int {#2}
284   \int_compare:nTF { 0 <= \l__template_tmp_int <= 9 }
285     {
286       \msg_info:nnnV { template } { declare-type }
287       {#1} \l__template_tmp_int
288       \prop_gput:NnV \g__template_type_prop {#1}
289       \l__template_tmp_int
290     }
291     {
292       \msg_error:nnnV { template } { bad-number-of-arguments }
293       {#1} \l__template_tmp_int
294     }
295 }

```

(End of definition for `\__template_define_type:nn` and `\__template_declare_type:nn`.)

13.7 Design part of template declaration

The “design” part of a template declaration defines the general behaviour of each key, and possibly a default value. However, it does not include the implementation. This means that what happens here is the two properties are saved to appropriate lists, which can then be used later to recover the information when implementing the keys.

`\__template_declare_template_keys:nnnn`

The main function for the “design” part of creating a template starts by checking that the type exists and that the number of arguments required agree. If that is all fine, then the two storage areas for defaults and keytypes are initialized. The mechanism is then

set up for the l3keys module to actually parse the keys. Finally, the code hands of to the storage routine to save the parsed information properly.

```

296 \cs_new_protected:Npn \__template_declare_template_keys:nnnn #1#2#3#4
297 {
298   \__template_execute_if_type_exist:nT {#1}
299   {
300     \__template_execute_if_arg_agree:nnT {#1} {#3}
301     {
302       \prop_clear:N \l__template_values_prop
303       \prop_clear:N \l__template_keytypes_prop
304       \prop_clear:N \l__template_vars_prop
305       \seq_clear:N \l__template_key_order_seq
306       \keyval_parse:NNn
307       \__template_parse_keys_elt:n \__template_parse_keys_elt:nn {#4}
308       \__template_store_defaults:nn {#1} {#2}
309       \__template_store_keytypes:nn {#1} {#2}
310       \__template_store_vars:nn {#1} {#2}
311     }
312   }
313 }

```

(End of definition for __template_declare_template_keys:nnnn.)

__template_parse_keys_elt:n Processing the key part of the key–value pair is always carried out using this function, even if a value was found. First, the key name is separated from the keytype, and if necessary the keytype is separated into two parts. This information is then used to check that the keytype is valid, before storing the keytype (plus argument if necessary) as a property of the key name. The key name is also stored (in braces) in the token list to record the order the keys are defined in.

```

314 \cs_new_protected:Npn \__template_parse_keys_elt:n #1
315 {
316   \__template_split_keytype:n {#1}
317   \bool_if:NF \l__template_error_bool
318   {
319     \__template_execute_if_keytype_exist:VT \l__template_keytype_tl
320     {
321       \seq_map_function:NN \c__template_keytypes_arg_seq
322       \__template_parse_keys_elt_aux:n
323       \bool_if:NF \l__template_error_bool
324       {
325         \__template_execute_if_keyname_free:VT
326         \l__template_key_name_tl
327         { \__template_parse_keys_elt_aux: }
328       }
329     }
330   }
331 }
332 \cs_new_protected:Npn \__template_parse_keys_elt_aux:n #1
333 {
334   \str_if_eq:VnT \l__template_keytype_tl {#1}
335   {
336     \tl_if_empty:NT \l__template_keytype_arg_tl
337     {
338       \msg_error:nnn { template } { keytype-requires-argument } {#1}

```



```

339         \bool_set_true:N \l__template_error_bool
340         \seq_map_break:
341     }
342 }
343 }
344 \cs_new_protected:Npn \__template_parse_keys_elt_aux:
345 {
346     \tl_set:Nx \l__template_tmp_tl
347     {
348         \l__template_keytype_tl
349         \tl_if_empty:NF \l__template_keytype_arg_tl
350         { { \l__template_keytype_arg_tl } }
351     }
352     \prop_put:NVV \l__template_keytypes_prop \l__template_key_name_tl
353     \l__template_tmp_tl
354     \seq_put_right:NV \l__template_key_order_seq \l__template_key_name_tl
355     \str_if_eq:VnT \l__template_keytype_tl { choice }
356     {
357         \clist_if_in:NnT \l__template_keytype_arg_tl { unknown }
358         { \msg_error:nn { template } { choice-unknown-reserved } }
359     }
360 }

```

(End of definition for __template_parse_keys_elt:n, __template_parse_keys_elt_aux:n, and __template_parse_keys_elt_aux:.)

__template_parse_keys_elt:nn For keys which have a default, the keytype and key name are first separated out by the __template_parse_keys_elt:n routine, before storing the default value in the scratch property list.

```

361 \cs_new_protected:Npn \__template_parse_keys_elt:nn #1#2
362 {
363     \__template_parse_keys_elt:n {#1}
364     \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#2}
365 }

```

(End of definition for __template_parse_keys_elt:nn.)

__template_split_keytype:n The keytype and key name should be separated by :. As the definition might be given inside or outside of a code block, the category code of colons is standardized. After that, the standard delimited argument method is used to separate the two parts.

```

366 \cs_new_protected:Npe \__template_split_keytype:n #1
367 {
368     \exp_not:N \bool_set_false:N \exp_not:N \l__template_error_bool
369     \tl_set:Nn \exp_not:N \l__template_tmp_tl {#1}
370     \tl_replace_all:Nnn \exp_not:N \l__template_tmp_tl { : } { \token_to_str:N : }
371     \tl_if_in:VnTF \exp_not:N \l__template_tmp_tl { \token_to_str:N : }
372     {
373         \exp_not:n
374         {
375             \tl_clear:N \l__template_key_name_tl
376             \exp_after:wN \__template_split_keytype_aux:w
377             \l__template_tmp_tl \s__template_stop
378         }
379     }

```

```

380     {
381         \exp_not:N \bool_set_true:N \exp_not:N \l__template_error_bool
382         \msg_error:nnn { template } { missing-keytype } {#1}
383     }
384 }
385 \use:e
386 {
387     \cs_new_protected:Npn \exp_not:N \__template_split_keytype_aux:w
388     #1 \token_to_str:N : #2 \s__template_stop
389     {
390         \tl_put_right:Ne \exp_not:N \l__template_key_name_tl
391         {
392             \exp_not:N \tl_trim_spaces:e
393             { \exp_not:N \tl_to_str:n {#1} }
394         }
395         \tl_if_in:nnTF {#2} { \token_to_str:N : }
396         {
397             \tl_put_right:Nn \exp_not:N \l__template_key_name_tl
398             { \token_to_str:N : }
399             \exp_not:N \__template_split_keytype_aux:w #2 \s__template_stop
400         }
401         {
402             \exp_not:N \tl_if_empty:NTF \exp_not:N \l__template_key_name_tl
403             {
404                 \msg_error:nnn { template } { empty-key-name }
405                 { \token_to_str:N : #2 }
406             }
407             { \exp_not:N \__template_split_keytype_arg:n {#2} }
408         }
409     }
410 }

```

(End of definition for __template_split_keytype:n and __template_split_keytype_aux:w.)

__template_split_keytype_arg:n
 __template_split_keytype_arg:V
 __template_split_keytype_arg_aux:n
 __template_split_keytype_arg_aux:w

The second stage of sorting out the keytype is to check for an argument. As there is no convenient delimiting token to look for, a check is made instead for each possible text value for the keytype. To keep things faster, this only involves the keytypes that need an argument. If a match is made, then a check is also needed to see that it is at the start of the keytype information. All being well, the split can then be applied. Any non-matching keytypes are assumed to be “correct” as given, and are left alone (this is checked by other code).

```

411 \cs_new_protected:Npn \__template_split_keytype_arg:n #1
412 {
413     \tl_set:Ne \l__template_keytype_tl { \tl_trim_spaces:n {#1} }
414     \tl_clear:N \l__template_keytype_arg_tl
415     \cs_set_protected:Npn \__template_split_keytype_arg_aux:n ##1
416     {
417         \tl_if_in:nnT {#1} {##1}
418         {
419             \cs_set:Npn \__template_split_keytype_arg_aux:w
420             #####1 ##1 #####2 \s__template_stop
421             {
422                 \tl_if_blank:nT {#####1}
423                 {

```

```

424         \tl_set:Nc \l__template_keytype_tl
425         { \tl_trim_spaces:n {##1} }
426         \tl_if_blank:nF {###2}
427         {
428             \tl_set:Nc \l__template_keytype_arg_tl
429             { \use:n ###2 }
430         }
431         \seq_map_break:
432     }
433 }
434 \__template_split_keytype_arg_aux:w #1 \s__template_stop
435 }
436 }
437 \seq_map_function:NN \c__template_keytypes_arg_seq
438     \__template_split_keytype_arg_aux:n
439 }
440 \cs_generate_variant:Nn \__template_split_keytype_arg:n { V }
441 \cs_new:Npn \__template_split_keytype_arg_aux:n #1 { }
442 \cs_new:Npn \__template_split_keytype_arg_aux:w #1 \s__template_stop { }

```

(End of definition for __template_split_keytype_arg:n, __template_split_keytype_arg_aux:n,
and __template_split_keytype_arg_aux:w.)

13.8 Equivalent interfaces

__template_declare_template_equiv:nnnn

```

443 \cs_new_protected:Npn \__template_declare_template_equiv:nnnn #1#2#3#4
444 {
445     \__template_execute_if_type_exist:nT {#1}
446     {
447         \__template_if_keys_exist:nnT {#1} {#2}
448         {
449             \__template_recover_keytypes:nn {#1} {#2}
450             \__template_recover_vars:nn {#1} {#2}
451             \keyval_parse:nnn
452             { \__template_parse_equiv_elt:n }
453             { \__template_parse_equiv_elt:nnnn { #1 / #2 } {#3} }
454             {#4}
455             \__template_store_keytypes:nn {#1} {#2}
456             \__template_store_vars:nn {#1} {#2}
457         }
458     }
459 }

```

(End of definition for __template_declare_template_equiv:nnnn.)

__template_parse_equiv_elt:n

We must have key(s) to map to.

```

460 \cs_new_protected:Npn \__template_parse_equiv_elt:n #1
461 { \msg_error:nnn { template } { key-no-equiv } {#1} }

```

(End of definition for __template_parse_equiv_elt:n.)

__template_parse_equiv_elt:nnnn

Most of the work is deferred here: the only thing done is to sanity-check the key names to avoid duplication.

__template_parse_equiv_elt_aux:nnn

__template_parse_equiv_alias:nnnn

__template_parse_equiv_meta:nnnn

__template_parse_equiv_auxi:nnnn

__template_parse_equiv_auxii:n

__template_parse_equiv_auxiii:nnnn

```

462 \cs_new_protected:Npn \__template_parse_equiv_elt:nnnn #1#2#3#4
463 {
464   \__template_split_keytype:n {#3}
465   \bool_if:NF \l__template_error_bool
466   {
467     \prop_get:NVNTF \l__template_keytypes_prop
468     \l__template_key_name_tl \l__template_tmp_tl
469     {
470       \seq_if_in:NeTF \c__template_keytypes_seq
471       { \tl_to_str:N \l__template_tmp_tl }
472       {
473         \msg_error:nnV { template } { equiv-key-interface }
474         \l__template_key_name_tl
475       }
476       { \__template_parse_equiv_elt_aux:nnn {#1} {#2} {#4} }
477     }
478     { \__template_parse_equiv_elt_aux:nnn {#1} {#2} {#4} }
479   }
480 }
481 \cs_new_protected:Npn \__template_parse_equiv_elt_aux:nnn #1#2#3
482 {
483   \cs_if_exist_use:cF
484   { __template_parse_equiv_ \l__template_keytype_tl :nnn }
485   {
486     \msg_error:nnV { template } { unknown-equiv-keytype }
487     \l__template_keytype_tl
488     \use_none:nnn
489   }
490   {#1} {#2} {#3}
491 }

```

As we only need the key name here, we use an inline mapping and the same code for aliases and meta keys. Note the need to detokenize key names for the sequence check.

```

492 \cs_new_protected:Npn \__template_parse_equiv_alias:nnn #1#2#3
493 { \__template_parse_equiv_auxi:nnnn {#1} {#2} { #3 = {##1} } {#3} }
494 \cs_new_protected:Npn \__template_parse_equiv_meta:nnn #1#2#3
495 { \__template_parse_equiv_auxi:nnnn {#1} {#2} {#3} {#3} }
496 \cs_new_protected:Npn \__template_parse_equiv_auxi:nnnn #1#2#3#4
497 {
498   \keyval_map_inline:nnn {#4}
499   { \__template_parse_equiv_auxii:n {##1} }
500   { \__template_parse_equiv_auxii:n {##1} }
501   \__template_parse_equiv_auxiii:nnnn {#1} {#2} {#3} {#4}
502 }
503 \cs_new_protected:Npn \__template_parse_equiv_auxii:n #1
504 {
505   \seq_if_in:NeF \l__template_key_order_seq { \tl_to_str:n {#1} }
506   {
507     \msg_error:nnn { template } { unknown-key } {#1}
508     \keyval_map_break:n { \use_none:nn }
509   }
510 }
511 \cs_new_protected:Npn \__template_parse_equiv_auxiii:nnnn #1#2#3#4
512 {

```

```

513 \prop_put:NVV \l__template_keytypes_prop \l__template_key_name_tl
514 \l__template_keytype_tl
515 \seq_put_right:NV \l__template_key_order_seq \l__template_key_name_tl
516 \prop_put:NVn \l__template_vars_prop \l__template_key_name_tl {#4}
517 \tl_if_empty:nF {#2}
518 {
519 \prop_put:NVe \l__template_msg_prop \l__template_key_name_tl
520 { \exp_not:n {#2} { \exp_not:V \l__template_key_name_tl } }
521 }
522 }

```

(End of definition for __template_parse_equiv_elt:nnnn and others.)

13.9 Implementation part of template declaration

__template_declare_template_code:nnnnn
__template_declare_template_code:nnnn

The main function for implementing a template starts with a couple of simple checks to make sure that there are no obvious mistakes: the number of arguments must agree and the template keys must have been declared.

```

523 \cs_new_protected:Npn \__template_declare_template_code:nnnnn #1#2#3#4#5
524 {
525 \__template_execute_if_type_exist:nT {#1}
526 {
527 \__template_execute_if_arg_agree:nnT {#1} {#3}
528 {
529 \__template_if_keys_exist:nnT {#1} {#2}
530 {
531 \__template_store_key_implementation:nnn {#1} {#2} {#4}
532 \str_if_in:nnTF {#5} { AssignTemplateKeys }
533 {
534 \regex_match:nnTF { \c { AssignTemplateKeys } } {#5}
535 { \__template_declare_template_code:nnnn {#1} {#2} {#3} {#5} }
536 {
537 \__template_declare_template_code:nnnn
538 {#1} {#2} {#3} { \AssignTemplateKeys #5 }
539 }
540 }
541 {
542 \__template_declare_template_code:nnnn
543 {#1} {#2} {#3} { \AssignTemplateKeys #5 }
544 }
545 }
546 }
547 }
548 }
549 \cs_new_protected:Npn \__template_declare_template_code:nnnn #1#2#3#4
550 {
551 \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
552 { \msg_info:nnnn { template } { declare-template-code } {#1} {#2} }
553 \cs_generate_from_arg_count:cNnn
554 { \c__template_code_root_tl #1 / #2 }
555 \cs_gset_protected:Npn {#3} {#4}
556 }

```

(End of definition for __template_declare_template_code:nnnnn and
__template_declare_template_code:nnnn.)

_template_store_key_implementation:nnn

Actually storing the implementation part of a template is quite easy as it only requires the list of keys given to be turned into a property list. There is also some error-checking to do, hence the need to have the list of defined keytypes available. In certain cases (when choices are involved) parsing the key results in changes to the default values. That is why they are loaded and then saved again.

```
557 \\cs_new_protected:Npn \\_template_store_key_implementation:nnn #1#2#3
558 {
559   \\_template_recover_defaults:nn {#1} {#2}
560   \\_template_recover_keytypes:nn {#1} {#2}
561   \\_template_recover_vars:nn {#1} {#2}
562   \\keyval_parse:nnn
563     { \\_template_parse_vars_elt:n } { \\_template_parse_vars_elt:nnn { #1 / #2 } } {#3}
564   \\_template_store_vars:nn {#1} {#2}
565   \\prop_map_inline:Nn \\l__template_keytypes_prop
566     {
567       \\bool_lazy_or:nnF
568       { \\str_if_eq_p:nn {##2} { alias } }
569       { \\str_if_eq_p:nn {##2} { meta } }
570       {
571         \\msg_error:nnnnn { template } { key-not-implemented }
572         {##1} {#2} {#1}
573       }
574     }
575 }
```

(End of definition for _template_store_key_implementation:nnn.)

_template_parse_vars_elt:n

At the implementation stage, every key must have a value given. So this is an error function.

```
576 \\cs_new_protected:Npn \\_template_parse_vars_elt:n #1
577 { \\msg_error:nnn { template } { key-no-variable } {#1} }
```

(End of definition for _template_parse_vars_elt:n.)

_template_parse_vars_elt:nnn
_template_parse_vars_elt_auxi:nn
_template_parse_vars_elt_auxii:nw
_template_parse_vars_elt_auxiii:nnn
_template_parse_vars_elt_auxiii:enn
_template_parse_vars_elt_auxiv:nnw
_template_parse_vars_elt_auxv:nnn
_template_parse_vars_elt_auxv:enn

The actual storage part here is very simple: the storage bin name is placed into the property list. At the same time, a comparison is made with the keytypes defined earlier: if there is a mismatch then an error is raised.

```
578 \\cs_new_protected:Npn \\_template_parse_vars_elt:nnn #1#2#3
579 {
580   \\tl_set:Ne \\l__template_key_name_tl
581     { \\tl_trim_spaces:e { \\tl_to_str:n {#2} } }
582   \\prop_get:NVNTF \\l__template_keytypes_prop
583     \\l__template_key_name_tl
584     \\l__template_keytype_tl
585     {
586       \\_template_split_keytype_arg:V \\l__template_keytype_tl
587       \\_template_parse_vars_elt_auxi:nn {#1} {#3}
588       \\prop_remove:NV \\l__template_keytypes_prop \\l__template_key_name_tl
589     }
590   { \\msg_error:nnn { template } { unknown-key } {#2} }
591 }
```

Split off any leading global and they look for the way to implement.

```

592 \cs_new_protected:Npn \__template_parse_vars_elt_auxi:nn #1#2
593 {
594   \__template_parse_vars_elt_auxii:nw {#1} #2 global global \s__template_stop
595 }
596 \cs_new_protected:Npn \__template_parse_vars_elt_auxii:nw
597 #1#2 global #3 global #4 \s__template_stop
598 {
599   \tl_if_blank:nTF {#4}
600   { \__template_parse_vars_elt_auxiii:nnn {#2} {#1} { } }
601   {
602     \tl_if_blank:nTF {#2}
603     {
604       \__template_parse_vars_elt_auxiii:enn
605       { \tl_trim_spaces:n {#3} }
606       {#1} { global }
607     }
608     { \msg_error:nnn { template } { bad-variable } { #2 global #3 } }
609   }
610 }
611 \cs_new_protected:Npn \__template_parse_vars_elt_auxiii:nnn #1#2#3
612 {
613   \cs_if_exist:cTF { __template_parse_vars_ \l__template_keytype_tl :nnn }
614   {
615     \use:c { __template_parse_vars_ \l__template_keytype_tl :nnn }
616     {#1} {#2} {#3}
617   }
618   {
619     \__template_parse_vars_elt_auxiv:nnw
620     {#2} {#3} #1 name name \s__template_stop
621   }
622 }
623 \cs_generate_variant:Nn \__template_parse_vars_elt_auxiii:nnn { e }
624 \cs_new_protected:Npn \__template_parse_vars_elt_auxiv:nnw
625 #1#2#3 name #4 name #5 \s__template_stop
626 {
627   \tl_if_blank:nTF {#5}
628   { \__template_parse_vars_elt_auxv:nnn {#3} {#1} {#2} }
629   {
630     \tl_if_blank:nTF {#3}
631     {
632       \__template_parse_vars_elt_auxv:enn
633       { \exp_not:c { \tl_trim_spaces:n {#4} } } {#1} {#2}
634     }
635     { \msg_error:nnn { template } { bad-variable } { #3 name #4 } }
636   }
637 }
638 \cs_new_protected:Npn \__template_parse_vars_elt_auxv:nnn #1#2#3
639 {
640   \tl_if_single:nTF {#1}
641   {
642     \cs_if_exist:NF #1
643     { \use:c { __template_map_var_type: _new:N } #1 }
644     \prop_put:NVn \l__template_vars_prop

```

```

645         \l__template_key_name_tl {#3#1}
646     }
647     { \msg_error:nnn { template } { bad-variable } {#3#1} }
648 }
649 \cs_generate_variant:Nn \__template_parse_vars_elt_auxv:nnn { e }

```

(End of definition for __template_parse_vars_elt:nnn and others.)

__template_parse_vars_choice:nnn
__template_parse_vars_function:nnn
__template_parse_vars_instance:nnn

The various special cases.

```

650 \cs_new_protected:Npn \__template_parse_vars_choice:nnn #1#2#3
651 { \__template_implement_choices:nn {#2} {#1} }
652 \cs_new_protected:Npn \__template_parse_vars_function:nnn #1#2#3
653 {
654     \cs_if_exist:NF #1
655     { \cs_new:Npn #1 { } }
656     \prop_put:NVn \l__template_vars_prop
657     \l__template_key_name_tl {#3#1}
658 }
659 \cs_new_protected:Npn \__template_parse_vars_instance:nnn #1#2#3
660 {
661     \prop_put:NVn \l__template_vars_prop
662     \l__template_key_name_tl {#3#1}
663 }

```

(End of definition for __template_parse_vars_choice:nnn, __template_parse_vars_function:nnn, and __template_parse_vars_instance:nnn.)

__template_map_var_type: Turn a “friendly” variable type into an expl3 one.

```

664 \cs_new:Npn \__template_map_var_type:
665 {
666     \str_case:Vn \l__template_keytype_tl
667     {
668         { boolean } { bool }
669         { commalist } { clist }
670         { integer } { int }
671         { length } { dim }
672         { muskip } { muskip }
673         { real } { fp }
674         { skip } { skip }
675         { tokenlist } { tl }
676     }
677 }

```

(End of definition for __template_map_var_type:.)

__template_implement_choices:nn
__template_implement_choices_default:

Implementing choices requires a second key–value loop. So after a little set-up, the standard parser is called.

```

678 \cs_new_protected:Npn \__template_implement_choices:nn #1#2
679 {
680     \clist_set:NV \l__template_tmp_clist \l__template_keytype_arg_tl
681     \prop_put:NVn \l__template_vars_prop \l__template_key_name_tl { }
682     \keyval_parse:nnn
683     { \__template_implement_choice_elt:n }
684     { \__template_implement_choice_elt:nnn {#1} }
685     {#2}

```



```

686 \prop_get:NVNT \l__template_values_prop \l__template_key_name_tl
687 \l__template_tmp_tl
688 { \__template_implement_choices_default: }
689 \clist_if_empty:NF \l__template_tmp_clist
690 {
691   \clist_map_inline:Nn \l__template_tmp_clist
692     { \msg_error:nnn { template } { choice-not-implemented } {##1} }
693 }
694 }

```

A sanity check for the default value, so that an error is raised now and not when converting to assignments.

```

695 \cs_new_protected:Npn \__template_implement_choices_default:
696 {
697   \tl_set:Nc \l__template_tmp_tl
698     { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
699   \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
700   {
701     \tl_set:Nc \l__template_tmp_tl
702       { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
703     \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
704     {
705       \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
706       \l__template_tmp_tl
707       \__template_split_keytype_arg:V \l__template_tmp_tl
708       \prop_get:NVN \l__template_values_prop \l__template_key_name_tl
709       \l__template_tmp_tl
710       \msg_error:nnVV { template } { unknown-default-choice }
711       \l__template_key_name_tl
712       \l__template_key_name_tl
713     }
714   }
715 }

```

(End of definition for __template_implement_choices:nn and
__template_implement_choices_default:.)

```

\__template_implement_choice_elt:nnn
\__template_implement_choice_elt_aux:nnn
\__template_implement_choice_elt_aux:n
\__template_implement_choice_elt:n

```

The actual storage of the implementation of a choice is mainly about error checking. The code here ensures that all choices have to have been declared, apart from the special **unknown** choice, which must come last. The code for each choice is stored along with the key name in the variables property list.

```

716 \cs_new_protected:Npn \__template_implement_choice_elt:nnn #1#2#3
717 {
718   \clist_if_empty:NTF \l__template_tmp_clist
719   {
720     \str_if_eq:nnTF {#2} { unknown }
721     { \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3} }
722     { \__template_implement_choice_elt_aux:n {#2} }
723   }
724   {
725     \clist_if_in:NnTF \l__template_tmp_clist {#2}
726     {
727       \clist_remove_all:Nn \l__template_tmp_clist {#2}
728       \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3}

```

```

729     }
730     { \_template_implement_choice_elt_aux:n {#2} }
731   }
732 }
733 \cs_new_protected:Npn \_template_implement_choice_elt_aux:n #1
734 {
735   \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
736   \l__template_tmp_tl
737   \_template_split_keytype_arg:V \l__template_tmp_tl
738   \msg_error:nnVn { template } { unknown-choice } \l__template_key_name_tl {#1}
739 }
740 \cs_new_protected:Npn \_template_implement_choice_elt_aux:nnn #1#2#3
741 {
742   \tl_set:Ne \l__template_tmp_tl
743   { \l__template_key_name_tl \c_space_tl #2 }
744   \prop_put:NVN \l__template_vars_prop \l__template_tmp_tl {#3}
745 }
746 \cs_new_protected:Npn \_template_implement_choice_elt:n #1
747 {
748   \msg_error:nnVn { template } { choice-requires-code }
749   \l__template_key_name_tl {#1}
750 }

```

(End of definition for _template_implement_choice_elt:nnn and others.)

13.10 Editing template defaults

_template_edit_defaults:nnn

Editing the template defaults means getting the values back out of the store, then parsing the list of new values before putting the updated list back into storage.

```

751 \cs_new_protected:Npn \_template_edit_defaults:nnn #1#2#3
752 {
753   \_template_if_keys_exist:nnT {#1} {#2}
754   {
755     \_template_recover_defaults:nn {#1} {#2}
756     \_template_parse_values:nnn {#1} {#2} {#3}
757     \_template_store_defaults:nn {#1} {#2}
758   }
759 }

```

(End of definition for _template_edit_defaults:nnn.)

_template_parse_values:nnn

_template_parse_known_values:nnn

_template_parse_values:NNnnn

The routine to parse values is the same for both editing a template and setting up an instance. So the code here does only the minimum necessary for reading the values. To allow code sharing, there are two code paths for a missing key: error here or store the information for later processing.

```

760 \cs_new_protected:Npn \_template_parse_values:nnn
761 { \_template_parse_values:NNnnn \use_i:nn \_template_parse_unknown_error:Vn }
762 \cs_new_protected:Npn \_template_parse_known_values:nnn
763 { \_template_parse_values:NNnnn \use_ii:nn \_template_parse_unknown_store:Vn }
764 \cs_new_protected:Npn \_template_parse_values:NNnnn #1#2#3#4#5
765 {
766   \_template_recover_keytypes:nn {#3} {#4}
767   \seq_clear:N \l__template_value_order_seq
768   \keyval_parse:nnn

```

```

769     { \_template_parse_values_elt:Nnn #1#2 }
770     { \_template_parse_values_elt:Nnn #2 }
771     {#5}
772 }

```

(End of definition for _template_parse_values:nnn, _template_parse_known_values:nnn, and _template_parse_values:NNnnn.)

_template_parse_values_elt:Nnn Almost key needs a value, so this is just an error routine other than for booleans.

```

773 \cs_new_protected:Npn \_template_parse_values_elt:Nnn #1#2#3
774 {
775   \prop_get:NnN \l__template_keytypes_prop {#3} \l__template_tmp_tl
776   \str_if_eq:VnTF \l__template_tmp_tl { boolean }
777   { \_template_parse_values_elt:Nnn #2 {#3} { true } }
778   {
779     #1
780     {
781       \bool_set_true:N \l__template_error_bool
782       \msg_error:nnn { template } { key-no-value } {#3}
783     }
784     { \clist_put_right:Nn \UnusedTemplateKeys {#3} }
785   }
786 }

```

(End of definition for _template_parse_values_elt:NNn.)

_template_parse_values_elt:Nnn To store the value, find the keytype then call the saving function. These need the current key name, stored in \l__template_key_name_tl.

```

787 \cs_new_protected:Npn \_template_parse_values_elt:Nnn #1#2#3
788 {
789   \use:e
790   {
791     \_template_parse_values_elt_auxi:w
792     \tl_trim_spaces:e { \tl_to_str:n { #2 : n : } }
793     \exp_not:N \q_stop
794   }
795   \cs_if_exist:cTF { __template_parse_values_set: \l__template_value_exp_str }
796   {
797     \prop_get:NVNTF \l__template_keytypes_prop \l__template_key_name_tl
798     \l__template_tmp_tl
799     { \_template_parse_values_elt_auxii:nn { set } }
800     { \_template_parse_values_elt_auxiii:Nn #1 }
801     {#3}
802   }
803   {
804     \msg_error:nnV { template } { unknown-expansion }
805     \l__template_value_exp_str
806   }
807 }
808 \use:e
809 {
810   \cs_new_protected:Npn \exp_not:N \_template_parse_values_elt_auxi:w
811   #1 \token_to_str:N : #2 \token_to_str:N : #3 \exp_not:N \q_stop
812 }
813 {

```

```

814     \tl_set:Nn \l__template_key_name_tl {#1}
815     \str_set:Nn \l__template_value_exp_str {#2}
816   }
817 \cs_new_protected:Npn \__template_parse_values_elt_auxii:nn #1#2
818 {
819   \__template_split_keytype_arg:V \l__template_tmp_tl
820   \cs_if_exist:cTF
821     { __template_parse_values_ \l__template_keytype_tl :nn }
822     {
823       \prop_get:NVNT \l__template_msg_prop \l__template_key_name_tl
824       \l__template_tmp_tl
825       { \l__template_tmp_tl }
826       \use:c { __template_parse_values_ \l__template_keytype_tl :nn } {#1}
827     }
828     { \use:c { __template_parse_values_ #1 : \l__template_value_exp_str } }
829     {#2}
830 }

```

Hunt for a trailing +: this allows the += syntax, treating the token as part of the mid-part not the key name.

```

831 \cs_new_protected:Npn \__template_parse_values_elt_auxiii:Nn #1#2
832 {
833   \exp_after:wN \__template_parse_values_elt_auxiv:w
834   \l__template_key_name_tl \q_nil + \q_nil \q_stop {#2} #1
835 }

```

We need the key name after dropping the + to see if it exists, but if the key name is not found, the + has to be restored.

```

836 \cs_new_protected:Npn \__template_parse_values_elt_auxiv:w #1 + \q_nil #2 \q_stop #3#4
837 {
838   \tl_if_empty:nTF {#2}
839     { #4 \l__template_key_name_tl }
840     {
841       \tl_set:Nc \l__template_key_name_tl { \tl_trim_spaces:n {#1} }
842       \prop_get:NVNTF \l__template_keytypes_prop \l__template_key_name_tl
843       \l__template_tmp_tl
844       { \__template_parse_values_elt_auxii:nn { add } }
845       {
846         \tl_put_right:Nn \l__template_key_name_tl { + }
847         #4 \l__template_key_name_tl
848       }
849     }
850     {#3}
851 }

```

Equivalent keys need to be “recycled” back around.

```

852 \cs_new_protected:Npn \__template_parse_values_alias:nn #1#2
853 {
854   \seq_remove_all:NV \l__template_value_order_seq \l__template_key_name_tl
855   \prop_get:NVN \l__template_vars_prop \l__template_key_name_tl
856   \l__template_tmp_tl
857   \tl_set_eq:NN \l__template_key_name_tl \l__template_tmp_tl
858   \__template_parse_values_alias_aux:nn {#2} {#1}
859 }
860 \cs_new_protected:Npn \__template_parse_values_alias_aux:nn #1#2

```

```

861 {
862   \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
863   \l__template_keytype_tl
864   \__template_parse_values_elt_auxii:nn {#2} {#1}
865 }
866 \cs_new_protected:Npn \__template_parse_values_meta:nn #1#2
867 {
868   \seq_remove_all:NV \l__template_value_order_seq \l__template_key_name_tl
869   \prop_get:NVN \l__template_vars_prop \l__template_key_name_tl
870   \l__template_tmp_tl
871   \keyval_parse:nnV
872   { \msg_error:nnn { template } { key-no-value } }
873   { \__template_parse_values_meta_elt:nnnn {#1} {#2} }
874   \l__template_tmp_tl
875 }
876 \cs_new_protected:Npn \__template_parse_values_meta_elt:nnnn #1#2#3#4
877 {
878   \tl_set:Nn \l__template_key_name_tl {#3}
879   \cs_set:Npn \__template_tmp:n ##1 {#4}
880   \tl_set:Nn \l__template_tmp_tl { \__template_tmp:n {#2} }
881   \exp_args:NV \__template_parse_values_alias_aux:nn
882   \l__template_tmp_tl {#1}
883 }

```

Setting and adding start quite similar, but diverge later.

```

884 \cs_new_protected:Npn \__template_parse_values_set:n #1
885 {
886   \seq_put_right:NV \l__template_value_order_seq \l__template_key_name_tl
887   \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#1}
888 }
889 \cs_generate_variant:Nn \__template_parse_values_set:n { o , V , v , e }
890 \cs_new_eq:NN \__template_parse_values_set:N \__template_parse_values_set:n
891 \cs_generate_variant:Nn \__template_parse_values_set:N { c }
892 \cs_new_protected:Npn \__template_parse_values_add:n #1
893 {
894   \seq_put_right:NV \l__template_value_order_seq \l__template_key_name_tl
895   \prop_get:NVNF \l__template_values_prop \l__template_key_name_tl
896   \l__template_tmp_tl
897   { \tl_clear:N \l__template_tmp_tl }
898   \use:c { __template_parse_values_add_ \l__template_keytype_tl :n } {#1}
899 }
900 \cs_generate_variant:Nn \__template_parse_values_add:n { o , V , v , e }
901 \cs_new_eq:NN \__template_parse_values_add:N \__template_parse_values_add:n
902 \cs_generate_variant:Nn \__template_parse_values_add:N { c }
903 \clist_map_inline:nn
904 { boolean , choice , function , instance }
905 {
906   \cs_new_protected:cpn { __template_parse_values_add_ #1 :n } ##1
907   { \msg_error:nnn { template } { augmented-key-inappropriate } {#1} }
908 }
909 \clist_map_variable:nNn
910 { integer , length , muskip , real , skip }
911 \l__template_keytype_tl
912 {
913   \cs_new_protected:cpe

```

```

914     { __template_parse_values_add_ \l__template_keytype_tl :n } #1
915     {
916         \tl_set:Ne \exp_not:N \l__template_tmp_tl
917         {
918             \exp_not:c { \__template_map_var_type: _eval:n }
919             { \exp_not:N \l__template_tmp_tl + #1 }
920         }
921         \prop_put:NVV \exp_not:N \l__template_values_prop
922         \exp_not:N \l__template_key_name_tl \exp_not:N \l__template_tmp_tl
923     }
924 }
925 \cs_new_protected:Npn \__template_parse_values_add_commalist:n #1
926 {
927     \clist_put_right:Nn \l__template_tmp_tl {#1}
928     \prop_put:NVV \l__template_values_prop \l__template_key_name_tl \l__template_tmp_tl
929 }
930 \cs_new_protected:Npn \__template_parse_values_add_tokenlist:n #1
931 {
932     \tl_put_right:Nn \l__template_tmp_tl {#1}
933     \prop_put:NVV \l__template_values_prop \l__template_key_name_tl \l__template_tmp_tl
934 }
935 \cs_new_protected:Npn \__template_parse_unknown_error:nn #1#2
936 { \msg_error:nnn { template } { unknown-key } {#1} }
937 \cs_generate_variant:Nn \__template_parse_unknown_error:nn { V }
938 \cs_new_protected:Npn \__template_parse_unknown_store:nn #1#2
939 { \clist_put_right:Nn \UnusedTemplateKeys { #1 = {#2} } }
940 \cs_generate_variant:Nn \__template_parse_unknown_store:nn { V }

```

(End of definition for __template_parse_values_elt:Nnn and others.)

__template_template_set_eq:nnn

To copy a template, each of the lists plus the code has to be copied across. To keep this independent of the list storage system, it is all done with two-part shuffles.

```

941 \cs_new_protected:Npn \__template_template_set_eq:nnn #1#2#3
942 {
943     \__template_recover_defaults:nn {#1} {#3}
944     \__template_store_defaults:nn {#1} {#2}
945     \__template_recover_keytypes:nn {#1} {#3}
946     \__template_store_keytypes:nn {#1} {#2}
947     \__template_recover_vars:nn {#1} {#3}
948     \__template_store_vars:nn {#1} {#2}
949     \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
950     { \msg_info:nnnn { template } { declare-template-code } {#1} {#2} }
951     \cs_gset_eq:cc { \c__template_code_root_tl #1 / #2 }
952     { \c__template_code_root_tl #1 / #3 }
953 }

```

(End of definition for __template_template_set_eq:nnn.)

13.11 Creating instances of templates

__template_declare_instance:nnnn

__template_declare_instance_aux:nnnn

Making an instance has two distinct parts. First, the keys given are parsed to transfer the values into the structured data format used internally. This allows the default and given values to be combined with no repetition. In the second step, the structured data

is converted to pre-defined variable assignments, and these are stored in the function for the instance.

```

954 \cs_new_protected:Npn \__template_declare_instance:nnnn #1#2#3#4
955 {
956   \__template_execute_if_code_exist:nnT {#1} {#2}
957   {
958     \__template_recover_defaults:nn {#1} {#2}
959     \__template_recover_vars:nn {#1} {#2}
960     \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
961   }
962 }
963 \cs_new_protected:Npn \__template_declare_instance_aux:nnnn #1#2#3#4
964 {
965   \bool_set_false:N \l__template_error_bool
966   \__template_parse_values:nnn {#1} {#2} {#4}
967   \bool_if:NF \l__template_error_bool
968   {
969     \prop_put:Nnn \l__template_values_prop { from-template } {#2}
970     \__template_store_values:nn {#1} {#3}
971     \__template_convert_to_assignments:N \l__template_key_order_seq
972     \cs_if_exist:cT { \c__template_instances_root_tl #1 / #3 }
973     { \msg_info:nnnn { template } { declare-instance } {#3} {#1} }
974     \cs_set_protected:cpe { \c__template_instances_root_tl #1 / #3 }
975     {
976       \tl_set:Nn \exp_not:N \l__template_current_type_tl {#1}
977       \tl_set:Nn \exp_not:N \l__template_current_template_tl {#2}
978       \exp_not:N \__template_assignments_push:n
979       { \exp_not:N \l__template_assignments_tl }
980       \exp_not:c { \c__template_code_root_tl #1 / #2 }
981     }
982   }
983 }

```

(End of definition for __template_declare_instance:nnnn and
__template_declare_instance_aux:nnnn.)

__template_instance_set_eq:nnn Copy-paste an instance.

```

984 \cs_new_protected:Npn \__template_instance_set_eq:nnn #1#2#3
985 {
986   \__template_if_instance_exist:nnTF {#1} {#3}
987   {
988     \__template_recover_values:nn {#1} {#3}
989     \__template_store_values:nn {#1} {#2}
990     \cs_if_exist:cT { \c__template_instances_root_tl #1 / #2 }
991     { \msg_info:nnnn { template } { declare-instance } {#2} {#1} }
992     \cs_set_eq:cc { \c__template_instances_root_tl #1 / #2 }
993     { \c__template_instances_root_tl #1 / #3 }
994   }
995   { \msg_error:nnnn { template } { unknown-instance } {#1} {#3} }
996 }

```

(End of definition for __template_instance_set_eq:nnn.)

__template_edit_instance:nnn
__template_edit_instance_aux:nnnn
__template_edit_instance_aux:nVnnn

Editing an instance is almost identical to declaring one. The only variation is the source of the values to use. When editing, they are recovered from the previous instance run.

```

997 \cs_new_protected:Npn \__template_edit_instance:nnn #1#2#3
998 {
999   \__template_if_instance_exist:nnTF {#1} {#2}
1000   {
1001     \__template_recover_values:nn {#1} {#2}
1002     \prop_get:NnN \l__template_values_prop { from-template }
1003     \l__template_tmp_tl
1004     \__template_edit_instance_aux:nVnn
1005     {#1} \l__template_tmp_tl {#2} {#3}
1006   }
1007   { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
1008 }
1009 \cs_new_protected:Npn \__template_edit_instance_aux:nnnn #1#2#3#4
1010 {
1011   \__template_recover_vars:nn {#1} {#2}
1012   \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
1013 }
1014 \cs_generate_variant:Nn \__template_edit_instance_aux:nnnn { nV }

(End of definition for \__template_edit_instance:nnn and \__template_edit_instance_aux:nnnn.)

```

The idea on converting to a set of assignments is to loop over each key, so that the loop order follows the declaration order of the keys. This is done using a sequence as property lists are not “ordered”.

```

1015 \cs_new_protected:Npn \__template_convert_to_assignments:N #1
1016 {
1017   \tl_clear:N \l__template_assignments_tl
1018   \seq_map_function:NN #1
1019   \__template_convert_to_assignments_aux:n
1020 }
1021 \cs_new_protected:Npn \__template_convert_to_assignments_aux:n #1
1022 {
1023   \prop_get:NnN \l__template_keytypes_prop {#1} \l__template_tmp_tl
1024   \__template_convert_to_assignments_aux:nV {#1} \l__template_tmp_tl
1025 }

```

The second auxiliary function actually does the work. The arguments here are the key name (#1) and the keytype (#2). From those, the value to assign and the name of the appropriate variable are recovered. A bit of work is then needed to sort out keytypes with arguments (for example instances), and to look for global assignments. Once that is done, a hand-off can be made to the handler for the relevant keytype.

```

1026 \cs_new_protected:Npn \__template_convert_to_assignments_aux:nn #1#2
1027 {
1028   \prop_pop:NnNT \l__template_values_prop {#1} \l__template_value_tl
1029   {
1030     \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_var_tl
1031     {
1032       \tl_set:Nn \l__template_key_name_tl {#1}
1033       \__template_split_keytype_arg:n {#2}
1034       \bool_lazy_any:nF
1035       {
1036         { \str_if_eq_p:Vn \l__template_keytype_tl { alias } }
1037         { \str_if_eq_p:Vn \l__template_keytype_tl { choice } }
1038         { \str_if_eq_p:Vn \l__template_keytype_tl { meta } }

```



```

1039         }
1040         { \_template_find_global: }
1041         \cs_if_exist_use:CF { \_template_assign_ \l__template_keytype_tl : }
1042         { \_template_assign_variable: }
1043     }
1044     { \msg_error:nnn { template } { unknown-key } {#1} }
1045 }
1046 }
1047 \cs_generate_variant:Nn \_template_convert_to_assignments_aux:nn { nV }

(End of definition for \_template_convert_to_assignments:N,
\_template_convert_to_assignments_aux:n, and \_template_convert_to_assignments_aux:nn.)

```

_template_find_global: Global assignments should have the phrase global at the front. This is pretty easy to find: no other error checking, though.

```

1048 \cs_new_protected:Npn \_template_find_global:
1049 {
1050     \bool_set_false:N \l__template_global_bool
1051     \tl_if_in:onT \l__template_var_tl { global }
1052     {
1053         \exp_after:wN \_template_find_global_aux:w \l__template_var_tl \s__template_stop
1054     }
1055 }
1056 \cs_new_protected:Npn \_template_find_global_aux:w #1 global #2 \s__template_stop
1057 {
1058     \tl_set:Nn \l__template_var_tl {#2}
1059     \bool_set_true:N \l__template_global_bool
1060 }

(End of definition for \_template_find_global: and \_template_find_global_aux:w.)

```

_template_instance_value:nnn For editing templates.

```

1061 \cs_new:Npn \_template_instance_value:nnn #1#2#3
1062 {
1063     \_template_if_instance_exist:nnT {#1} {#2}
1064     { \prop_item:cn { \c__template_values_root_tl #1 / #2 } {#3} }
1065 }

(End of definition for \_template_instance_value:nnn.)

```

13.12 Using templates directly

_template_use_template:nnn Directly use a template with a particular parameter setting. This is also picked up if used in a nested fashion inside a parameter list. The idea is essentially the same as creating an instance, just with no saving of the result.

```

1066 \cs_new_protected:Npn \_template_use_template:nnn #1#2#3
1067 {
1068     \_template_execute_if_code_exist:nnT {#1} {#2}
1069     {
1070         \_template_recover_defaults:nn {#1} {#2}
1071         \_template_recover_vars:nn {#1} {#2}
1072         \_template_parse_values:nnn {#1} {#2} {#3}
1073         \_template_convert_to_assignments:N \l__template_key_order_seq
1074         \_template_debug_typeout:n

```

```

1075         { Using~template~'#2'~of~type~'#1'~directly~ \msg_line_context: }
1076         \tl_set:Nn \l__template_current_type_tl {#1}
1077         \tl_set:Nn \l__template_current_template_tl {#2}
1078         \use:c { \c__template_code_root_tl #1 / #2 }
1079     }
1080 }

```

(End of definition for __template_use_template:nnn.)

13.13 Assigning values to variables

__template_assign_boolean: Setting a Boolean value is slightly different to everything else as the value can be used to work out which `set` function to call. As long as there is no need to recover things from another variable, everything is pretty easy. If there is, then we need to allow for the fact that the recovered value here will *not* be expandable, so needs to be converted to something that is.

```

1081 \cs_new_protected:Npn \__template_assign_boolean:
1082 {
1083     \bool_if:NTF \l__template_global_bool
1084     { \__template_assign_boolean_aux:n { bool_gset } }
1085     { \__template_assign_boolean_aux:n { bool_set } }
1086 }
1087 \cs_new_protected:Npn \__template_assign_boolean_aux:n #1
1088 {
1089     \__template_if_key_value:VTF \l__template_value_tl
1090     {
1091         \__template_key_to_value:
1092         \tl_put_right:Ne \l__template_assignments_tl
1093         {
1094             \exp_not:c { #1 _eq:NN }
1095             \exp_not:V \l__template_var_tl
1096             \exp_not:V \l__template_value_tl
1097         }
1098     }
1099     {
1100         \tl_put_right:Ne \l__template_assignments_tl
1101         {
1102             \exp_not:c { #1 _ \l__template_value_tl :N }
1103             \exp_not:V \l__template_var_tl
1104         }
1105     }
1106 }

```

(End of definition for __template_assign_boolean: and __template_assign_boolean_aux:n.)

__template_assign_choice: The idea here is to find either the choice as-given or else the special `unknown` choice, and to copy the appropriate code across.

```

\__template_assign_choice_aux:nF
\__template_assign_choice_aux:eF
1107 \cs_new_protected:Npn \__template_assign_choice:
1108 {
1109     \__template_assign_choice_aux:eF
1110     { \l__template_key_name_tl \c_space_tl \l__template_value_tl }
1111     {
1112         \__template_assign_choice_aux:eF
1113         { \l__template_key_name_tl \c_space_tl unknown }

```

```

1114         {
1115             \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
1116             \l__template_tmp_tl
1117             \__template_split_keytype_arg:V \l__template_tmp_tl
1118             \msg_error:nnVV { template } { unknown-choice }
1119             \l__template_key_name_tl
1120             \l__template_value_tl
1121         }
1122     }
1123 }
1124 \cs_new_protected:Npn \__template_assign_choice_aux:nF #1
1125 {
1126     \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_tmp_tl
1127     { \tl_put_right:NV \l__template_assignments_tl \l__template_tmp_tl }
1128 }
1129 \cs_generate_variant:Nn \__template_assign_choice_aux:nF { e }

(End of definition for \__template_assign_choice: and \__template_assign_choice_aux:nF.)

```

__template_assign_function: This looks a bit messy but is only actually one function.

```

\__template_assign_function_aux:N
1130 \cs_new_protected:Npn \__template_assign_function:
1131 {
1132     \bool_if:NTF \l__template_global_bool
1133     { \__template_assign_function_aux:N \cs_gset_protected:Npn }
1134     { \__template_assign_function_aux:N \cs_set_protected:Npn }
1135 }
1136 \cs_new_protected:Npn \__template_assign_function_aux:N #1
1137 {
1138     \tl_put_right:Ne \l__template_assignments_tl
1139     {
1140         \cs_generate_from_arg_count:NNnn
1141         \exp_not:V \l__template_var_tl
1142         \exp_not:N #1
1143         { \exp_not:V \l__template_keytype_arg_tl }
1144         { \exp_not:V \l__template_value_tl }
1145     }
1146 }

```

(End of definition for __template_assign_function: and __template_assign_function_aux:N.)

__template_assign_instance: Using an instance means adding the appropriate function creation to the tl. No checks are made at this stage, so if the instance is not valid then errors will arise later.

```

\__template_assign_instance_aux:N
1147 \cs_new_protected:Npn \__template_assign_instance:
1148 {
1149     \bool_if:NTF \l__template_global_bool
1150     { \__template_assign_instance_aux:N \cs_gset_protected:Npn }
1151     { \__template_assign_instance_aux:N \cs_set_protected:Npn }
1152 }
1153 \cs_new_protected:Npn \__template_assign_instance_aux:N #1
1154 {
1155     \tl_put_right:Ne \l__template_assignments_tl
1156     {
1157         \exp_not:N #1 \exp_not:V \l__template_var_tl
1158         {

```

```

1159         \__template_use_instance:nn
1160         { \exp_not:V \l__template_keytype_arg_tl }
1161         { \exp_not:V \l__template_value_tl }
1162     }
1163 }
1164 }

```

(End of definition for __template_assign_instance: and __template_assign_instance_aux:N.)

__template_assign_variable: A general-purpose function for all of the other assignments. As long as the value is not coming from another variable, the stored value is simply transferred for output. We use V-type expansion for the \KeyValue case: for token lists this is essential, whilst for register-based variables, it does no harm and avoids needing a low-level test.

```

1165 \cs_new_protected:Npn \__template_assign_variable:
1166 {
1167     \exp_args:Ne \__template_assign_variable:n
1168     {
1169         \__template_map_var_type:
1170         -
1171         \bool_if:NT \l__template_global_bool { g }
1172         set:N
1173     }
1174 }

```

Notice we need a V-type variant for each (g)set operation here: these need to be provided by expl3.

```

1175 \cs_new_protected:Npn \__template_assign_variable:n #1
1176 {
1177     \__template_if_key_value:VTF \l__template_value_tl
1178     {
1179         \__template_key_to_value:
1180         \tl_put_right:Ne \l__template_assignments_tl
1181         {
1182             \exp_not:c { #1 V } \exp_not:V \l__template_var_tl
1183             \exp_not:V \l__template_value_tl
1184         }
1185     }
1186     {
1187         \tl_put_right:Ne \l__template_assignments_tl
1188         {
1189             \exp_not:c { #1 n } \exp_not:V \l__template_var_tl
1190             { \exp_not:V \l__template_value_tl }
1191         }
1192     }
1193 }

```

(End of definition for __template_assign_variable: and __template_assign_variable:n.)

__template_key_to_value: The idea here is to recover the attribute value of another key. To do that, the marker is removed and a look up takes place. If this is successful, then the name of the variable of the attribute is returned. This assumes that the value will be used in context where it will be converted to a value, for example when setting a number. There is also a need to check in case the copied value happens to be global.

```

1194 \cs_new_protected:Npn \__template_key_to_value:

```

```

1195 { \exp_after:wN \__template_key_to_value_auxi:w \l__template_value_tl }
1196 \cs_new_protected:Npn \__template_key_to_value_auxi:w \KeyValue #1
1197 {
1198   \tl_set:Ne \l__template_tmp_tl { \tl_trim_spaces:e { \tl_to_str:n {#1} } }
1199   \prop_get:NVNTF \l__template_vars_prop \l__template_tmp_tl
1200     \l__template_value_tl
1201   {
1202     \exp_after:wN \__template_key_to_value_auxii:w \l__template_value_tl
1203     \s__template_mark global \q__template_nil \s__template_stop
1204   }
1205   { \msg_error:nnV { template } { unknown-attribute } \l__template_tmp_tl }
1206 }
1207 \cs_new_protected:Npn \__template_key_to_value_auxii:w #1 global #2#3 \s__template_stop
1208 {
1209   \__template_quark_if_nil:NF #2
1210   { \tl_set:Nn \l__template_value_tl {#2} }
1211 }

```

(End of definition for __template_key_to_value:, __template_key_to_value_auxi:w, and __template_key_to_value_auxii:w.)

13.14 Using instances

__template_use_instance:nn Using an instance is just a question of finding the appropriate function. If nothing is found, an error is raised. One complication is that if the first token of argument #2 is \UseTemplate then that is also valid. There is an error-test to make sure that the types agree, and if so the template is used directly.

```

1212 \cs_new_protected:Npn \__template_use_instance:nn #1#2
1213 {
1214   \__template_if_use_template:nTF {#2}
1215   { \__template_use_instance_aux:nNnnn {#1} #2 }
1216   { \__template_use_instance_auxi:nn {#1} {#2} }
1217 }
1218 \cs_new_protected:Npn \__template_use_instance_aux:nNnnn #1#2#3#4#5
1219 {
1220   \str_if_eq:nnTF {#1} {#3}
1221   { \__template_use_template:nnn {#3} {#4} {#5} }
1222   { \msg_error:nnnn { template } { type-mismatch } {#1} {#3} }
1223 }
1224 \cs_new_protected:Npn \__template_use_instance_auxi:nn #1#2
1225 {
1226   \__template_if_instance_exist:nnTF {#1} {#2}
1227   { \__template_use_instance_auxii:nn {#1} {#2} }
1228   { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
1229 }
1230 \cs_new_protected:Npn \__template_use_instance_auxii:nn #1#2
1231 {
1232   \__template_debug_typeout:n
1233   { Using~instance~'#2'~of~type~'#1'~ \msg_line_context: }
1234   \use:c { \c__template_instances_root_tl #1 / #2 }
1235 }

```

(End of definition for __template_use_instance:nn and others.)

13.15 Assignment manipulation

A few functions to transfer assignments about, as this is needed by \AssignTemplateKeys.

```
\__template_assignments_pop: To actually use the assignments.
1236 \cs_new:Npn \__template_assignments_pop: { \l__template_assignments_tl }
(End of definition for \__template_assignments_pop:.)
```

```
\__template_assignments_push:n Here, the assignments are stored for later use.
1237 \cs_new_protected:Npn \__template_assignments_push:n #1
1238 { \tl_set:Nn \l__template_assignments_tl {#1} }
(End of definition for \__template_assignments_push:n.)
```

13.16 Ad hoc key setting

For (mainly) user-overrides at point of use. We aim to keep the key structures “inside” ltttemplates.

```
\__template_keys_set:nnn Setting keys in the document body requires that we work in a similar way to creating
  \__template_keys_set_known:nnn instances. To have keys set in source order, we use a sequence which is set during parsing.
  \__template_keys_set_known_aux:nnn This allows us to reuse internals. Note that the key argument is expanded exactly once,
which allows for users “recycling” keys as \UnusedTemplateKeys directly rather than by value.
```

```
1239 \cs_new_protected:Npn \__template_keys_set:nnn #1#2#3
1240 {
1241   \__template_keys_set_known:nnn {#1} {#2} {#3}
1242   \tl_if_empty:NF \UnusedTemplateKeys
1243   {
1244     \msg_error:nnV { template } { unknown-ad-hoc-keys }
1245     \UnusedTemplateKeys
1246   }
1247 }
1248 \cs_new_protected:Npn \__template_keys_set_known:nnn #1#2#3
1249 { \exp_args:No \__template_keys_set_known_aux:nnn {#3} {#1} {#2} }
```

Key setting is skipped if the input is entirely blank or if it is exactly \NoValue (which could happen if material is passed at a higher level without a branching check). As well as setting native keys, we also check for any in l3keys for compatibility with the previous approach taken.

```
1250 \cs_new_protected:Npn \__template_keys_set_known_aux:nnn #1#2#3
1251 {
1252   \tl_clear:N \UnusedTemplateKeys
1253   \bool_lazy_or:nnF
1254     { \tl_if_blank_p:n {#1} }
1255     { \tl_if_novalue_p:n {#1} }
1256   {
1257     \keys_set_known:nnN { template / #2 / #3 } {#1} \l__template_tmp_tl
1258     \__template_recover_vars:nn {#2} {#3}
1259     \exp_args:NnnV \__template_parse_known_values:nnn {#2} {#3} \l__template_tmp_tl
1260     \__template_convert_to_assignments:N \l__template_value_order_seq
1261     \tl_use:N \l__template_assignments_tl
1262   }
1263 }
```

(End of definition for `__template_keys_set:nnn`, `__template_keys_set_known:nnn`, and `__template_keys_set_known_aux:nnn`.)

13.17 Showing templates and instances

`__template_show_code:nn` Showing the code for a template is just a translation of `\cs_show:c`.

```
1264 \cs_new_protected:Npn __template_show_code:nn #1#2
1265   { \cs_show:c { \c__template_code_root_tl #1 / #2 } }
```

(End of definition for `__template_show_code:nn`.)

`__template_show_defaults:nn` A modified version of the property-list printing code, such that the output refers to templates and instances rather than to the underlying structures.

```
__template_show_keytypes:nn
__template_show_vars:nn
__template_show:Nnnn
1266 \cs_new_protected:Npn __template_show_defaults:nn #1#2
1267   {
1268     __template_if_keys_exist:nnT {#1} {#2}
1269     {
1270       __template_recover_defaults:nn {#1} {#2}
1271       __template_show:Nnnn \l__template_values_prop
1272         {#1} {#2} { default-values }
1273     }
1274   }
1275 \cs_new_protected:Npn __template_show_keytypes:nn #1#2
1276   {
1277     __template_if_keys_exist:nnT {#1} {#2}
1278     {
1279       __template_recover_keytypes:nn {#1} {#2}
1280       __template_show:Nnnn \l__template_keytypes_prop
1281         {#1} {#2} { interface }
1282     }
1283   }
1284 \cs_new_protected:Npn __template_show_vars:nn #1#2
1285   {
1286     __template_execute_if_code_exist:nnT {#1} {#2}
1287     {
1288       __template_recover_vars:nn {#1} {#2}
1289       __template_show:Nnnn \l__template_vars_prop
1290         {#1} {#2} { variable-mapping }
1291     }
1292   }
1293 \cs_new_protected:Npn __template_show:Nnnn #1#2#3#4
1294   {
1295     \msg_show:nneeee { template } { show-attribute }
1296     { \tl_to_str:n {#2} }
1297     { \tl_to_str:n {#3} }
1298     { \tl_to_str:n {#4} }
1299     { \prop_map_function:NN #1 \msg_show_item_unbraced:nn }
1300   }
```

(End of definition for `__template_show_defaults:nn` and others.)

`__template_show_values:nn` Instance values are a little more complex, as is the template to consider.

```
1301 \cs_new_protected:Npn __template_show_values:nn #1#2
1302   {
```

```

1303 \__template_if_instance_exist:nnTF {#1} {#2}
1304 {
1305   \__template_recover_values:nn {#1} {#2}
1306   \msg_show:nneee { template } { show-values }
1307   { \tl_to_str:n {#1} }
1308   { \tl_to_str:n {#2} }
1309   {
1310     \prop_map_function:NN \l__template_values_prop
1311     \msg_show_item_unbraced:nn
1312   }
1313 }
1314 { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
1315 }

```

(End of definition for __template_show_values:nn.)

13.18 Messages

The text for error messages: short and long text for all of them.

```

1316 \msg_new:nnnn { template } { argument-number-mismatch }
1317 { Template-type-~'~#1'~takes-~#2~argument(s). }
1318 {
1319   Templates~of~type~'~#1'~require~#2~argument(s).\\
1320   You~have~tried~to~make~a~template~for~'~#1'~
1321   with~#3~argument(s),~which~is~not~possible:~
1322   the~number~of~arguments~must~agree.
1323 }
1324 \msg_new:nnnn { template } { augmented-key-inappropriate }
1325 { Augmented~key~syntax-~"<key>~+=~<value>"~inappropriate. }
1326 {
1327   Augmented~key~settings~does~not~apply~to~keys~of~the~type~'~#1'.
1328 }
1329 \msg_new:nnnn { template } { bad-number-of-arguments }
1330 { Bad~number~of~arguments~for~template~type~'~#1'. }
1331 {
1332   A~template~may~accept~between~0~and~9~arguments.\\
1333   You~asked~to~use~#2~arguments:~this~is~not~supported.
1334 }
1335 \msg_new:nnnn { template } { bad-variable }
1336 { Incorrect~variable~description~'~#1'. }
1337 {
1338   The~argument~'~#1'~is~not~of~the~form \\
1339   ~~~<variable>\\
1340   ~or~\\
1341   ~~~global~<variable>'\\.\\
1342   It~must~be~given~in~one~of~these~formats~to~be~used~in~a~template.
1343 }
1344 \msg_new:nnnn { template } { choice-not-implemented }
1345 { The~choice~'~#1'~has~no~implementation. }
1346 {
1347   Each~choice~listed~in~the~interface~for~a~template~must~
1348   have~an~implementation.
1349 }

```



```

1350 \msg_new:nnnn { template } { choice-no-code }
1351 { The~choice~'#1'~requires~implementation~details. }
1352 {
1353   When~creating~template~code~using~\DeclareTemplateCode,~
1354   each~choice~name~must~have~an~associated~implementation.\\
1355   This~should~be~given~after~a~'= '~sign:~LaTeX~did~not~find~one.
1356 }
1357 \msg_new:nnnn { template } { choice-requires-code }
1358 { The~choice~'#2'~for~key~'#1'~requires~an~implementation. }
1359 {
1360   You~should~have~put:\\
1361   \ \ #1~:~choice~{~#2 = <code> ~} \\
1362   but~LaTeX~did~not~find~any~<code>.
1363 }
1364 \msg_new:nnnn { template } { deprecated-equiv }
1365 { The~key~equivalent~"#3"~ #2 ~deprecated~on~#1~\msg_line_context:. }
1366 {
1367   The~key~equivalent~"#3"~ #2 ~deprecated,~and~since~#1~should~
1368   not~be~used:~you~will~need~to~update~your~source.
1369 }
1370 \msg_new:nnnn { template } { duplicate-key-interface }
1371 { Key~'#1'~appears~twice~in~interface~definition~\msg_line_context:. }
1372 {
1373   Each~key~can~only~have~one~interface~declared~in~a~template.\\
1374   LaTeX~found~two~interfaces~for~'#1'.
1375 }
1376 \msg_new:nnnn { template } { equiv-key-interface }
1377 {
1378   Key~'#1'~appears~in~interface~definition~and~this~name~therefore~
1379   cannot~be~used~as~an~equivalent~\msg_line_context:.
1380 }
1381 {
1382   Each~key~can~only~have~one~interface~declared~in~a~template.\\
1383   LaTeX~found~two~interfaces~for~'#1':~a~base~one~and~an~equivalent.
1384 }
1385 \msg_new:nnnn { template } { keytype-requires-argument }
1386 { The~key~type~'#1'~requires~an~argument~\msg_line_context:. }
1387 {
1388   You~should~have~put:\\
1389   \ \ <key-name>~:~#1~{~<argument>~} \\
1390   but~LaTeX~did~not~find~an~<argument>.
1391 }
1392 \msg_new:nnnn { template } { key-no-equiv }
1393 { The~key~'#1'~has~no~equivalents~\msg_line_context:. }
1394 {
1395   When~creating~a~compatibility~key~for~a~template~
1396   every~key~listed~must~include~an~equivalent:\\
1397   \ \ <old-key>~:~<type>~==~<new-key(s)>
1398 }
1399 \msg_new:nnnn { template } { key-no-value }
1400 { The~key~'#1'~has~no~value~\msg_line_context:. }
1401 {

```

```

1402     When~creating~an~instance~of~a~template~
1403     every~key~listed~must~include~a~value:\\
1404     \ \ <key>~::~<value>
1405 }
1406 \msg_new:nnnn { template } { key-no-variable }
1407 { The~key~'#1'~requires~implementation~details~\msg_line_context:. }
1408 {
1409     When~creating~template~code~using~\DeclareTemplateCode,~
1410     each~key~name~must~have~an~associated~implementation.\\
1411     This~should~be~given~after~a~'='~sign:~LaTeX~did~not~find~one.
1412 }
1413 \msg_new:nnnn { template } { key-not-implemented }
1414 { Key~'#1'~has~no~implementation~\msg_line_context:. }
1415 {
1416     The~definition~of~key~implementations~for~template~'#2'~
1417     of~template~type~'#3'~does~not~include~any~details~for~key~'#1'.\\
1418     The~key~was~declared~in~the~interface~definition,~
1419     and~so~an~implementation~is~required.
1420 }
1421 \msg_new:nnnn { template } { missing-keytype }
1422 { The~key~'#1'~is~missing~a~key~type~\msg_line_context:. }
1423 {
1424     Key~interface~definitions~should~be~of~the~form\\
1425     \ \ #1~::~<key-type>\\
1426     but~LaTeX~could~not~find~a~<key-type>.
1427 }
1428 \msg_new:nnnn { template } { no-template-code }
1429 {
1430     The~template~'#2'~of~type~'#1'~is~unknown~
1431     or~has~no~implementation.
1432 }
1433 {
1434     There~is~no~code~available~for~the~template~name~given.\\
1435     This~should~be~given~using~\DeclareTemplateCode.
1436 }
1437 \msg_new:nnnn { template } { type-already-defined }
1438 { Template~type~'#1'~already~defined. }
1439 {
1440     You~have~used~\NewTemplateType~
1441     with~a~template~type~that~has~already~been~defined.
1442 }
1443 \msg_new:nnnn { template } { type-mismatch }
1444 { Template~types~'#1'~and~'#2'~do~not~agree. }
1445 {
1446     You~are~trying~to~use~a~template~directly~with~\UseInstance
1447     (or~a~similar~function),~but~the~template~types~do~not~match.
1448 }
1449 \msg_new:nnnn { template } { unknown-ad-hoc-keys }
1450 { Unknown~template~keys~"#1". }
1451 {
1452     LaTeX~was~asked~to~set~'ad-hoc'~template~keys,~but~the~keys~
1453     "#1"~are~not~known~here.
1454 }

```

```

1455 \msg_new:nnnn { template } { unknown-attribute }
1456 { The~template~attribute~'#1'~is~unknown. }
1457 {
1458   There~is~a~definition~in~the~current~template~reading\\
1459   \ \ \token_to_str:N \KeyValue {~#1~} \\
1460   but~there~is~no~key~called~'#1'.
1461 }
1462 \msg_new:nnnn { template } { unknown-choice }
1463 { The~choice~'#2'~was~not~declared~for~key~'#1'. }
1464 {
1465   The~key~'#1'~takes~a~fixed~list~of~choices~
1466   and~this~list~does~not~include~'#2'.
1467 }
1468 \msg_new:nnnn { template } { unknown-default-choice }
1469 { The~default~choice~'#2'~was~not~declared~for~key~'#1'. }
1470 {
1471   The~key~'#1'~takes~a~fixed~list~of~choices~
1472   and~this~list~does~not~include~'#2'.
1473 }

1474 \msg_new:nnnn { template } { unknown-equiv-keytype }
1475 { The~equivalent~key~type~'#1'~is~unknown. }
1476 {
1477   Valid~equivalent~key~types~are:\\
1478   --alias;\\
1479   --meta.\\
1480 }
1481 \msg_new:nnnn { template } { unknown-expansion }
1482 { The~expansion~type~'#1'~is~unknown. }
1483 {
1484   Key~values~can~only~be~expanded~using~one~of~the~pre~defined~methods:~
1485   n,~o,~V,~v,~e,~N~or~c.
1486 }
1487 \msg_new:nnnn { template } { unknown-instance }
1488 { The~instance~'#2'~of~type~'#1'~is~unknown. }
1489 {
1490   You~have~asked~to~use~an~instance~'#2',~
1491   but~this~has~not~been~created.
1492 }
1493 \msg_new:nnnn { template } { unknown-key }
1494 { Unknown~template~key~'#1'. }
1495 {
1496   The~key~'#1'~was~not~declared~in~the~interface~
1497   for~the~current~template.
1498 }
1499 \msg_new:nnnn { template } { unknown-keytype }
1500 { The~key~type~'#1'~is~unknown. }
1501 {
1502   Valid~key~types~are:\\
1503   --boolean;\\
1504   --choice;\\
1505   --commalist;\\
1506   --function;\\
1507   --instance;\\

```

```

1508 --integer;\
1509 --length;\
1510 --muskip;\
1511 --real;\
1512 --skip;\
1513 --tokenlist.
1514 }
1515 \msg_new:nnnn { template } { unknown-type }
1516 { The~template~type~'#1'~is~unknown. }
1517 {
1518   A~template~type~needs~to~be~defined~with~\NewTemplateType
1519   prior~to~using~it.
1520 }
1521 \msg_new:nnnn { template } { unknown-template }
1522 { The~template~'#2'~of~type~'#1'~is~unknown. }
1523 {
1524   No~interface~has~been~declared~for~a~template~
1525   '#2'~of~template~type~'#1'.
1526 }

```

Information messages only have text: more text should not be needed.

```

1527 \msg_new:nnn { template } { declare-instance }
1528 { Declaring~instance~'#1'~of~type~'#2'~\msg_line_context:. }
1529 \msg_new:nnn { template } { declare-template-code }
1530 { Declaring~code~for~template~'#2'~of~template~type~'#1'~\msg_line_context:. }
1531 \msg_new:nnn { template } { declare-template-interface }
1532 {
1533   Declaring~interface~for~template~'#2'~of~template~type~'#1'~
1534   \msg_line_context:.
1535 }
1536 \msg_new:nnn { template } { declare-type }
1537 { Declaring~template~type~'#1'~taking~#2~argument(s)~\msg_line_context:. }
1538 \msg_new:nnn { template } { show-attribute }
1539 {
1540   The~template~'#2'~of~type~'#1'~has~
1541   \tl_if_empty:nTF {#4} { no~#3. } { #3 : #4 }
1542 }
1543 \msg_new:nnn { template } { show-values }
1544 {
1545   The~instance~'#2'~of~type~'#1'~has~
1546   \tl_if_empty:nTF {#3} { no~values. } { values: #3 }
1547 }

```

Also add template to the LaTeX messages.

```

1548 \prop_gput:Nnn \g_msg_module_type_prop { template } { LaTeX }

```

13.19 User functions

All simple translations.

```

\NewTemplateType
\DeclareTemplateInterface
\DeclareTemplateCode
\DeclareTemplateCopy
\EditTemplateDefaults
\UseTemplate
\DeclareInstance
\DeclareInstanceCopy
\EditInstance
\UseInstance
1549 \cs_new_protected:Npn \NewTemplateType #1#2
1550 { \__template_define_type:nn {#1} {#2} }
1551 \cs_new_protected:Npn \DeclareTemplateInterface #1#2#3#4
1552 { \__template_declare_template_keys:nnnn {#1} {#2} {#3} {#4} }
1553 \cs_new_protected:Npn \DeclareTemplateCode #1#2#3#4#5

```

```

1554 { \__template_declare_template_code:nnnnn {#1} {#2} {#3} {#4} {#5} }
1555 \cs_new_protected:Npn \DeclareTemplateCopy #1#2#3
1556 { \__template_template_set_eq:nnn {#1} {#2} {#3} }
1557 \cs_new_protected:Npn \EditTemplateDefaults #1#2#3
1558 { \__template_edit_defaults:nnn {#1} {#2} {#3} }
1559 \cs_new_protected:Npn \UseTemplate #1#2#3
1560 { \__template_use_template:nnn {#1} {#2} {#3} }
1561 \cs_new_protected:Npn \DeclareInstance #1#2#3#4
1562 { \__template_declare_instance:nnnn {#1} {#3} {#2} {#4} }
1563 \cs_new_protected:Npn \DeclareInstanceCopy #1#2#3
1564 { \__template_instance_set_eq:nnn {#1} {#2} {#3} }
1565 \cs_new_protected:Npn \EditInstance #1#2#3
1566 { \__template_edit_instance:nnn {#1} {#2} {#3} }
1567 \cs_new_protected:Npn \UseInstance #1#2
1568 { \__template_use_instance:nn {#1} {#2} }

```

(End of definition for \NewTemplateType and others. These functions are documented on page 370.)

\DeclareTemplateInterfaceEquivalents The optional argument is filtered out directly, with everything else passed on.

```

1569 \NewDocumentCommand \DeclareTemplateInterfaceEquivalents { +m +m o +m }
1570 {
1571   \IfNoValueTF {#3}
1572   { \__template_declare_template_equiv:nnnn {#1} {#2} { } {#4} }
1573   {
1574     \__template_date_compare:nNnTF
1575     {
1576       \int_use:N \c_sys_year_int -
1577       \int_use:N \c_sys_month_int -
1578       \int_use:N \c_sys_day_int
1579     }
1580     >
1581     {#3}
1582     {
1583       \__template_declare_template_equiv:nnnn {#1} {#2}
1584       {
1585         \msg_error:nnnnn { template } { deprecated-equiv }
1586         {#3} { was }
1587       }
1588       {#4}
1589     }
1590     {
1591       \__template_declare_template_equiv:nnnn {#1} {#2}
1592       {
1593         \msg_warning:nnnnn { template } { deprecated-equiv }
1594         {#3} { will~be }
1595       }
1596       {#4}
1597     }
1598   }
1599 }

```

(End of definition for \DeclareTemplateInterfaceEquivalents. This function is documented on page 378.)

\ShowTemplateCode The show functions are again just translation.

\ShowTemplateDefaults

\ShowTemplateInterface

\ShowTemplateVariables

\ShowInstanceValues

File 11: lttemplates.dtx Date: 2026-10-29 Version v1.0o

418

```

1600 \cs_new_protected:Npn \ShowTemplateCode #1#2
1601   { \__template_show_code:nn {#1} {#2} }
1602 \cs_new_protected:Npn \ShowTemplateDefaults #1#2
1603   { \__template_show_defaults:nn {#1} {#2} }
1604 \cs_new_protected:Npn \ShowTemplateInterface #1#2
1605   { \__template_show_keytypes:nn {#1} {#2} }
1606 \cs_new_protected:Npn \ShowTemplateVariables #1#2
1607   { \__template_show_vars:nn {#1} {#2} }
1608 \cs_new_protected:Npn \ShowInstanceValues #1#2
1609   { \__template_show_values:nn {#1} {#2} }

```

(End of definition for `\ShowTemplateCode` and others. These functions are documented on page 379.)

`\IfInstanceExistsT` More direct translation.

```

\IfInstanceExistsF 1610 \cs_new:Npn \IfInstanceExistsTF #1#2
\IfInstanceExistsTF 1611   { \__template_if_instance_exist:nnTF {#1} {#2} }
1612 \cs_new:Npn \IfInstanceExistsT #1#2
1613   { \__template_if_instance_exist:nnT {#1} {#2} }
1614 \cs_new:Npn \IfInstanceExistsF #1#2
1615   { \__template_if_instance_exist:nnF {#1} {#2} }

```

(End of definition for `\IfInstanceExistsT`, `\IfInstanceExistsF`, and `\IfInstanceExistsTF`. These functions are documented on page 376.)

`\KeyValue` Simply dump the argument when executed: this should not happen.

```

1616 \cs_new_protected:Npn \KeyValue #1 {#1}

```

(End of definition for `\KeyValue`. This function is documented on page 372.)

`\InstanceValue`

```

1617 \cs_new:Npn \InstanceValue #1#2#3
1618   { \__template_instance_value:nnn {#1} {#2} {#3} }

```

(End of definition for `\InstanceValue`. This function is documented on page 376.)

`\AssignTemplateKeys` A short call to use a token register by proxy.

```

1619 \cs_new_protected:Npn \AssignTemplateKeys { \__template_assignments_pop: }

```

(End of definition for `\AssignTemplateKeys`. This function is documented on page 372.)

`\SetKnownTemplateKeys` A friendly wrapper, with some speed up for the common case of the third argument being empty.
`\SetTemplateKeys`

```

1620 \cs_new_protected:Npn \SetKnownTemplateKeys #1#2#3
1621   { \__template_keys_set_known:nnn {#1} {#2} {#3} }
1622 \cs_new_protected:Npn \SetTemplateKeys #1#2#3
1623   { \__template_keys_set:nnn {#1} {#2} {#3} }

```

(End of definition for `\SetKnownTemplateKeys` and `\SetTemplateKeys`. These functions are documented on page 373.)

`\SetCurrentKnownTemplateKeys`

```
1624 \cs_new_protected:Npn \SetCurrentKnownTemplateKeys #1
1625 {
1626   \exp_args:NVV \__template_keys_set_known:nnn
1627   \l__template_current_type_tl
1628   \l__template_current_template_tl
1629   {#1}
1630 }
```

(End of definition for \SetCurrentKnownTemplateKeys. This function is documented on page 373.)

`\UnusedTemplateKeys`

```
1631 \tl_new:N \UnusedTemplateKeys
```

(End of definition for \UnusedTemplateKeys. This function is documented on page 373.)

`\DebugTemplatesOn`

`\DebugTemplatesOff`

```
1632 \cs_new_protected:Npn \DebugTemplatesOn { \__template_debug_on: }
1633 \cs_new_protected:Npn \DebugTemplatesOff { \__template_debug_off: }
```

(End of definition for \DebugTemplatesOn and \DebugTemplatesOff. These functions are documented on page 379.)

```
1634 <latexrelease> \IncludeInRelease{0000/00/00}{\tttemplates}%
1635 <latexrelease> {Prototype~document~commands}%
1636 <latexrelease>
1637 <latexrelease> \EndModuleRelease
```

```
1638 \ExplSyntaxOff
```

```
1639 </2ekernel | latexrelease>
```

We need to stop DocStrip treating @@ in a special way at this point.

```
1640 <@@=>
```

File 12

lalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

```
1 <*2ekernel>
```

The following are from plain T_EX:

`\z@` A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent 0pt`.

`\@ne` The number 1.

`\m@ne` The number -1 .

`\tw@` The number 2.

`\sist@@n` The number 16.

`\@m` The number 1000.

`\@MM` The number 20000.

`\@xxxii` The constant 32.

```
2 \chardef\@xxxii=32
```

(End of definition for \@xxxii.)

`\@Mi` Constants 10001–10004.

```
\@Mii 3 \mathchardef\@Mi=10001
```

```
\@Miii 4 \mathchardef\@Mii=10002
```

```
\@Miv 5 \mathchardef\@Miii=10003
```

```
6 \mathchardef\@Miv=10004
```

(End of definition for \@Mi and others.)

`\@tempcnta` Scratch count registers used by L^AT_EX kernel commands.

```
\@tempcntb 7 \newcount\@tempcnta
```

```
8 \newcount\@tempcntb
```

(End of definition for \@tempcnta and \@tempcntb.)

`\if@tempswa` General boolean switch used by L^AT_EX kernel commands.

```
9 \newif\if@tempswa
```

(End of definition for \if@tempswa.)

`\@tempdima` Scratch dimen registers used by L^AT_EX kernel commands.

```
\@tempdimb 10 \newdimen\@tempdima
```

```
\@tempdimc 11 \newdimen\@tempdimb
```

```
12 \newdimen\@tempdimc
```

(End of definition for \@tempdima, \@tempdimb, and \@tempdimc.)

`\@tempboxa` Scratch box register used by L^AT_EX kernel commands.
¹³ `\newbox\@tempboxa`
(End of definition for \@tempboxa.)

`\@tempskipa` Scratch skip registers used by L^AT_EX kernel commands.
`\@tempskipb` ¹⁴ `\newskip\@tempskipa`
¹⁵ `\newskip\@tempskipb`
(End of definition for \@tempskipa and \@tempskipb.)

`\@temptokena` Scratch token register used by L^AT_EX kernel commands.
¹⁶ `\newtoks\@temptokena`
(End of definition for \@temptokena.)

`\@flushglue` Glue used for `\right-` & `\leftskip = 0pt plus 1fil`
¹⁷ `\newskip\@flushglue \@flushglue = 0pt plus 1fil`
(End of definition for \@flushglue.)
¹⁸ `\</2ekernel>`

File 13

ltnctrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1 <*2kernel>
2 \message{control,}
```

```
\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.
```

```
\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.
```

```
\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
    ... ,An .
    Executes BODY n times, with NAME = Ai on the i-th iteration.
    Optimized for the normal case of n = 1. Works for n=0.
```

```
\@tfor NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.
```

NOTES: 1. These macros use no \@temp sequences.
2. These macros do not work if the body contains anything that looks syntactically to TeX like an improperly balanced \if \else \fi.

```
\@whilenum TEST \do {BODY} ==
BEGIN
  if TEST
  then BODY
    \@iwhilenum{TEST \relax BODY}
END
```

```
\@iwhilenum {TEST BODY} ==
BEGIN
  if TEST
  then BODY
```

```

        \@nextwhile = def(\@iwhilenum)
    else \@nextwhile = def(\@whilenoop)
    fi
    \@nextwhile {TEST BODY}
END

\@whilesw SWITCH \fi {BODY} ==
BEGIN
    if SWITCH
    then BODY
        \@iwhilesw {SWITCH BODY}\fi
    fi
END

\@iwhilesw {SWITCH BODY} \fi ==
BEGIN
    if SWITCH
    then BODY
        \@nextwhile = def(\@iwhilesw)
    else \@nextwhile = def(\@whileswnoop)
    fi
    \@nextwhile {SWITCH BODY} \fi
END

```

End of historical L^AT_EX 2.09 comments.

```

\@whilenoop
\@whilenum 3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
\@iwhilenum 4 #2\relax}\fi}
5 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
6 \else\expandafter\@gobble\fi{#1}}

```

(End of definition for \@whilenoop, \@whilenum, and \@iwhilenum.)

```

\@whiledim
\@iwhiledim 7 \long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
8 \long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
9 \else\expandafter\@gobble\fi{#1}}

```

(End of definition for \@whiledim and \@iwhiledim.)

```

\@whileswnoop
\@whilesw 10 \long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\@iwhilesw 11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12 \else\@gobbletwo\fi{#1}\fi}

```

(End of definition for \@whileswnoop, \@whilesw, and \@iwhilesw.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@for NAME := LIST \do {BODY} ==
  BEGIN \@forloop expand(LIST),\@nil,\@nil \@@ NAME {BODY} END
```

```
\@forloop CAR, CARCDR, CDRCDR \@@ NAME {BODY} ==
  BEGIN
    NAME = CAR
    if def(NAME) = def(\@nnil)
      else BODY;
      NAME = CARCDR
      if def(NAME) = def(\@nnil)
        else BODY
          \@iforloop CDRCDR \@@ NAME \do {BODY}
        fi
      fi
  END
```

```
\@iforloop CAR, CDR \@@ NAME {BODY} =
  NAME = CAR
  if def(NAME) = def(\@nnil)
    then \@nextwhile = def(\@fornoop)
    else BODY ;
      \@nextwhile = def(\@iforloop)
    fi
  \@nextwhile name cdr {body}
```

```
\@tfor NAME := LIST \do {BODY}
  = \@tforloop LIST \@nil \@@ NAME {BODY}
```

```
\@tforloop car cdr \@@ name {body} =
  name = car
  if def(name) = def(\@nnil)
    then \@nextwhile == \@fornoop
    else body ;
      \@nextwhile == \@forloop
    fi
  \@nextwhile name cdr {body}
```

End of historical L^AT_EX 2.09 comments.

\@nnil

```
13 \def\@nnil{\@nil}
```

(End of definition for \@nnil.)

\@empty

```
14 \def\@empty{}
```

(End of definition for \@empty.)

```

\@fornoop
15 \long\def\@fornoop#1\@@#2#3{}
(End of definition for \@fornoop.)

\@for
16 \long\def\@for#1:=#2\do#3{%
17   \expandafter\def\expandafter\@fortmp\expandafter{#2}%
18   \ifx\@fortmp\@empty \else
19     \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
(End of definition for \@for.)

\@forloop
20 \long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
21   #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
(End of definition for \@forloop.)

\@iforloop
22 \long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
23   \expandafter\@fornoop \else
24   #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
(End of definition for \@iforloop.)

\@tfor
25 \def\@tfor#1:={\@tfor#1 }
26 \long\def\@tfor#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
27   \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
28 \long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
29   \expandafter\@fornoop \else
30   #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
(End of definition for \@tfor.)

\@break@tfor Break out of a \@tfor loop. This should be called inside the scope of an \if. See
\@iffilenamepath for an example.
31 \long\def\@break@tfor#1\@@#2#3{\fi\fi}
(End of definition for \@break@tfor.)

\@removeelement Removes an element from a comma-separated list and puts it into a control sequence,
called as \@removeelement{<element>}{<list>}{<cs>}. Due to the implementation
method the <element> is not allowed to contain braces.
32 \def\@removeelement#1#2#3{%
33   \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
34   \def\reserved@b##1,\reserved@b##2\reserved@b{%
35     \ifx,##1\@empty\else##1\fi}%
36   \edef#3{%
37     \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
(End of definition for \@removeelement.)
38 </2ekernel>

```

File 14

lterror.dtx

1 Error handling and tracing

This section defines L^AT_EX's error commands.

```
1 <*2kernel>
```

The ‘2kernel’ code ensures that a `\usepackage{autoerr}` is essentially ignored if a ‘full’ format is being used that has the error messages already in the format.

These days we don't support autoloading approach any longer, but this part bit is kept in case it is used in old documents.

```
2 \expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

1.1 General commands

\MessageBreak This command prints a new-line inside a message, followed by a continuation line begun with `\@msg@continuation`. Normally it is defined to be `\relax`, but inside messages, it is let to `\@message@break`.

```
3 \let\MessageBreak\relax
```

(End of definition for \MessageBreak.)

\GenericInfo This takes two arguments: a continuation and a message, and sends the result to the log file.

```
4 \DeclareRobustCommand{\GenericInfo}[2]{%
5   \begingroup
6     \def\MessageBreak{^^J#1}%
7     \set@display@protect
8     \immediate\write\m@ne{#2\on@line.}%
9   \endgroup
10 }
```

(End of definition for \GenericInfo.)

\GenericWarning This takes two arguments: a continuation and a message, and sends the result to the screen.

```
11 \DeclareRobustCommand{\GenericWarning}[2]{%
12   \begingroup
13     \def\MessageBreak{^^J#1}%
14     \set@display@protect
15     \immediate\write\@unused{^^J#2\on@line.^^J}%
16   \endgroup
17 }
```

(End of definition for \GenericWarning.)

`\GenericError` This macro takes four arguments: a continuation, an error message, where to go for further information, and the help information. It displays the error message, and sets the error help (the result of typing `h` to the prompt), and does a horrible hack to turn the last context line (which by default is the only context line) into just three dots. This could be made more efficient.

```

18 \bgroup
19 \lccode'\@=' \ %
20 \lccode'\~=' \ %
21 \lccode'\}=' \ %
22 \lccode'\{=' \ %
23 \lccode'\T=' \T%
24 \lccode'\H=' \H%
25 \catcode'\ =11\relax%
26 \lowercase{%
27 \egroup%
```

Unfortunately $\TeX$ versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old $\TeX$'s we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

.

To appear on the terminal, but if you do not like it, you can always upgrade your $\TeX$! In order for your format to use this version, you must define the macro `\TeXversion` to be the version number, e.g., 3.14 of the underlying $\TeX$. See the comments in `ltdircheck.dtx`.

```

28 \dimen@ifx\TeXversion\undefined4\else\TeXversion\fi\p@%
29 \ifdim\dimen@>3.14\p@%
    First the 'standard case'.
30 \DeclareRobustCommand{\GenericError}[4]{%
31 \begingroup%
32 \immediate\write\@unused{}%
33 \def\MessageBreak{^^J}%
34 \set@display@protect%
35 \edef%
36 % %<-----do not delete this space!----->%
37 \@err@
38 {{#4}}%
39 \errhelp
40 % %<-----do not delete this space!----->%
41 \@err@
42 \let
43 % %<-----do not delete this space!----->%
44 \@err@
45 \@empty
46 \def\MessageBreak{^^J#1}%
47 \def~{\errmessage{%
48 #2.^^J^^J%
49 #3^^J%
50 Type H <return> for immediate help%
51 % %<-----do not delete this space!----->%
52 \@err@
```

```

53 }}%
54 ~%
55 \endgroup}%
56 \else%
    Secondly the version for old TEX's.
57 \DeclareRobustCommand{\GenericError}[4]{%
58 \begingroup%
59 \immediate\write\@unused{}%
60 \def\MessageBreak{^^J}%
61 \set@display@protect%
62 \edef%
63 %    %<-----do not delete this space!----->%
64 \@err@
65 {{#4}}%
66 \errhelp
67 %    %<-----do not delete this space!----->%
68 \@err@
69 \let
70 %    %<-----do not delete this space!----->%
71 \@err@
72 \errmessage
73 \def\MessageBreak{^^J#1}%
74 \def~{\typeout{! %
75 #2.^^J^^J%
76 #3^^J%
77 Type H <return> for immediate help.}%
78 %    %<-----do not delete this space!----->%
79 \@err@
80 }}%
81 ~%
82 \endgroup}%
83 \fi}%

```

(End of definition for \GenericError.)

<pre> \PackageError \PackageWarning \PackageWarningNoLine \PackageInfo \ClassError \ClassWarning \ClassWarningNoLine \ClassInfo </pre>	<p>These commands are intended for use by package and class writers, to give information to authors. The syntax is:</p> <pre> \PackageError{<package>}{<error>}{<help>} \PackageWarning{<package>}{<warning>} \PackageWarningNoLine{<package>}{<warning>} \PackageInfo{<package>}{<info>} </pre>
--	--

and similarly for classes. The **Error** commands print the `<error>` message, and present the interactive prompt; if the author types `h`, then the `<help>` information is displayed. The **Warning** commands produce a warning but do not present the interactive prompt. The **WarningNoLine** commands do the same, but don't print the input line number. The **Info** commands write the message to the `log` file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:


```

\newcommand{\foo}{F00}
\PackageWarning{ethel}{%
  Your hovercraft is full of eels,\MessageBreak
  and \protect\foo\space is \foo}

```

produces:

```

Package ethel warning: Your hovercraft is full of eels,
(ethel)                and \foo is F00 on input line 54.

```

```

84 \gdef\PackageError#1#2#3{%
85   \GenericError{%
86     (#1)\@spaces\@spaces\@spaces\@spaces
87   }{%
88     Package #1 Error: #2%
89   }{%
90     See the #1 package documentation for explanation.%
91   }{#3}%
92 }
93 \def\PackageWarning#1#2{%
94   \GenericWarning{%
95     (#1)\@spaces\@spaces\@spaces\@spaces
96   }{%
97     Package #1 Warning: #2%
98   }%
99 }
100 \def\PackageWarningNoLine#1#2{%
101   \PackageWarning{#1}{#2\@gobble}%
102 }
103 \def\PackageInfo#1#2{%
104   \GenericInfo{%
105     (#1) \@spaces\@spaces\@spaces
106   }{%
107     Package #1 Info: #2%
108   }%
109 }
110 \gdef\ClassError#1#2#3{%
111   \GenericError{%
112     (#1) \space\@spaces\@spaces\@spaces
113   }{%
114     Class #1 Error: #2%
115   }{%
116     See the #1 class documentation for explanation.%
117   }{#3}%
118 }
119 \def\ClassWarning#1#2{%
120   \GenericWarning{%
121     (#1) \space\@spaces\@spaces\@spaces
122   }{%
123     Class #1 Warning: #2%
124   }%
125 }
126 \def\ClassWarningNoLine#1#2{%

```

```

127 \ClassWarning{#1}{#2@gobble}%
128 }
129 \def\ClassInfo#1#2{%
130 \GenericInfo{%
131 (#1) \space\space\@spaces\@spaces
132 }{%
133 Class #1 Info: #2%
134 }%
135 }

```

(End of definition for \PackageError and others.)

```

\ClassNote
\ClassNoteNoLine 136 </2ekernel>
\PackageNote      137 <*2ekernel | latexrelease>
\PackageNoteNoLine 138 <latexrelease> \IncludeInRelease{2021/11/15}%
139 <latexrelease> { \ClassNote } { Notes for classes/packages }%
140 \def\ClassNote#1#2{%
141 \GenericWarning{%
142 (#1) \space\space\@spaces\@spaces
143 }{%
144 Class #1 Info: #2%
145 }%
146 }
147 \def\ClassNoteNoLine#1#2{ \ClassNote{#1}{#2@gobble} }
148 \def\PackageNote#1#2{%
149 \GenericWarning{%
150 (#1) \@spaces\@spaces\@spaces
151 }{%
152 Package #1 Info: #2%
153 }%
154 }
155 \def\PackageNoteNoLine#1#2{ \PackageNote{#1}{#2@gobble} }
156 </2ekernel | latexrelease>
157 <latexrelease> \EndIncludeInRelease

```

We don't roll back, because if this code is used by packages then most often they will not have rollback code implemented, so they would immediately break even if they otherwise would work fine.

```

158 <latexrelease> \IncludeInRelease{0000/00/00}%
159 <latexrelease> { \ClassNote } { Notes for classes/packages }%
160 <latexrelease>
161 <latexrelease> \EndIncludeInRelease
162 <*2ekernel>

```

(End of definition for \ClassNote and others.)

```

\@latex@error Errors and other info, for use in the LATEX core.
\@latex@warning 163 \gdef\@latex@error#1#2{%
\@latex@warning@no@line 164 \GenericError{%
\@latex@info 165 \space\space\space\@spaces\@spaces\@spaces
\@latex@info@no@line 166 }{%
167 LaTeX Error: #1%
168 }{%

```

```

169     See the LaTeX manual or LaTeX Companion for explanation.%
170   }{#2}%
171 }
172 \def\@latex@warning#1{%
173   \GenericWarning{%
174     \space\space\space\@spaces\@spaces\@spaces
175   }{%
176     LaTeX Warning: #1%
177   }%
178 }
179 \def\@latex@warning@no@line#1{%
180   \@latex@warning{#1\@gobble}}
181 \def\@latex@info#1{%
182   \GenericInfo{%
183     \@spaces\@spaces\@spaces
184   }{%
185     LaTeX Info: #1%
186   }%
187 }
188 \def\@latex@info@no@line#1{%
189   \@latex@info{#1\@gobble}}

```

\@font@warning and \@font@info are defined later since they have to be redefined by the tracefnt package.

```

def\@font@warning#1{%
  \GenericWarning{%
    {(font)\@spaces\@spaces}%
    {Font Warning: #1}%
  }
def\@font@info#1{%
  \GenericInfo{%
    (font)\space\@spaces
  }{%
    Font Info: #1%
  }%
}

```

(End of definition for \@latex@error and others.)

```

\@latex@note These are “info” messages that display on the terminal not just in the transcript.
\@latex@note@no@line
190 </2ekernel>
191 <*2ekernel | latexrelease>
192 <latexrelease> \IncludeInRelease{2021/11/15}%
193 <latexrelease> {\@latex@note}{Display notes}%
194 \def\@latex@note#1{%
195   \GenericWarning{%
196     \@spaces\@spaces\@spaces
197   }{%
198     LaTeX Info: #1%
199   }%
200 }

```

```

201 \def\@latex@note@no@line#1{%
202   \@latex@note{#1@gobble}}

```

We don't make them undefined but rather point to `\@latex@info` because that's what they replace. This way we can change `\@latex@info` elsewhere without the need to further rollback sections.

```

203 </2ekernel | latexrelease>
204 <latexrelease>\EndIncludeInRelease
205 <latexrelease>\IncludeInRelease{0000/00/00}%
206 <latexrelease>          {\@latex@note}{Display notes}%
207 <latexrelease>
208 <latexrelease>\let\@latex@note\@latex@info
209 <latexrelease>\let\@latex@note@no@line\@latex@info@no@line
210 <latexrelease>\EndIncludeInRelease
211 <*2ekernel>

```

(End of definition for \@latex@note and \@latex@note@no@line.)

\c@errorcontextlines `\errorcontextlines` as a L^AT_EX counter, so that it may be manipulated with `\setcounter` (once it is defined :-)

```

212 \let\c@errorcontextlines\errorcontextlines
213 \c@errorcontextlines=-1

```

(End of definition for \c@errorcontextlines.)

\on@line The message ‘on input line *n*’.

```

214 \def\on@line{ on input line \the\inputlineno}

```

(End of definition for \on@line.)

\@warning Older L^AT_EX messages. For the moment, these `\let` to the new message commands. They may be changed later, once only obsolete packages and classes contain them.

```

\@warning
\@warning
\@latexerr
215 \let\@warning\@latex@warning
216 \let\@@warning\@latex@warning@no@line
217 \global\let\@latexerr\@latex@error

```

(End of definition for \@warning, \@@warning, and \@latexerr.)

\@spaces Four spaces.

```

218 \def\@spaces{\space\space\space\space}

```

(End of definition for \@spaces.)

1.2 Specific errors

\@eha The more common error help messages.

```

\@ehb
219 \gdef\@eha{%
\@ehc
220   Your command was ignored.\MessageBreak
\@ehd
221   Type \space I <command> <return> \space to replace it %
222   with another command,\MessageBreak
223   or \space <return> \space to continue without it.}
224 \gdef\@ehb{%
225   You've lost some text. \space \@ehc}
226 \gdef\@ehc{%
227   Try typing \space <return> %

```

```

228 \space to proceed.\MessageBreak
229 If that doesn't work, type \space X <return> \space to quit.}
230 \gdef\@ehd{%
231 You're in trouble here. \space\@ehc}

(End of definition for \@eha and others.)

\@notdefinable Error message generated in \@ifdefinable from calls to one of the commands \newcommand,
\newlength or \newtheorem specifying an already-defined command name or one that
begins \end....
232 \gdef\@notdefinable{%
233 \@latex@error{%
234 Command \@backslashchar\reserved@a\space
235 already defined.\MessageBreak
236 Or name \@backslashchar\@qend... illegal,
237 see p.192 of the manual}\@eha}

(End of definition for \@notdefinable.)

\@nolnerr Generated by \newline and \\ when called in vertical mode.
238 \gdef\@nolnerr{%
239 \@latex@error{There's no line here to end}\@eha}

(End of definition for \@nolnerr.)

\@nocounterr Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined
counter <cnt>.

\@nocnterr Obsolete error message generated in LATEX2.09 by \setcounter, \addtocounter or
\newcounter for undefined counter. DO NOT use for LATEX 2ε it MIGHT vanish! Use
\@nocounterr{<cnt>} instead.
240 \gdef\@nocounterr#1{%
241 \@latex@error{No counter '#1' defined}\@eha}
242 \gdef\@nocnterr{\@nocounterr?}

(End of definition for \@nocounterr and \@nocnterr.)

\@ctrerr Called when trying to print the value of a counter numbered by letters that's greater
than 26.
243 \gdef\@ctrerr{%
244 \@latex@error{Counter too large}\@ehb}

(End of definition for \@ctrerr.)

\@nodocument Error produced if paragraphs are typeset in the preamble.
245 \gdef\@nodocument{%
246 \@latex@error{Missing \protect\begin{document}}\@ehd}

(End of definition for \@nodocument.)

```

`\@badend` Called by `\end` that doesn't match its `\begin`. RmS 1992/08/24: added code to `\@badend` to display position of non-matching `\begin`. FMi 1993/01/14: missing space added.

The environment name has to literally match, i.e., what is stored in `\@currenvir` (after one expansion) must match what is passed to `\end` (without expansion). If not we complain. Not the absolute best solution but at least it avoids getting `\begin{foo}` ended by `\end{foo}` which was possible in the past.

```
247 \gdef\@badend#1{%
248   \latex@error{\protect\begin
249     {\detokenize\expandafter{\@currenvir}}\@currenvline
250     \space ended by \protect\end{\detokenize{#1}}}\@eha}
```

(End of definition for \@badend.)

`\@badmath` Called by `\[, \]`, `\(` or `\)` when used in wrong mode.

```
251 \gdef\@badmath{%
252   \latex@error{Bad math environment delimiter}\@eha}
```

(End of definition for \@badmath.)

`\@toodeep` Called by a list environment nested more than six levels deep, or an `enumerate` or `itemize` nested more than four levels.

```
253 \gdef\@toodeep{%
254   \latex@error{Too deeply nested}\@ehd}
```

(End of definition for \@toodeep.)

`\@badpoptabs` Called by `\endtabbing` when not enough `\poptabs` have occurred, or by `\poptabs` when too many have occurred.

```
255 \gdef\@badpoptabs{%
256   \latex@error{\protect\pushtabs\space and \protect\poptabs
257     \space don't match}\@ehd}
```

(End of definition for \@badpoptabs.)

`\@badtab` Called by `\>`, `\+`, `\-` or `\<` when stepping to an undefined tab.

```
258 \gdef\@badtab{%
259   \latex@error{Undefined tab position}\@ehd}
```

(End of definition for \@badtab.)

`\@preamerr` This error is special: it appears in places where we normally have to `\protect` expansions. However, to prevent a protection of the error message itself (which would result in the message getting printed not issued on the terminal) we need to locally reset `\protect` to `\relax`.

```
260 \gdef\@preamerr#1{%
261   \begingroup
262     \let\protect\relax
263     \latex@error{\ifcase #1 Illegal character\or
264       Missing @-exp\or Missing p-arg\fi\space
265       in array arg}\@ehd
266   \endgroup}
```

(End of definition for \@preamerr.)

`\@badlinearg` Occurs in `\line` and `\vector` command when a bad slope argument is encountered.

```

267 \gdef\@badlinearg{%
268   \latexerror{%
269     Bad \protect\line\space or \protect\vector
270     \space argument}\@ehb}

(End of definition for \@badlinearg.)

```

`\@LRmoderr` A command is not allowed in restricted horizontal mode, i.e., in LR-mode in L^AT_EX terminology.

```

271 \gdef\@LRmoderr{%
272   \latexerror{Not allowed in LR mode}\@ehb}

(End of definition for \@LRmoderr.)

```

`\@parmoderr` Occurs in a float environment or a `\marginpar` when encountered in inner vertical mode.

```

273 \gdef\@parmoderr{%
274   \latexerror{Not in outer par mode}\@ehb}

(End of definition for \@parmoderr.)

```

`\@fltovf` Occurs in float environment or `\marginpar` when there are no more free boxes for storing floats.

```

275 \gdef\@fltovf{%
276   \latexerror{Too many unprocessed floats}\@ehb}

(End of definition for \@fltovf.)

```

`\@latexbug` Occurs in output routine. This is bad news.

```

277 \gdef\@latexbug{%
278   \latexerror{This may be a LaTeX bug}{Call for help}}

(End of definition for \@latexbug.)

```

`\@badcrerr` This error was removed and replaced by `\@nolnerr`.

```

279 %\def\@badcrerr {\latexerror{Bad use of \protect\\}\@ehc}

(End of definition for \@badcrerr.)

```

`\@noitemerr` `\addvspace` or `\addpenalty` was called when not in vmode. Probably caused by a missing `\item`.

```

280 \gdef\@noitemerr{%
281   \latexerror{Something's wrong--perhaps a missing %
282     \protect\item}\@ehc}

(End of definition for \@noitemerr.)

```

`\@notprerr` A command that can be used only in the preamble appears after the command `\begin{document}`.

```

283 \gdef\@notprerr{%
284   \latexerror{Can be used only in preamble}\@eha}

(End of definition for \@notprerr.)

```

`\@inmatherr` Issued by commands that don't work correctly within math (like `\item`). There is no real error recovery happening, e.g., the user might get additional errors afterwards.

```

285 \gdef\@inmatherr#1{%
286   \relax
287   \ifmmode
288     \@latex@error{Command \protect#1 invalid in math mode}\@ehc
289   \fi}

```

(End of definition for \@inmatherr.)

`\@invalidchar` An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

```

290 %\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}

```

(End of definition for \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in L^AT_EX2.09 include:

Environment --- undefined

Issued by `\begin` for undefined environment.

Tab overflow

Occurs in `\=` when maximum number of tabs exceeded.

< in mid line

Occurs in `<` when it appears in middle of line.

Float(s) lost

In output routine, caused by a float environment or `\marginpar` occurring in inner vertical mode.

1.3 Tracing

The `trace` package implements the commands `\traceon` and `\traceoff` that work similar to `\tracingall` but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros `\conditionally@traceoff` and `\contionally@traceon`.

For the kernel code the `trace` package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to `\traceon` we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between `trace` being loaded or not.

`\conditionally@traceon` These are only dummy definitions. For details see the `trace` package.

```

\conditionally@traceoff
291 \let\conditionally@traceon\@empty
292 \let\conditionally@traceoff\@empty

```

(End of definition for \conditionally@traceon and \conditionally@traceoff.)

```

293 \</2kernel>

```


File 15

ltpar.dtx

1 Paragraphs

This section of the kernel declares the commands used to set `\par` and `\everypar` when ever their function needs to be changed for a long time.

This file here describes the interfaces that have been in the kernel forever, used to implement the scenarios described below. They remain valid but are now augmented in the next file (`ltpara.dtx`) to add hooks to paragraphs. At some point we will consolidate the two files further.

There are two situations in which `\par` may be changed:

- Long-term changes, in which the new value is to remain in effect until the current environment is left. The environments that change `\par` in this way are the following:
 - All list environments (itemize, quote, etc.)
 - Environments that turn `\par` into a noop: tabbing, array and tabular.
- Temporary changes, in which `\par` is restored to its previous value the next time it is executed. The following are all such uses.
 - `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
 - The mechanism for avoiding page breaks and getting the spacing right after section heads.

1.1 Implementation

`\@setpar` To permit the proper interaction of these two situations, long-term changes are made by the `\@setpar{<VAL>}` command. Its function is:

To set `\par`. It `\def`'s `\par` and `\@par` to `<VAL>`.

`\@restorepar` Short-term changes are made by the usual `\def\par` commands. The original values are restored after a short-term change by the `\@restorepar` commands.

`\@@par` `\@@par` always is defined to be the original TeX `\par`.

`\everypar` `\everypar` is changed only for the short term. Whenever `\everypar` is set non-null, it should restore itself to null when executed.

The following commands change `\everypar` in this way:

- `\item`
- `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
- `\minipage`

When dealing with `\par` and `\everypar` remember the following two warnings:

1. Commands that make short-term changes to `\par` and `\everypar` must take account of the possibility that the new commands and the ones that do the restoration may be executed inside a group. In particular, `\everypar` is executed inside a group whenever a new paragraph begins with a left brace. The `\everypar` command that restores its definition should be local to the current group (in case the command is inside a minipage used inside someplace where `\everypar` has been redefined). Thus, if `\everypar` is redefined to do an `\everypar{}` it could take several executions of `\everypar` before the restoration “holds”. This usually causes no problem. However, to prevent the extra executions from doing harm, use a global switch to keep anything harmful in the new `\everypar` from being done twice.
2. Commands that change `\everypar` should remember that `\everypar` might be supposed to set the following switches false:
 - `@nobreak`
 - `@minipage`

they should do the setting if necessary.

```

1 <*2kernel>
2 \message{par,}

```

`\@setpar` Initiate a long-term change to `\par`.

```

\@par 3 \def\@setpar#1{\def\par{#1}\def\@par{#1}}

```

The default definition of `\@par` will ensure that if `\@restorepar` defines `\par` to execute `\@par` it will redefine itself to the primitive `\@@par` after one iteration.

```

4 \def\@par{\let\par\@@par\par}

```

(End of definition for \@setpar and \@par.)

`\@restorepar` Restore from a short-term change to `\par`.

```

5 \def\@restorepar{\def\par{\@par}}
6 </2kernel>

```

(End of definition for \@restorepar.)

File 16

ltpara.dtx

Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

1 Introduction

The building of paragraphs in the T_EX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the T_EXbook [?, chap. 14] (for the full truth you may even have to study the program code).

1.1 The default processing done by the engine

T_EX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, T_EX will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.²⁴

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added²⁵. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, T_EX offers the possibility to put some special code into `\everypar` to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

```
\everypar{\setbox\z@\lastbox}\everypar{} ...}
```

This removes the paragraph indentation box again (that was already placed by T_EX), then resets `\everypar` so that it doesn’t do anything on the next paragraph start and then does whatever it wants to do, e.g., in an `\item` of a list it will typeset the label in front of the paragraph text. However, there is only one such `\everypar` token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while T_EX is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates as an error depending on the mode T_EX is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

²⁴Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

²⁵That’s a bit different from placing a zero-sized box!

If this primitive ends the paragraph it does some special “end of horizontal list” processing, then calls $\text{\TeX}$ ’s paragraph builder; this breaks the horizontal list into lines and then these lines are added as boxes to the enclosing vertical list and $\text{\TeX}$ returns to vertical mode.

This $\text{\par}$ command can be given explicitly, but there are also situations in which $\text{\TeX}$ is generating it on the fly. Most often this happens when $\text{\TeX}$ encounters a blank line which is automatically changed to a $\text{\par}$ command which is then executed. The other possibility is that $\text{\TeX}$ encounters a command which is incompatible with horizontal processing, e.g., $\text{\vskip}$ (a request for adding vertical space). In such cases it silently backs up, and inserts a $\text{\par}$ in the hope that this gets it out of horizontal mode and makes the vertical command acceptable.

The important point to note here is that $\text{\TeX}$ really inserts the command with the name $\text{\par}$, which can be redefined. Thus, it may not have its original “primitive” meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a $\text{\TeX}$ document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at A, redefines $\text{\par}$, then sees $\text{\vskip}$ and inserts $\text{\par}$ to end the paragraph. But this now only runs $\text{\relax}$ so nothing changes and $\text{\vskip}$ is read again, issues a $\text{\par}$ which In short, it only takes a plain $\text{\TeX}$ document with five tokens to run forever (since no memory is consumed and therefore eventually exhausted).

There is no way other than changing $\text{\par}$ to gain control at the end of a paragraph, i.e., there is no token list like $\text{\everypar}$ that is inserted. Hence the only way to change the default behavior is to modify the action that $\text{\par}$ executes, with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the $\text{\par}$ in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where $\text{\TeX}$ handles the situation differently (most likely for speed reasons back when computers were much slower). If $\text{\TeX}$ finds itself in unrestricted horizontal mode at the end of building a vertical box (for an $\text{\insert}$, $\text{\vadjust}$ or executing the output routine code), it will finish the horizontal list not by issuing a $\text{\par}$ command (which would be consistent with all other places) but by simply executing the primitive meaning of $\text{\par}$, regardless of the actual definition that $\text{\par}$ has at the time.

Thus, if you have carefully crafted a redefined $\text{\par}$ to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph ... text.}
```

you will find that your code does not get run for the last paragraph in that box. $\text{\LaTeX}$ avoids this problem, by making sure that its boxes (such as $\text{\parbox}$ or the $\text{\minipage}$ environment, etc.) all internally add an explicit $\text{\par}$ at the end so that such code is run and $\text{\TeX}$ finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the $\text{\LaTeX}$ kernel; if some package uses low-level $\text{\vboxes}$ without adding this precaution the $\text{\TeX}$ optimization kicks in and no special $\text{\par}$ code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., $\text{\[...]}$ in $\text{\LaTeX}$ or $\text{\$ \$... \$ \$}$ in plain $\text{\TeX}$, then $\text{\TeX}$ will resume horizontal mode afterward, i.e., it will start to build a new horizontal list

without inserting an indentation box or `\everypar` at that point. However, if that list immediately ends with an explicit or implicit `\par` then `TEX` will simply throw away this “null” paragraph and not do its usual “end of horizontal list” processing, so this special case also needs to be accounted for when introducing any extended processing.

2 The new mechanism implemented for `LATEX`

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in such a way that these can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by `TEX`, e.g., we install special code inside `\everypar` to provide hooks at the beginning and we redefine `\par` to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of `\everypar` is not overwriting our code. This is done through a trick: we basically hide the real `\everypar` from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new `\everypar` at the right place inside the process so that it looks as if it was the primitive `\everypar`.²⁶

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that `TEX` inserts the token `\par` (that doesn’t have a defined meaning) we can’t hide “the real thingTM” and offer the package an indistinguishable alternate.

Fortunately, `LATEX` has already redefined `\par` for its own purposes. As a result there aren’t many packages that attempt to change `\par`, because without a lot of extra care that would fail miserably. But the bottom line is that, if you load a package that alters `\par` then the end of paragraph hooks are most likely not executing while that redefinition is active.²⁷

²⁶Ideally, `\everypar` wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in `\everypar`), this will not happen overnight. For that reason support for this legacy method is retained.

²⁷Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

2.1 The provided hooks

<hr/>	The following four public hooks are defined and executed for each paragraph:
para/before	
para/begin	
para/end	para/before This hook is executed after the kernel hook <code>\@kernel@before@para@before</code> (discussed below) in vertical mode immediately after $\text{\TeX}$ has contributed <code>\parskip</code> to the vertical list and before the actual paragraph processing in horizontal mode starts.
para/after	This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion. ²⁸
	para/begin This hook is executed after the kernel hook <code>\@kernel@before@para@begin</code> (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with <code>\noindent</code>). The indentation box to be typeset is available to the hook as <code>\IndentBox</code> and its automatic placement (after the hook is executed) can be prevented through <code>\OmitIndent</code> . More precisely <code>\OmitIndent</code> voids the box. The indentation box is then typeset directly after the hook execution by something equivalent to <code>\box\IndentBox</code> followed by the current content of the token register <code>\everypar</code> that it is available to the kernel or to packages (that run some legacy code). One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a <code>\parbox</code> or a <code>\marginpar</code> inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever).
	para/end This hook is executed at the end of a paragraph when $\text{\TeX}$ is ready to return to vertical mode and after it has removed the last horizontal glue (but not any kerns) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook <code>\@kernel@after@para@end</code> is executed and then $\text{\TeX}$ returns to vertical mode. The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook. ²⁹ This hook is implemented as a reversed hook.
	para/after This hook is executed directly after $\text{\TeX}$ has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a <code>\vadjust</code>) has processed.

²⁸One could allow it but only if the newly started paragraph is processed without any hooks. Further—more correct spacing would be a bit of a nightmare so for now this is forbidden.

²⁹Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

```
\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
```

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after `TEX` has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

2.2 Altered and newly provided commands

```
\par
\endgraf
\para_end:
```

An explicit request for ending a paragraph is provided in plain `TEX` under the name `\endgraf`, which simply uses the primitive meaning (regardless of what `\par` may have as its current definition). In `LATEX` `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in `LATEX`, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (along with `\par` remaining subject to its current meaning).

The `expl3` name for this functionality is `\para_end:`.

Note: *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

<hr/> <code>\OmitIndent</code> <code>\para_omit_indent:</code> <hr/>	<p>Inside the <code>para/begin</code> hook one can use this command to suppress the indentation box at the start of the paragraph by emptying the <code>\IndentBox</code>.</p> <p>If the command is used in vertical mode then it simply issues a <code>\noindent</code>, i.e., starts a paragraph without indentation.</p> <p>The <code>expl3</code> name for the function is <code>\para_omit_indent:</code>.</p>
<hr/> <code>\IndentBox</code> <code>\g_para_indent_box</code> <hr/>	<p>The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. The <code>\OmitIndent</code> command clears its content.</p> <p>The <code>expl3</code> name for the box register is <code>\g_para_indent_box</code>.</p>
<hr/> <code>\RawIndent</code> <code>\para_raw_indent:</code> <code>\RawNoindent</code> <code>\para_raw_noindent:</code> <code>\RawParEnd</code> <code>\para_raw_end:</code> <hr/>	<p><code>\RawIndent</code> <i>hmode material</i> <code>\RawParEnd</code> <code>\RawNoindent</code> <i>hmode material</i> <code>\RawParEnd</code></p> <p>The commands <code>\RawIndent</code> and <code>\RawNoindent</code> are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where <code>T_EX</code>'s low-level algorithm is used to achieve special effects, but where the result is not a "paragraph".</p> <p>They are called "raw", because they bypass L^AT_EX's hook mechanism for paragraphs and simply invoke the low-level <code>T_EX</code> algorithm. I.e., they are like the original <code>T_EX</code> primitives <code>\indent</code> and <code>\noindent</code> (that is they execute no hooks other than <code>\everypar</code>) except that they can only be used in vertical mode and generate an error if found elsewhere.</p> <p>To avoid issues a paragraph started by them should always be ended by <code>\RawParEnd</code>³⁰ and not by <code>\par</code> (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray <code>\pars</code>.</p> <p>The <code>expl3</code> names for the functions are <code>\para_raw_indent:</code>, <code>\para_raw_noindent:</code> and <code>\para_raw_end:</code>.</p>

2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

```

PARA: 1-i start (document env. on input line 38)
PARA: 1-i end   (document env. on input line 38)

```

³⁰Technical note for those who know their *T_EXbook*: the `\RawParEnd` command invokes the original `TEX` engine definition of `\par` that (solely) triggers the paragraph builder in `TEX` when found inside unrestricted horizontal mode and does nothing in other processing modes.


```

PARA: 2-i start (document env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
PARA: 3-ii end (minipage env. on input line 40)
PARA: 2-i end (document env. on input line 41)

```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that L^AT_EX considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

```

\newcounter{paracnt}          % sequence counter
\newcounter{paralevel}        % level counter

```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using expl3 functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of L^AT_EX 2_ε thrown in as well. When popping, the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
  {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop {\seq_gpop:NNF \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff

```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```

\makeatletter % because we use a few internal 2e commands
\AddToHook{para/begin}{%
  \stepcounter{paracnt}\stepcounter{paralevel}%
  \ParaPush
}

```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```

\typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
  (\@currenenvir\space env.\on@line)}%

```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where L^AT_EX sees the start and end of the paragraphs in the document.

```

\llap{\color{red}\tiny\arabic{paracnt}\ }%
}

```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```

\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
    (\@currenvir\space env.\on@line)}%
}

```

We also typeset again a tiny red number with that value, this time sticking out to the right.³¹ We also decrement the level counter since our level has finished.

```

\rlap{\color{red}\tiny\ \paracntvalue}%
\addtocounter{paralevel}{-1}%
}
\makeatother

```

2.3.2 Mark the first paragraph of each `itemize`

The code for this is rather simple. We supply some code that is executed only once inside a hook at the start of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```

\AddToHook{env/itemize/begin}{%
  \AddToHookNext{para/begin}{\color{blue}}%
  \AddToHookNext{para/end}{\color{black}}%
}

```

As a result the first paragraph of each `itemize` will appear in blue.

2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

2.4.1 Glue items between paragraphs (found with `fancypar`)

In the past $\text{\LaTeX}$ placed two glue items between two consecutive paragraphs, e.g.,

```
text1 \par text2 \par
```

would show something like

```

\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669

```

but now there is another `\parskip` glue (that is always 0pt):

```

\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669

```

³¹Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn’t contribute anything vertically but if somebody writes code that unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else it will fail.

3 The Implementation

```

1 <@=para>
2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2021/06/01}{ltpara}
5 <latexrelease> {Paragraph-handling-and-hooks}

```

3.1 Providing hooks for paragraphs

`para/before` The public hooks. They are implemented as a paired set of hooks.

```

para/after 6 \hook_new_pair:n{para/before}{para/after}
para/begin 7 \hook_new_pair:n{para/begin}{para/end}
para/end

```

(End of definition for `para/before` and others. These functions are documented on page 443.)

`\@kernel@before@para@before` The corresponding kernel hooks (for tagging and future extensions).

```

8 \let \@kernel@before@para@before \@empty
9 \let \@kernel@before@para@begin \@empty
10 \let \@kernel@after@para@end \@empty
11 \let \@kernel@after@para@after \@empty

```

(End of definition for `\@kernel@before@para@before` and others. These functions are documented on page 444.)

`\g__para_standard_everypar_tl`

Whenever $\text{\TeX}$ starts a paragraph it inserts first an indentation box and then executes the tokens stored in `\tex_everypar:D` (known to $\text{\LaTeX}$ as `\everypar`). We alter this behavior slightly here, so that hooks are added into the right place. Otherwise the process change remains transparent to any legacy code for this space.

We keep the standard code to be used by `\tex_everypar:D` in a separate token list because we have to switch back and forth for error recovery and so altering `\tex_everypar:D` all the time should be a tiny bit faster.

```

12 <latexrelease> \IncludeInRelease{2023/06/01}
13 <latexrelease> { \g__para_standard_everypar_tl } { minipage~ fix }
14 \tl_new:N \g__para_standard_everypar_tl

```

Here is now its definition:

```

15 \tl_gset:Nn \g__para_standard_everypar_tl {

```

First we remove the indentation box and store it in `\g_para_indent_box`. If there was none because the paragraph was started by `\noindent` the box register will be void.

```

16 \box_gset_to_last:N \g_para_indent_box

```

This will make the newly started horizontal list empty, so if we stop it now and return to vertical mode it will be dropped by $\text{\TeX}$. We do that but inside a group so that any `\parshape` settings will not get lost as we need them for later.

```

17 \group_begin:
18 \tex_par:D
19 \group_end:

```

We then change `\tex_everypar:D` to generate an error so that we can detect and report if the `para/before` hook illegally changed out of vmode.

```

20 \tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
21 \@kernel@before@para@before
22 \hook_use:n {para/before}

```

Assuming the hooks have been well behaved it is time to return to horizontal mode and start the paragraph in earnest. We already have the indentation box saved away so we now have to restart the paragraph with an empty `\tex_everypar:D` and with `\tex_noindent:D`. And we need to make sure not to get another `\parskip` or rather (since we can't prevent that) that it is of zero size.

```

23 \group_begin:
24 \tex_everypar:D {}

```

There has been a long-standing problem with L^AT_EX's minipages in that invisible material at the beginning of a minipage (such as a `\color` setting) would result in `\parskip` being added in front of the first paragraph—something that is not done by T_EX if a vertical list is completely empty. As this is happening on a very low-level in the engine it wasn't really possible to find out if this `\parskip` was added or if a space we see in front of the current point is legitimate. However, with the new paragraph handling we are in a better position: while we still don't know if there is such a space or not, we do know if we have just created an empty paragraph. Thus, if we now set `\parskip` to `-\parskip` the two will cancel each other if present and if the first was ignored because the vertical list was empty, then the second will be ignored too because it is still empty. Of course, we don't want to cancel always but only at the start of a minipage and that is signaled with the `@minipage` switch.

```

25 \skip_set:Nn \tex_parskip:D
26 { \if@minipage -\tex_parskip:D \else: \c_zero_skip \fi: }
27 \tex_noindent:D
28 \group_end:

```

That brings us back to the start of the horizontal list but we need to change `\tex_everypar:D` back to its normal content in case there are nested paragraphs coming up.

```

29 \tex_everypar:D{\g__para_standard_everypar_tl}

```

This is followed by executing the kernel and the public hook. The kernel hook is there to enable tagging.

```

30 \@kernel@before@para@begin
31 \hook_use:n {para/begin}

```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```

32 \if_mode_horizontal: \else:
33 \msg_error:nnnn { hooks }{ para-mode }{begin}{horizontal} \fi:

```

Finally we reinsert the indentation box (unless suppressed) and then call `\everypar` the way legacy L^AT_EX code expects it.

However, adding the public `\everypar` is a bit tricky (see below) so we add that later, and indirectly.

```

34 \__para_handle_indent:
35 % \the \everypar % <--- done differently below
36 }

```

```

37 <latexrelease>\cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
38 <latexrelease>\tl_gput_right:Nx \g__para_standard_everypar_tl {
39 <latexrelease> \exp_not:N \the
40 <latexrelease> \exp_not:N \toks
41 <latexrelease> \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
42 <latexrelease> \c_space_tl
43 <latexrelease>}
44 <latexrelease>\EndIncludeInRelease
45 <latexrelease>\IncludeInRelease{2021/06/01}
46 <latexrelease> \{ \g__para_standard_everypar_tl \} {minipage~ fix}
47 <latexrelease>
48 <latexrelease>\tl_gset:Nn \g__para_standard_everypar_tl {
49 <latexrelease> \box_gset_to_last:N \g_para_indent_box
50 <latexrelease> \group_begin:
51 <latexrelease> \tex_par:D
52 <latexrelease> \group_end:
53 <latexrelease> \tex_everypar:D { \msg_error:nnnn { hooks } { para-mode } { before } { vertical } }
54 <latexrelease> \@kernel@before@para@before
55 <latexrelease> \hook_use:n { para/before }
56 <latexrelease> \group_begin:
57 <latexrelease> \tex_everypar:D {}
58 <latexrelease> \skip_zero:N \tex_parskip:D
59 <latexrelease> \tex_noindent:D
60 <latexrelease> \group_end:
61 <latexrelease> \tex_everypar:D { \g__para_standard_everypar_tl }
62 <latexrelease> \@kernel@before@para@begin
63 <latexrelease> \hook_use:n { para/begin }
64 <latexrelease> \if_mode_horizontal: \else:
65 <latexrelease> \msg_error:nnnn { hooks } { para-mode } { begin } { horizontal } \fi:
66 <latexrelease> \__para_handle_indent:
67 <latexrelease>}

```

We also have to add the `\everypar` toks register at the end. In case of rollback this is already allocated and we have to find out the correct number (hope this is correctly done)

```

68 <latexrelease>\cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
69 <latexrelease>\tl_gput_right:Nx \g__para_standard_everypar_tl {
70 <latexrelease> \exp_not:N \the
71 <latexrelease> \exp_not:N \toks
72 <latexrelease> \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
73 <latexrelease> \c_space_tl
74 <latexrelease>}
75 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\g__para_standard_everypar_tl`.)

`\tex_everypar:D` `\tex_everypar:D` then only has to execute `\g__para_standard_everypar_tl` by default.

```

76 \tex_everypar:D { \g__para_standard_everypar_tl }

```

(End of definition for `\tex_everypar:D`.)

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy L^AT_EX code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc.

Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy L^AT_EX code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is now their token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```
77 \newtoks \everypar
```

After we have allocated a new toks register with the name `\everypar` the actual register number is available (briefly) inside `\allocationnumber`. So instead of `\the\everypar` we have to put `\the\toks{allocated number}` at the end of `\tex_everypar:D`.

So what remains doing is to append a few tokens to the token list `\g__para_standard_everypar_tl` which we do now. We use `x` expansion here to get the value of `\allocationnumber` in, all the other tokens should not be expanded at this point.

One important point here is to terminate the register allocation number with a real space. This space will get swallowed up when the number is read. Anything else, such as `\scan_stop:` would remain in the input and that would mean that it would interfere with `\everypar` code that attempts to scan ahead to see how the paragraph text starts.

```
78 \tl_gput_right:Ne \g__para_standard_everypar_tl {
79   \exp_not:N \the
80   \exp_not:N \toks
81   \the \allocationnumber
82   \c_space_tl
83 }
```

(End of definition for `\everypar`.)

`\g_para_indent_box` For managing the indentation we need to provide a public accessible box register

```
84 \box_new:N \g_para_indent_box
```

(End of definition for `\g_para_indent_box`. This function is documented on page 445.)

`__para_handle_indent:` Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```
85 \cs_new:Npn __para_handle_indent: {
86   \box_use_drop:N \g_para_indent_box
87 }
```

The declaration `\para_omit_indent:` (or `\OmitIndent`) changes that to do nothing—the case when used inside the `para/begin` hook. If encountered in vertical mode it starts the paragraph without indentation.

```
88 \cs_new_protected:Npn \para_omit_indent: {
89   \mode_if_vertical:TF
90     { \noindent }
91     { \box_gclear:N \g_para_indent_box }
92 }
```

(End of definition for `\_para_handle_indent:`)

`\IndentBox` The L^AT_EX 2_ε names for the indentation box and for suppressing it for use in the `para/begin` hook.

```
93 \cs_set_eq:NN \IndentBox \g_para_indent_box
94 \cs_set_eq:NN \OmitIndent \para_omit_indent:
```

(End of definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 445.)

`\para_end:` Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by T_EX to end horizontal mode and return to vertical mode, i.e., `\par`.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by T_EX in certain situations:

- when using `\par` in the code or the document
- when using a blank line (which is converted to `\par`)
- when T_EX finds any commands incompatible with horizontal mode it issues a `\par` and then rereads the command.

Unfortunately, T_EX has some (these days) unnecessary optimizations: if a `\vbox` ends and T_EX is still in horizontal mode it simply exercises the paragraph builder instead of issuing a `\par`. It is therefore necessary for L^AT_EX to ensure that this case doesn't happen and all boxes internally have a `\par` command at their end.

This `\par` may or may not run the “par primitive” (which is always available as `\tex_par:D` in `expl3`); it is permissible to have a changed meaning and it is in fact changed by L^AT_EX in various ways at various points inside `latex.ltx`. For this L^AT_EX 2_ε code has the following conventions: `\@@par` and `\endgraf` both refer to the default meaning (in the past this was the `initex` primitive) while `\par` is the current meaning which maybe does something else.

We are now going to change this default meaning to instead run `\para_end:`, which ultimately executes the `initex` primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy L^AT_EX 2_ε code.

In most cases `\para_end:` should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as `\tex_par:D`. This way we don't have to care about whether T_EX just does nothing (e.g., if in vertical mode already) or generates an error, etc.

```
95 \cs_new_protected:Npn \para_end: {
```

CCC Maybe needs more explanation. TEMP NOTE: What should happen if in outer hmode with an empty hlist?

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an `\hbox`).

```
\bool_lazy_and:nnT
  { \mode_if_horizontal_p: }
  { \bool_not_p:n { \mode_if_inner_p: } }
  { ...
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible, so we do not use the above construct but two conditionals instead. Using low-level `\if_mode...` conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If `\para_end:` is executed while $\TeX$ is currently doing a low-level assignment the test for horizontal mode may get executed as part of the assignment. That is normally not an issue but we just found one case where it is:

```
\afterassignment\lst@vskip\@tempskipa \z@ \par
```

If $\TeX$ is in hmode while that assignment happens then the `\par` is seen in hmode because in the above case the assignment may not be finished (one should have used `\z@skip`) and the `\lst@vskip` will get inserted into the middle of the conditional. The `\lst@vskip` then changes to vmode and you get a surprising error about the `para/end` hook having changed modes even if you don't have any hook code(!): it is the inserted `\lst@vskip` that is actually causing the change of mode. This is what happened when the output routines got started while a `lstlisting` environment (that redefines `\vskip` in this way) was active. This is really faulty coding, but we try to be proactive and guard the conditional so that any scanning is first stopped, thus:

```
96   \scan_stop:
97   \mode_if_horizontal:TF {
98     \mode_if_inner:F {
```

In that case the action of the primitive would be to remove the last glue (but no kerns) from the horizontal list (constructed to form a paragraph) and then to append a penalty of 10000 and the `\parfillskip`; it then passes the whole list to the paragraph builder, which breaks it into lines and $\TeX$ then returns to vertical mode.

What we want to do is to add this hook code at the end of the horizontal list before any of the above happens. If there was a glue item at the end of the list then it should get removed before the hook code gets added so we have to arrange for this removal.

As in other similar cases, it may be best to add here a `\nobreak` in case the hook itself adds glue and thus creates a non-explicit and unwanted potential breakpoint. On the other hand (as has been argued) the code in the hook should perhaps have the responsibility for adding such a guard penalty in this case. This needs further analysis and decisions (as in emails).

In either case, good documentation of these hooks is essential, covering what the hook may or should provide and all such related considerations concerning the content.

There is not much point in checking if there was really a glue item at the end of the horizontal list, instead we simply try to remove one using `\tex_unskip:D`: if there wasn't one this will do nothing.

```
99       \tex_unskip:D
```

We then execute the public hook (which may add some final typeset material) followed by the kernel hook that we need for adding tagging support. None of this is supposed to change the mode—at the moment we make only a very simple test for this, more devious changes go unnoticed, but too bad as they will then probably backfire badly.

```
100     \hook_use:n{para/end}
101     \@kernel@after@para@end
102     \mode_if_horizontal:TF {
```


The final action (before getting to the point where `\tex_par:D` is called) is to add an extra glue item so that the primitive is prevented from removing intended glue (if there was some). If we don't do this and the horizontal list ends in several glue items we would end up removing two glue items instead of just the last one, which would be wrong. We use glue (rather than a kern) as that will be removed by the primitive.

There is however one other T_EX optimization that hurts: in a sequence like this `$$ ... $$ \par` (with `\par` being the primitive) T_EX will be in horizontal mode after the display, ready to receive further paragraph text, but since the `\par` follows immediately there is a “null” paragraph at the end and T_EX simply throws that away. The space between `$$` and `\par` got already dropped during the display processing so the `\par` is not removing any space and appending `\parfillskip`, instead it simply goes silently to vmode.

Now if we would have added something (to prevent glue removal) that would look to T_EX like material after the display and so we would end up with an empty paragraph just containing a penalty and `\parfillskip`.

We therefore check if the current hlist does end in glue (`\tex_lastnodetype:D` has the value 11) and if so we add a zero-length guard skip which will be removed by the following `\tex_par:D`.

```

103         \if_int_compare:w 11 = \tex_lastnodetype:D
104         \tex_hskip:D \c_zero_dim
105         \fi:

```

To run the `para/after` hook we first end the paragraph. This means that the `\tex_par:D` at the very end is unnecessary but executing it there unnecessarily is better than having code that tests for all the different mode possibilities.

```

106         \tex_par:D
107         \hook_use:n{para/after}
108         \@kernel@after@para@after
109     }

```

If we were not horizontal mode (the F case from above) then the earlier hook `para/end` must have been at fault, so we report that.

```

110         { \msg_error:nnnn { hooks }{ para-mode }{end}{horizontal} }

```

Finally close out the nested conditionals.

```

111     }
112 }

```

And then we can use the primitive to truly end the paragraph.

```

113 \tex_par:D
114 }

```

(End of definition for `\para_end:`. This function is documented on page 444.)

`\para_raw_indent:` The commands `\para_raw_indent:` and `\para_raw_noindent:` are like the primitives `\indent` and `\noindent` except that they can only be used in vertical mode.

`\para_raw_end:` To avoid issues a paragraph started by them should always be ended by `\para_raw_end:` and not by `\para_end:` or `\par` as the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired.

```

115 \cs_new:Npn \para_raw_indent: {
116     \mode_if_vertical:TF
117     {

```

```

118         \tex_everypar:D {
119             \box_gset_to_last:N \g_para_indent_box
120             \tex_everypar:D { \g__para_standard_everypar_tl }
121             \__para_handle_indent:
122             \the\everypar }
123     }
124     { \msg_error:nn { latex2e }{ raw-para } }
125 \tex_indent:D
126 }

127 \cs_new:Npn \para_raw_noindent: {
128     \mode_if_vertical:TF
129     {
130         \tex_everypar:D {
131             \tex_everypar:D { \g__para_standard_everypar_tl }
132             \the\everypar }
133     }
134     { \msg_error:nn { latex2e }{ raw-para } }
135 \tex_noindent:D
136 }

137 \cs_new_eq:NN \para_raw_end: \tex_par:D

```

(End of definition for `\para_raw_indent:`, `\para_raw_noindent:`, and `\para_raw_end:`. These functions are documented on page 445.)

`\RawIndent` The L^AT_EX 2_ε names for starting and ending a paragraph without adding any hooks.
`\RawNoIndent`
`\RawParEnd`

```

138 \cs_set_eq:NN \RawIndent \para_raw_indent:
139 \cs_set_eq:NN \RawNoindent \para_raw_noindent:
140 \cs_set_eq:NN \RawParEnd \para_raw_end:

```

(End of definition for `\RawIndent`, `\RawNoIndent`, and `\RawParEnd`. These functions are documented on page 445.)

This ends the `para` module code.

```

141 <@@=

```

`\par` Having the new default definition for `\par` we also have to set it up so that it gets used.
`\endgraf` This involves three commands: `\par`, `\@@par` (to which L^AT_EX resets `\par` occasionally)
`\@@par` and `\endgraf`, which is another name for the “default” action of `\par`.

```

142 \cs_set_eq:NN \par \para_end:
143 \cs_set_eq:NN \@@par \para_end:
144 \cs_set_eq:NN \endgraf \para_end:

```

(End of definition for `\par`, `\endgraf`, and `\@@par`. These functions are documented on page 444.)

While this is not integrated properly into the format we have to redo the `\everypar` setting from the kernel, otherwise that gets lost (as it happens before that file is loaded).

```

145 \everypar{\@nodocument} %% To get an error if text appears before the \document

```

3.2 The error messages

This one is used when we detect that some hook code has changed the mode where it shouldn't, e.g., by starting or ending a paragraph. The first argument is the hook name second the mode it should have stayed in but didn't.

```

146 \msg_new:nnnn { hooks } { para-mode }
147 {
148   Illegal~mode~ change~ in~ hook~ 'para/#1'.\\
149   Hook~ code~ did~ not~ remain~ in~ #2~ mode.
150 }
151 {
152   Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
153   endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
154   in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
155   code~ with~ \iow_char:N \\\ShowHook~ to~ find~ the~ issue.
156 }

```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```

157 \msg_new:nnnn { latex2e } { raw-para }
158 {
159   Not~ in~ vertical~ mode.
160 }
161 {
162   Starting~ a~ paragraph~ with~ \iow_char:N \\\RawIndent~ or~
163   \iow_char:N \\\RawNoindent \\
164   (or~ \iow_char:N \\\para_raw_indent:~ or~
165   \iow_char:N \\\para_raw_noindent:)~ is~ only~ allowed \\
166   if~ LaTeX~ is~ in~ vertical~ mode.
167 }

168 %
169 <latexrelease>\IncludeInRelease{0000/00/00}%
170 <latexrelease>          {ltpara}{Undo~hooks~for~paragraphs}
171 <latexrelease>
172 <latexrelease>\let \OmitIndent \@undefined
173 <latexrelease>\let \IndentBox \@undefined
174 <latexrelease>\let \RawIndent \@undefined
175 <latexrelease>\let \RawNoindent \@undefined
176 <latexrelease>\let \RawParEnd \@undefined
177 <latexrelease>
178 <latexrelease>\cs_set_eq:NN \par \tex_par:D
179 <latexrelease>\cs_set_eq:NN \@@par \tex_par:D
180 <latexrelease>\cs_set_eq:NN \endgraf \tex_par:D
181 <latexrelease>

```

We also need to clean up the primitive “everypar” as that should no longer execute any code by default. And, of course, make \everypar become the primitive again.

```

182 <latexrelease>\tex_everypar:D {}
183 <latexrelease>\cs_set_eq:NN \everypar \tex_everypar:D
184 <latexrelease>
185 <latexrelease>\EndModuleRelease
186 \ExplSyntaxOff
187 </2ekernel | latexrelease>

```

File 17

ltmeta.dtx

Abstract

This code defines the `\DocumentMetadata` interface.

1 Introduction

In the past there was no dedicated location to declare settings concerning a document as a whole. Settings are placed somewhere in the preamble or with the class options or even with some package options. For some settings this can be too late, for example the pdf version can no longer be changed if a package has used code which already opened the PDF.

`\DocumentMetadata` as a new command unifies such settings in one place. It must be used before `\documentclass` but can be issued more than once there.

At the moment most of the code run by `\DocumentMetadata` is external to the format and subject to change. This includes the supported key/values.

For that reason all that happens right now in the format is to look for suitable support files and if found, to redirect the processing to them.

1.1 `\DocumentMetadata`

<code>\DocumentMetadata</code>	<code>\DocumentMetadata{<key-value list>}</code>
--------------------------------	--

The keys defined for `\DocumentMetadata` currently allow to set the PDF version, to set the PDF `/Lang`, to uncompress a PDF, to set the language and to declare a few PDF standards and some color profiles.

`\DocumentMetadata` is also used to activate the new PDF management code and it loads a number of required files for the PDF management code. As this forces the loading of the backend files, a backend which can't be detected automatically like `dvipdfmx`, must be set in the first `\DocumentMetadata` call (if there is more than one).

The full set of keys currently supported is documented in `documentmetadata-support.pdf` for now.

1.2 Storing and retrieving document properties

We provide a storage container where classes, packages and users can store properties and metadata about the current document which may be of interest to other packages or to the user. This data can then be retrieved as necessary by packages or by the document author.

The properties are stored with a key `<label>/<property>`. The values can be retrieved expandably.

<hr/> <code>\AddToDocumentProperties</code> <hr/>	<code>\AddToDocumentProperties[\langle label \rangle]{\langle property \rangle}{\langle value \rangle}</code> This stores $\langle value \rangle$ under the key $\langle label \rangle/\langle property \rangle$. By default $\langle label \rangle$ is the current package name <code>\@currname</code> . If another label is chosen, it should be one which avoids clashes with other packages using the container. The label <code>document</code> is used by the PDF management and by the L ^A T _E X kernel and so reserved. If the key $\langle label \rangle/\langle property \rangle$ already exists in the container, its value is overwritten without warning or error.
<hr/> <code>\GetDocumentProperty</code> <hr/>	<code>\GetDocumentProperty{\langle label/property \rangle}</code> Expands to the $\langle value \rangle$ corresponding to $\langle label/property \rangle$ in the container. If $\langle label/property \rangle$ is missing, this has an empty expansion. In particular, there is no error! The result is returned within <code>\exp_not:n</code> , which means that the $\langle value \rangle$ does not expand further when appearing in an e-type or x-type argument expansion.
<hr/> <code>\ShowDocumentProperties</code> <code>\LogDocumentProperties</code> <hr/>	<code>\ShowDocumentProperties</code> <code>\LogDocumentProperties</code> This shows/logs the current content of the container.

2 The Implementation

2.1 `\DocumentMetadata`

```
1 < *2ekernel | latexrelease >
```

Not needed yet but ...

```
2 %\ExplSyntaxOn
3 < latexrelease > \NewModuleRelease{2022/06/01}{ltmeta}
4 < latexrelease > \DocumentMetadata handling
```

We start by making the conditionals testing for the use of `\DocumentMetadata` execute the false branch. Then, inside `\DocumentMetadata`, we change them to select the true branch.

```
5 \let \IfDocumentMetadataTF \@secondoftwo
6 \let \IfDocumentMetadataT \@gobble
7 \let \IfDocumentMetadataF \@firstofone
8 \protected\def\DocumentMetadata{%
9   \InputIfFileExists{documentmetadata-support.ltx}%
10  {%
```

Support for `\DocumentMetadata` is available, so change the conditionals.

```
11   \let \IfDocumentMetadataTF \@firstoftwo
12   \let \IfDocumentMetadataT \@firstofone
13   \let \IfDocumentMetadataF \@gobble
14 }%
```

The above file is changing `\DocumentMetadata` to a suitable definition (or so we hope) so that it can be safely used several times in the preamble.

If the file can't be found, we say so and carry on without it.

```
15   {%
16     \latex@error{No support files for
17               \noexpand\DocumentMetadata found}
18     {Is the 'LaTeX-lab' bundle installed?}%
19     \MessageBreak
```

20 Without it, the declaration is ignored.}%

No point in trying this more than once if there are several calls in the document.

```
21       \let\DocumentMetadata@gobble
22       }%
23       \DocumentMetadata
24   }
```

2.2 Document properties

The container for the document properties is a prop:

`\g__kernel_documentproperties_prop`

```
25   \ExplSyntaxOn
26   \prop_new:N \g__kernel_documentproperties_prop
```

`\AddToDocumentProperties`

```
27   \NewDocumentCommand\AddToDocumentProperties{0{\@currname}mm}
28   {
29     \prop_gput:Nen \g__kernel_documentproperties_prop
30     {
31       \tl_if_blank:eTF {#1}{top-level/}{#1/} #2
32     }
33     { #3}
34   }
```

(End of definition for `\AddToDocumentProperties`. This function is documented on page 458.)

`\GetDocumentProperty`

```
35   \cs_new:Npn\GetDocumentProperty #1
36   {
37     \prop_item:Nn \g__kernel_documentproperties_prop {#1}
38   }
```

(End of definition for `\GetDocumentProperty`. This function is documented on page 458.)

`\ShowDocumentProperties`

`\LogDocumentProperties`

```
39   \msg_new:nnn { latex2e } { show-properties }
40   {
41     The~following~document~properties~have~been~stored:
42     #1
43   }
44   \NewDocumentCommand\ShowDocumentProperties {}
45   {
46     \msg_show:nne {latex2e}{show-properties}
47     {
48       \prop_map_function:NN \g__kernel_documentproperties_prop \msg_show_item:nn
49     }
50   }
51   \NewDocumentCommand\LogDocumentProperties {}
52   {
53     \msg_log:nne {latex2e}{show-properties}
54     {
55       \prop_map_function:NN \g__kernel_documentproperties_prop \msg_show_item:nn
```

```

56     }
57   }
58   \ExplSyntaxOff

```

(End of definition for `\ShowDocumentProperties` and `\LogDocumentProperties`. These functions are documented on page 458.)

2.3 Targets

To allow package and class authors to support document links we also provide the new interface commands of the `hyperref` package for the creation of targets. The commands are not only useful for document links: When tagging is active the commands can also be used to reference structures. This is done in the `recordtarget` socket, which records the target name. Both for links and structure references it can be useful to change the name of the next target. We therefore insert a hook `target/setname/after` after the name has been set. This hook is used in the `\NextLinkTarget` command to adapt the target name `\@currentHref`.

```

\MakeLinkTarget
\LinkTargetOn      59 <latexrelease>\IncludeInRelease{2026/06/01}%
\LinkTargetOff    60 <latexrelease>                {\MakeLinkTarget}{Record target name for tagging support}%
\NextLinkTarget   61 \ExplSyntaxOn

```

luatex complains about non-PDF specials if `\special` is used, so we use `\latelua` instead to create an empty whatsit node.

```

62 \ifx\directlua\@undefined
63   \def\@kernel@whatsit{\special{}}
64 \else
65   \def\@kernel@whatsit{\latelua{}}
66 \fi

```

This definition is copied from `hyperref`, only the command names are changed. It is the code `hyperref` uses around anchors/destination to avoid affecting the spacing.

```

67 \let\RestoreLastSkip\relax
68 \def\SaveLastSkip{%
69   \let\RestoreLastSkip\relax
70   \ifvmode
71     \ifdim\lastskip=\z@
72       \ifnum\lastnodetype=11 %
73         \let\RestoreLastSkip\nobreak
74       \else
75         \let\RestoreLastSkip\relax
76       \fi
77     \else
78       \begingroup
79         \skip@=-\lastskip
80         \edef\x{%
81           \endgroup
82           \def\noexpand\RestoreLastSkip{%
83             \noexpand\ifvmode
84               \noexpand\nobreak
85               \vskip\the\skip@
86               \vskip\the\lastskip\relax
87             \noexpand\fi

```

```

88         }%
89     }%
90     \x
91     \fi
92     \else
93         \ifhmode
94             \ifdim\lastskip=\z@
95                 \let\RestoreLastSkip\nobreak
96             \else
97                 \beginingroup
98                     \skip@=-\lastskip
99                     \edef\x{%
100                         \endgroup
101                         \def\noexpand\RestoreLastSkip{%
102                             \noexpand\ifhmode
103                                 \noexpand\nobreak
104                                 \hskip\the\skip@
105                                 \hskip\the\lastskip\relax
106                             \noexpand\fi
107                         }%
108                     }%
109                 \x
110             \fi
111         \fi
112     \fi
113 }
114 \hook_new:n{target/setname/after}
115 \int_new:N\g__kernel_target_int
116 \NewDocumentCommand\MakeLinkTarget{s0{}m}{%
117     \ifvmode
118         \SaveLastSkip\@kernel@whatsit\RestoreLastSkip
119     \else
120         \ifhmode
121             \@savs\spacefactor
122             \SaveLastSkip\smash{}\RestoreLastSkip
123             \spacefactor\@savs
124         \fi
125     \fi
126     \IfBooleanTF {#1}
127     {
128         \tl_gset:Ne \@currentHref {#3}
129     }
130     {
131         \int_gincr:N\g__kernel_target_int
132         \tl_gset:Ne \@currentHref {target*.\int_use:N\g__kernel_target_int}
133     }
134     \hook_use:n{target/setname/after}
135     \UseTaggingSocket{recordtarget}
136 }
137 \ExplSyntaxOff
138 <latexrelease> \EndIncludeInRelease
139 <latexrelease> \IncludeInRelease{2024/11/01}%
140 <latexrelease>           {\MakeLinkTarget}{Record target name for tagging support}%
141 <latexrelease> \ExplSyntaxOn

```



```

142 <latexrelease> \NewDocumentCommand\MakeLinkTarget{sO{m}}{%
143 <latexrelease> \ifvmode
144 <latexrelease> \special{}%
145 <latexrelease> \else
146 <latexrelease> \@savsf\spacefactor
147 <latexrelease> \smash{}%
148 <latexrelease> \spacefactor\@savsf
149 <latexrelease> \fi
150 <latexrelease> \IfBooleanTF {#1}
151 <latexrelease> {
152 <latexrelease> \tl_gset:Ne \@currentHref {#3}
153 <latexrelease> }
154 <latexrelease> {
155 <latexrelease> \int_gincr:N\g__kernel_target_int
156 <latexrelease> \tl_gset:Ne \@currentHref {target*.\int_use:N\g__kernel_target_int}
157 <latexrelease> }
158 <latexrelease> \hook_use:n{target/setname/after}
159 <latexrelease> \UseTaggingSocket{recordtarget}
160 <latexrelease> }
161 <latexrelease> \ExplSyntaxOff
162 <latexrelease> \EndIncludeInRelease
163 <latexrelease> \IncludeInRelease{2022/06/01}%
164 <latexrelease> { \MakeLinkTarget } { Record target name for tagging support } %
165 <latexrelease> \NewDocumentCommand\MakeLinkTarget{sO{m}}{%
166 <latexrelease> \ifvmode
167 <latexrelease> \special{}%
168 <latexrelease> \else
169 <latexrelease> \@savsf\spacefactor
170 <latexrelease> \smash{}%
171 <latexrelease> \spacefactor\@savsf
172 <latexrelease> \fi}
173 <latexrelease> \EndIncludeInRelease
174 \NewDocumentCommand\LinkTargetOn{}{}
175 \NewDocumentCommand\LinkTargetOff{}{}
176 \NewDocumentCommand\NextLinkTarget{m}{%
177 \AddToHookNext{target/setname/after}{\xdef\@currentHref {#1}}}
(End of definition for \MakeLinkTarget and others.)

We do not undo \MakeLinkTarget and friends if we roll back, in case they are used
in packages that themselves do not offer rollback. This way a roll forward adds them,
but the dummies remain if you roll back and you don't get missing csname errors if they
are used.

178 <latexrelease> \IncludeInRelease{0000/00/00}{ltmeta}%
179 <latexrelease> {Undo Document Metadata handling}
180 <latexrelease>
181 <latexrelease> \let\DocumentMetadata\@undefined
182 <latexrelease>
183 <latexrelease> \EndModuleRelease
Again for the future ...
184 %\ExplSyntaxOff
185 </2ekernel | latexrelease>
Restore module prefix (if any):
186 <@@=>

```

File 18

ltspace.dtx

1 Spacing

This section deals with spacing, and line- and page-breaking.

1.1 User Commands

`\nopagebreak` [$\langle i \rangle$] : $\langle i \rangle = 0, \dots, 4$.
Default argument = 4. Puts a penalty into the vertical list output as follows:
0 : penalty = 0
1 : penalty = `\@lowpenalty`
2 : penalty = `\@medpenalty`
3 : penalty = `\@highpenalty`
4 : penalty = 10000
`\pagebreak` [$\langle i \rangle$] : same as except negatives of its penalty
`\linebreak` [$\langle i \rangle$] : analog of the above
`\nolinebreak` [$\langle i \rangle$] : analog of the above
`\samepage` : inhibits page breaking most places by setting the following penalties to 10000:
`\interlinepenalty`
`\predisplaypenalty`
`\postdisplaypenalty`
`\interdisplaylinepenalty`
`\@beginparpenalty`
`\@endparpenalty`
`\@itempenalty`
`\@secpenalty`
`\interfootnotelinepenalty`
`\` : initially defined to be `\newline`
`\` [$\langle length \rangle$] : initially defined to be `\vspace{\langle length \rangle}\newline`
Note: `\`* adds a `\adjust{\penalty 10000}`
OBSOLETE COMMANDS (which never made it into the manual):
`\obeycr` : defines `<CR> == \`
`\restorecr` : restores `<CR>` to its usual meaning.

1.2 Chris' comments

There are several aspects of the handling of space in horizontal mode that are inconsistent or do not work well in some cases. These are largely concerned with ignoring the effect of space tokens that would otherwise typeset an inter-word space.

Negating the effect of such space tokens is achieved by two mechanisms:

- `\unskip` is used to remove the glue just added by a space that has already had its effect; it is sometimes invoked after an `\ifdim` test on `\lastskip` (see below);
- `\ignorespaces` is used to ignore space-tokens yet to come.

The test done on `\lastskip` is sometimes for equality with zero and sometimes for being positive. Recall also that the test is only on the natural length of the glue and that no glue cannot be distinguished from glue whose natural length is zero: to summarise, a pretty awful test. It is not clear why these tests are not all the same; I think that they should all be for equality. One place where `\unskip` is often used is just before a `\par` (which itself internally does an `\unskip`) and one bit of code (in `\@item`) even has two `\unskips` before a `\par`. These uses may be fossil code but if they are necessary, maybe `\@killglue` would be even safer.

Such removal of glue by `\unskip` may sometimes have the wrong result, removing not the glue from a space-token but other explicit glue; this is sometimes not what is intended.

A common way to prevent such removal is to add an `\hskip\z@` after the glue that should not be removed. This protects that glue against one `\unskip` with no test but not against more than one. It does work for ‘tested `\unskips`’. This is used by `\hspace*` but not by `\hspace`; this is inconsistent as the star is supposed to prevent removal only at the beginning of a line, not at the end, or in a tabular, etc.

If this reason for removing glue were the only consideration then a tested-`\unskip` and protection by `\hskip\z@` would suffice but would need to be consistently implemented.

However, the class of invisibles, commands and environments tries to be even cleverer: one of these tries to leave only one inter-word space whenever there is one before it and one after it; and it does this quite well.

But problems can arise when there is not a space-token on both sides of it; in particular, when an invisible appears at the beginning or end of a piece of text the method still leaves one space token whereas usually in these cases it should leave none.

Also, the current rules do not work well when more than one such command appears consecutively, separated by space-tokens; it leaves glue between every other invisible.

There is also a question about what these commands should do when they occur next to spaces that do not come from space tokens but, for example, from `\hspace`. Should they still produce ‘just one space’? If so, which one? It is good to note that the manual is sufficiently cautious about invisibles that we are not obliged to make anything work.

Another interesting side-road to explore is whether the space-tokens either side of an `\hspace{...}` should be ignored.

One alternative to the current algorithm that is often suggested is that all glue around the invisible should be consolidated into a space after it (usually without stating how much glue should be put there). The command `\nolinebreak` is implemented this way (and `\linebreak` should also be). This does not work correctly for the following common case:

```
... some text
\index{some-word}
some-word and more text.
```

This is optimal coding since it is normal to index a word that gets split across a page-break on its starting page. This would, on the other hand, fix another common (and documented) failure of the current system: when the invisible is the last thing in a paragraph the space before it is not removed and, worse, it is also hidden from the paragraph-ending mechanism so that an ‘empty’ line can be created at the end of the paragraph.

Another deficiency (I think) of the current system is that the following is treated as having the `\index` command between the paragraphs, which is probably not what the author intended (since there is no empty line after it).

```
\index{beginnings}
Beginnings of paragraphs ...
```

I know of no algorithm that will handle satisfactorily even all the most common cases; note that it could be that the best algorithm may be different for different invisibles since, for example, the common uses and expected behaviour of `\index`, `\marginpar`, `\linebreak`, `\pagebreak` and `\vspace` are somewhat different. [For example, is `\vspace` ever used in the middle of a paragraph?]

One method that can (and is) used to make invisible commands produce no space when used at the beginning of text is to put in some glue that is nearly enough the same as no glue or glue of zero length in all respects except for the precise test for not being exactly equal to zero; examples of such glue are `\hskip 1sp` and, possibly better but more complex, `\hskip -1sp \hskip 1sp`. However, this only works when it is known that user-supplied text is about to start.

Some similar concerns apply to the handling of space and penalties in vertical mode; there is an extra hurdle here as `\unskip` does not work on the main vertical list. The complexity of the tests done by `\addvspace` have never been explained.

The implementation of space hacks etc for vertical mode is another major area that needs further attention; my earlier experiments did not produce much improvement over the current unsatisfactory situation.

One particular problem is what happens when the following very natural coding is used (part of the problem here is that this looks like an hmode problem, but it is not):

```
... end of text.

\begin{enumerate}
  \item \label{item:xxx} Item text.
\end{enumerate}
```

1.3 Some immediate actions

- Fix bug in `\linebreak`.
- Fix bug in `\\*`.
- Reimplement `\\`, etc, removing extra `\adjusts` and getting better error trapping (this seems to involve a lot more tokens).
- Investigate whether `\\`, etc need to be errors in vmode; I think that they could be noops (maybe with a warning).
- Make all(?) `\unskips` include test for zero skip (rather than other tests or no test).
- Consider replacing `\hskip 1sp` by something better (here called an ‘infinitesimal’ skip).
- Look at all `\hskip\z@` (or similar) to see if they should be changed to an ‘infinitesimal’ skip.

- Resolve the inconsistency between `\hspace` and `\hspace*`.
- Remove unnecessary `\unskips`.
- Investigate and rationalize the ‘newline’ code.
- Find better algorithms for all sorts of things or, easier(?), fix T_EX itself.

1.4 The code

```

1 <*2ekernel>
2 \message{spacing,}
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease>                {\pagebreak}{Make commands robust}%

\pagebreak
\nopagebreak
7 \DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
8 \DeclareRobustCommand\nopagebreak{\@testopt{\@no@pgbk4}

(End of definition for \pagebreak and \nopagebreak.)

\linebreak
\nolinebreak
9 \DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
10 \DeclareRobustCommand\nolinebreak{\@testopt{\@no@lnbk4}

(End of definition for \linebreak and \nolinebreak.)

\samepage
11 \DeclareRobustCommand\samepage{\interlinepenalty\@M
12   \predisplaypenalty\@M
13   \postdisplaypenalty\@M
14   \interdisplaylinepenalty\@M
15   \@beginparpenalty\@M
16   \@endparpenalty\@M
17   \@itempenalty\@M
18   \@secpenalty\@M
19   \interfootnotelinepenalty\@M}

(End of definition for \samepage.)

20 </2ekernel | latexrelease>
21 <latexrelease>\EndIncludeInRelease
22 <latexrelease>\IncludeInRelease{0000/00/00}%
23 <latexrelease>                {\pagebreak}{Make commands robust}%
24 <latexrelease>
25 <latexrelease>\kernel@make@fragile\pagebreak
26 <latexrelease>\kernel@make@fragile\nopagebreak
27 <latexrelease>\kernel@make@fragile\linebreak
28 <latexrelease>\kernel@make@fragile\nolinebreak
29 <latexrelease>\kernel@make@fragile\samepage
30 <latexrelease>
31 <latexrelease>\EndIncludeInRelease
32 <*2ekernel>

```

`\@no@pgbk`

```
33 \def\@no@pgbk #1[#2]{%
34   \ifvmode
35     \penalty #1\@getpen{#2}%
36   \else
37     \@bsphack
38     \vadjust{\penalty #1\@getpen{#2}}%
39     \@esphack
40   \fi}
```

(End of definition for \@no@pgbk.)

`\@no@lnbk`

```
41 \def\@no@lnbk #1[#2]{%
42   \ifvmode
43     \@nolnerr
44   \else
45     \@tempskipa\lastskip
46     \unskip
47     \penalty #1\@getpen{#2}%
48     \ifdim\@tempskipa>\z@
49       \hskip\@tempskipa
50       \ignorespaces
51     \fi
52   \fi}
```

(End of definition for \@no@lnbk.)

`\` The purpose of the new code is to fix a few bugs; however, it also attempts to optimize the following, in order of priority:

1. efficient execution of plain `\`;
2. efficient execution of `\[...]`;
3. memory use;
4. name-space use.

The changes should make no difference to the typeset output. It appears to be safe to use `\reserved@e` and `\reserved@f` here (other reserved macros are somewhat disastrous).

These changes made `\newline` even less robust than it had been, so now it is explicitly robust, like `\`.

The internal definition of the ‘normal’ definition of `\`.

`\@normalcr`

```
53 </2ekernel>
54 <*2ekernel | latexrelease>
55 <latexrelease> \IncludeInRelease{2020/02/02}%
56 <latexrelease>           {\@normalcr}{Make robust}%
57 \protected\def\@normalcr{%
58   \let \reserved@e \relax
59   \let \reserved@f \relax
60   \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
61             \@xnewline}%
62   \@xnewline}
```

```

63 \let\\@normalcr
64 </2ekernel | latexrelease>
65 <latexrelease>\EndIncludeInRelease
66 <latexrelease>\IncludeInRelease{0000/00/00}%
67 <latexrelease>          {\@normalcr}{Make robust}%
68 <latexrelease>
69 <latexrelease>\DeclareRobustCommand\\{%
70 <latexrelease>  \let \reserved@e \relax
71 <latexrelease>  \let \reserved@f \relax
72 <latexrelease>  \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
73 <latexrelease>          \@xnewline}%
74 <latexrelease>  \@xnewline}
75 <latexrelease>\expandafter\let\expandafter\@normalcr
76 <latexrelease>  \csname\expandafter\@gobble\string\\ \endcsname
77 <latexrelease>
78 <latexrelease>\EndIncludeInRelease
79 <*2ekernel>

```

(End of definition for \\ and \@normalcr.)

\@vspace@calcify Helper command to produce a \vskip that is first run through \setlength. This way the calc package can operate on the argument value.

```

80 </2ekernel>
81 <*2ekernel | latexrelease>
82 <latexrelease>\IncludeInRelease{2020/10/01}%
83 <latexrelease>          {\@vspace@calcify}{Add calc support}%
84 \def\@vspace@calcify#1{\setlength\sp@ce@skip{#1}\vskip\sp@ce@skip}
85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease>          {\@vspace@calcify}{Add calc support}%
89 <latexrelease>
90 <latexrelease>\let\@vspace@calcify\@undefined
91 <latexrelease>\EndIncludeInRelease
92 <*2ekernel>

```

(End of definition for \@vspace@calcify.)

\newline A simple form of the ‘normal’ definition of \\.

```

93 \DeclareRobustCommand\newline{\@normalcr\relax}

```

(End of definition for \newline.)

\@xnewline

```

94 \def\@xnewline{\@ifnextchar[% ] bracket matching
95               \@newline
96               {\@gnewline\relax}}

```

(End of definition for \@xnewline.)

\@newline

```

97 </2ekernel>
98 <*2ekernel | latexrelease>
99 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

100 <latexrelease>                {\@newline}{\newline calc support}%
101 \def\@newline[#1]{\let \reserved@e \vadjust
102                \@gnewline {\@vspace@calcify{#1}}}
103 </2ekernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease

105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease>                {\@newline}{\newline calc support}%
107 <latexrelease>
108 <latexrelease>\def\@newline[#1]{\let \reserved@e \vadjust
109 <latexrelease>                \@gnewline {\vskip #1}}
110 <latexrelease>\EndIncludeInRelease
111 <*2ekernel>

```

(End of definition for \@newline.)

\@gnewline The \nobreak added to prevent null lines when \\ ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```

112 \def\@gnewline #1{%
113     \ifvmode
114         \@nolnerr
115     \else
116         \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
117     \fi}

```

(End of definition for \@gnewline.)

\@getpen

```

118 \def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
119             \@medpenalty \or \@highpenalty
120             \else \@M \fi}

```

(End of definition for \@getpen.)

\if@nobreak Switch used to avoid page breaks caused by \label after a section heading, etc. It should be **GLOBALLY** set true after the \nobreak and **globally** set false by the next invocation of \everypar.

Commands that reset \everypar should globally set it false if appropriate.

```

121 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
122 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
123 \@nobreakfalse

```

(End of definition for \if@nobreak.)

\@savsk Registers used to save the space factor and last skip.

```

\@savsf
124 \newdimen\@savsk
125 \newcount\@savsf

```

(End of definition for \@savsk and \@savsf.)

`\@bsphack` `\@bsphack` and `\@esphack` used by macros such as `\index` and `\begin{@float} ... \end{@float}` that want to be invisible — i.e., not leave any extra space when used in the middle of text. Such a macro should begin with `\@bsphack` and end with `\@esphack`. The macro in question should not create any text, nor change the mode.

Before giving the current definition we give an extended definition that is currently not used (because it doesn't work as advertised:-)

These are generalized hacks which attempt to do sensible things when 'invisible commands' appear in vmode too.

They need to cope with space in both hmode (plus spacefactor) and vmode, and also cope with breaks etc. In vmode this means ensuring that any following `\addvspace`, etc sees the correct glue in `\lastskip`.

In fact, these improved versions should be used for other cases of 'whatsits, thingies etc' which should be invisible. They are only for commands, not environments (see notes on `\@Esphack`).

BTW, anyone know why the standard hacks are surrounded by `\ifmmode\else` rather than simply `\ifhmode`?

And are there any cases where saving the spacefactor is essential? I have some extensions where it is, but it does not appear to be so in the standard uses.

```
def \@bsphack{%
  \relax \ifvmode
    \@savsk \lastskip
    \ifdim \lastskip=\z@
    \else
      \vskip -\lastskip
    \fi
  \else
    \ifhmode
      \@savsk \lastskip
      \@savsf \spacefactor
    \fi
  \fi
}
```

I think that, in vmode, it is the safest to put in a `\nobreak` immediately after such things since writes, inserts etc followed by glue give valid breakpoints and, in general, it is possible to create breaks but impossible to destroy them.

```
def \@esphack{%
  \relax \ifvmode
    \nobreak
    \ifdim \@savsk=\z@
    \else
      \vskip \@savsk
    \fi
  \else
    \ifhmode
      \spacefactor \@savsf
      \ifdim \@savsk>\z@
        \ignorespaces
      \fi
    \fi
  \fi
}
```

```

\fi
\fi

```

For the moment we are going to ignore the vertical versions until they are correct.

```

126 \def\@bsphack{%
127   \relax
128   \ifhmode
129     \@savsk\lastskip
130     \@savsf\spacefactor
131   \fi}

```

(End of definition for \@bsphack.)

\@esphack Companion to \@bsphack. If this command is not properly paired with \@bsphack one might end up with a low-level T_EX error: “BAD spacefactor”. One possible cause is calling \@bsphack in vertical mode, then doing something that gets you (sometimes) into horizontal mode and finally calling \@esphack. Even if no error is generated that is wrong, because \@esphack will then use the saved values for \@savsk and \@savsf from some earlier invocation of \@bsphack which will have nothing to do with the current situation.

```

132 </2ekernel>
133 <latexrelease>\IncludeInRelease{2026/06/01}%
134 <latexrelease>          {\@esphack}{chained space hacks}%
135 <*2ekernel | latexrelease>
136 \def\@esphack{%
137   \relax
138   \ifhmode
139     \spacefactor\@savsf
140     \ifdim\@savsk>\z@
141
142       \ifdim\lastskip=\z@

```

If there is more than one space hack in succession (with a space in between) L^AT_EX still ended up with more than one space because the \z@skip prevented the second space hack from recognizing that no space should be added. To avoid this we add a special “zero” space that looks like a positive space to \lastskip.

There is a side effect though: if the space hack is the last thing in a paragraph the 1sp is dropped and we actually allow the last line to be over by 1sp, but this is better than the extra space introduced in the past.

```

142       \nobreak
143 %       \hskip\z@skip
144       \hskip-1sp\relax\hskip1sp\relax
145       \fi
146       \ignorespaces
147     \fi
148   \else
149     \ifvmode
150       \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
151     \fi

```

```

152 \fi
153 }%
154 </2ekernel | latexrelease>
155 <latexrelease>\EndIncludeInRelease

156 <latexrelease>\IncludeInRelease{2018/10/10}%
157 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
158
159 <latexrelease>\def\@esphack{%
160 <latexrelease> \relax
161 <latexrelease> \ifhmode
162 <latexrelease> \spacefactor\@savsf
163 <latexrelease> \ifdim\@savsk>\z@
164 <latexrelease> \ifdim\lastskip=\z@
165 <latexrelease> \nobreak \hskip\z@skip
166 <latexrelease> \fi
167 <latexrelease> \ignorespaces
168 <latexrelease> \fi
169 <latexrelease> \else
170 <latexrelease> \ifvmode
171 <latexrelease> \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
172 <latexrelease> \fi
173 <latexrelease> \fi}%
174 <latexrelease>\EndIncludeInRelease

175 <latexrelease>\IncludeInRelease{2015/10/01}%
176 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
177 <latexrelease>\def\@esphack{%
178 <latexrelease> \relax
179 <latexrelease> \ifhmode
180 <latexrelease> \spacefactor\@savsf
181 <latexrelease> \ifdim\@savsk>\z@
182 <latexrelease> \ifdim\lastskip=\z@
183 <latexrelease> \nobreak \hskip\z@skip
184 <latexrelease> \fi
185 <latexrelease> \ignorespaces
186 <latexrelease> \fi
187 <latexrelease> \fi}%
188 <latexrelease>\EndIncludeInRelease
189 <latexrelease>\IncludeInRelease{2015/01/01}%
190 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
191 <latexrelease>\def\@esphack{%
192 <latexrelease> \relax
193 <latexrelease> \ifhmode
194 <latexrelease> \spacefactor\@savsf
195 <latexrelease> \ifdim\@savsk>\z@
196 <latexrelease> \nobreak \hskip\z@skip
197 <latexrelease> \ignorespaces
198 <latexrelease> \fi
199 <latexrelease> \fi}%
200 <latexrelease>\EndIncludeInRelease
201 <latexrelease>\IncludeInRelease{0000/00/00}%
202 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
203 <latexrelease>\def\@esphack{%
204 <latexrelease> \relax

```

```

205 <latexrelease> \ifhmode
206 <latexrelease> \spacefactor\@savsf
207 <latexrelease> \ifdim\@savsk>\z@
208 <latexrelease> \ignorespaces
209 <latexrelease> \fi
210 <latexrelease> \fi}%
211 <latexrelease>\EndIncludeInRelease
212 <*2ekernel>

```

(End of definition for \@esphack.)

\@Esphack A variant of \@esphack that sets the @ignore switch to true (as \@esphack used to do previously). This is currently used only for floats and similar environments.

```

213 </2ekernel>
214 <latexrelease>\IncludeInRelease{2015/01/01}%
215 <latexrelease> \{\@Esphack\}{hyphenation after space hack}%
216 <*2ekernel | latexrelease>
217 \def\@Esphack{%
218   \relax
219   \ifhmode
220     \spacefactor\@savsf
221     \ifdim\@savsk>\z@
222       \nobreak \hskip\z@skip
223       \@ignoretrue
224       \ignorespaces
225     \fi
226   \fi}%
227 </2ekernel | latexrelease>
228 <latexrelease>\EndIncludeInRelease
229 <latexrelease>\IncludeInRelease{0000/00/00}%
230 <latexrelease> \{\@Esphack\}{hyphenation after space hack}%
231 <latexrelease>\def\@Esphack{%
232 <latexrelease> \relax
233 <latexrelease> \ifhmode
234 <latexrelease> \spacefactor\@savsf
235 <latexrelease> \ifdim\@savsk>\z@
236 <latexrelease> \ignoretrue
237 <latexrelease> \ignorespaces
238 <latexrelease> \fi
239 <latexrelease> \fi}%
240 <latexrelease>\EndIncludeInRelease
241 <*2ekernel>

```

(End of definition for \@Esphack.)

\@vbsphack Another variant which is useful for invisible things which should not live in vmode (this is how some people feel about marginals).

If it occurs in vmode then it enters hmode and ensures that \@savsk is nonzero so that the \ignorespaces is put in later. It is not used at present.

```

\def \@vbsphack{ %
  \relax \ifvmode
    \leavevmode
    \@savsk 1sp

```

```

        \@savsf \spacefactor
    \else
        \ifhmode
            \@savsk \lastskip
            \@savsf \spacefactor
        \fi
    \fi
}

```

(End of definition for \@vbsphack.)

1.5 Vertical spacing

L^AT_EX supports the plain T_EX commands `\smallskip`, `\medskip` and `\bigskip`. However, it redefines them using `\vspace` instead of `\vskip`.

Extra vertical space is added by the command `\addvspace{⟨skip⟩}`, which adds a vertical skip of `⟨skip⟩` to the document. The sequence `\addvspace{⟨s1⟩} \addvspace{⟨s2⟩}` is equivalent to `\addvspace{⟨maximum of s1, s2⟩}`.

`\addvspace` should be used only in vertical mode, and gives an error if it's not. The `\addvspace` command does *not* add vertical space if `@minipage` is true. The minipage environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the `\addpenalty{⟨penalty⟩}` command. It works properly when `\addpenalty` and `\addvspace` commands are mixed.

The `@nobreak` switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

`\@xaddvskip` Internal macro for `\vspace` handling the case that space has previously been added.

```

242 \def\@xaddvskip{%
243   \ifdim\lastskip<\@tempskipb
244     \vskip-\lastskip
245     \vskip\@tempskipb
246   \else
247     \ifdim\@tempskipb<\z@
248       \ifdim\lastskip<\z@
249         \else
250           \advance\@tempskipb\lastskip
251           \vskip-\lastskip
252           \vskip \@tempskipb
253         \fi
254       \fi
255     \fi}

```

(End of definition for \@xaddvskip.)

`\addvspace` Add vertical space taking into account space already added, as described above.

```

256 </2ekernel>
257 <*2ekernel | latexrelease>
258 <latexrelease> \IncludeInRelease{2024/11/01}%
259 <latexrelease> { \addvspace } { drop unnecessary no-item error }%
260 \protected\def\addvspace#1{%

```

When this is encountered in hmode, we check whether we are in an hbox and if so generate a L^AT_EX error, as otherwise this would cause a bunch of low-level errors. In unrestricted hmode we simply switch to vmode by issuing a `\par`.

```

261 \ifhmode \ifinner \@LRmoderr \else \par \fi \fi
262 \if@minipage\else
263 \ifdim \lastskip =\z@
264 \@vspace@calcify{#1}%
265 \else
266 \setlength\@tempskipb{#1}%
267 \@xaddvskip
268 \fi
269 \fi
270 }
271 </2ekernel | latexrelease>
272 <latexrelease>\EndIncludeInRelease

273 <latexrelease>\IncludeInRelease{2020/10/01}%
274 <latexrelease> \{ \addvspace \} { \addvspace calc support } %
275 <latexrelease>\def\addvspace#1{%
276 <latexrelease> \ifvmode
277 <latexrelease> \if@minipage\else
278 <latexrelease> \ifdim \lastskip =\z@
279 <latexrelease> \@vspace@calcify{#1}%
280 <latexrelease> \else
281 <latexrelease> \setlength\@tempskipb{#1}%
282 <latexrelease> \@xaddvskip
283 <latexrelease> \fi
284 <latexrelease> \fi
285 <latexrelease> \else
286 <latexrelease> \@noitemerr
287 <latexrelease> \fi}
288 <latexrelease>\EndIncludeInRelease

289 <latexrelease>\IncludeInRelease{0000/00/00}%
290 <latexrelease> \{ \addvspace \} { \addvspace calc support } %
291 <latexrelease>
292 <latexrelease>\def\addvspace#1{%
293 <latexrelease> \ifvmode
294 <latexrelease> \if@minipage\else
295 <latexrelease> \ifdim \lastskip =\z@
296 <latexrelease> \vskip #1\relax
297 <latexrelease> \else
298 <latexrelease> \@tempskipb#1\relax
299 <latexrelease> \@xaddvskip
300 <latexrelease> \fi
301 <latexrelease> \fi
302 <latexrelease> \else
303 <latexrelease> \@noitemerr
304 <latexrelease> \fi}
305 <latexrelease>\EndIncludeInRelease
306 <*2ekernel>

```

(End of definition for \addvspace.)

`\addpenalty`

```
307 </2ekernel>
308 <latexrelease>\IncludeInRelease{2024/11/01}%
309 <latexrelease>          {\addpenalty}{\addpenalty drop error}%
310 <*2ekernel | latexrelease>
```

```
311 \protected\def\addpenalty#1{%
```

See description of `\addvspace` for documentation of the next line of code.

```
312   \ifhmode \ifinner \@LRmoderr \else \par \fi \fi
```

Fix provided by Donald (though the original fix was not good enough). In 2005 Plamen Tanovski discovered that this fix wasn't good enough either as the `\vskip` kept getting bigger if several `\addpenalty` commands followed each other. Donald kindly send a new fix.

```
313   \if@minipage
314   \else
315     \if@nobreak
316     \else
317       \ifdim\lastskip=\z@
318       \penalty#1\relax
319     \else
320       \@tempskipb\lastskip
```

We have to make sure the final `\vskip` seen by T_EX is the correct one, namely `\@tempskipb`. However, we may have to adjust for `\prevdepth` when placing the penalty; that should not affect the skip we pass on to T_EX.

```
321     \begingroup
322     \@tempskipa\@tempskipb
323     \advance \@tempskipb
324     \ifdim\prevdepth>\maxdepth\maxdepth\else
```

If `\prevdepth` is -1000pt due to `\nointerlineskip` we better not add it!

```
325       \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
326       \fi
327       \vskip -\@tempskipb
328       \penalty#1%
329       \ifdim\@tempskipa=\@tempskipb
```

Do nothing if the `\prevdepth` check made no adjustment.

```
330     \else
```

Combine the `\prevdepth` adjustment into a single skip.

```
331       \advance\@tempskipb -\@tempskipa
332       \vskip \@tempskipb
333     \fi
```

The final skip is always the specified length.

```
334       \vskip \@tempskipa
335     \endgroup
336   \fi
337 \fi
338 \fi
339 }
340 </2ekernel | latexrelease>
341 <latexrelease>\EndIncludeInRelease
```

```

342 <latexrelease>\IncludeInRelease{2015/01/01}%
343 <latexrelease>          {\addpenalty}{\addpenalty}%
344 <latexrelease>\def\addpenalty#1{%
345 <latexrelease>  \ifvmode
346 <latexrelease>    \if@minipage
347 <latexrelease>    \else
348 <latexrelease>      \if@nobreak
349 <latexrelease>      \else
350 <latexrelease>        \ifdim\lastskip=\z@
351 <latexrelease>        \penalty#1\relax
352 <latexrelease>        \else
353 <latexrelease>          \@tempskipb\lastskip
354 <latexrelease>          \begingroup
355 <latexrelease>            \@tempskipa\@tempskipb
356 <latexrelease>            \advance \@tempskipb
357 <latexrelease>              \ifdim\prevdepth>\maxdepth\maxdepth\else
358 <latexrelease>                \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
359 <latexrelease>              \fi
360 <latexrelease>            \vskip -\@tempskipb
361 <latexrelease>            \penalty#1%
362 <latexrelease>            \ifdim\@tempskipa=\@tempskipb
363 <latexrelease>            \else
364 <latexrelease>              \advance\@tempskipb -\@tempskipa
365 <latexrelease>              \vskip \@tempskipb
366 <latexrelease>            \fi
367 <latexrelease>            \vskip \@tempskipa
368 <latexrelease>          \endgroup
369 <latexrelease>        \fi
370 <latexrelease>      \fi
371 <latexrelease>    \fi
372 <latexrelease>  \else
373 <latexrelease>    \@noitemerr
374 <latexrelease>  \fi}%

375 <latexrelease>\EndIncludeInRelease
376 <latexrelease>\IncludeInRelease{0000/00/00}%
377 <latexrelease>          {\addpenalty}{\addpenalty}%
378 <latexrelease>\def\addpenalty#1{%
379 <latexrelease>  \ifvmode
380 <latexrelease>    \if@minipage
381 <latexrelease>    \else
382 <latexrelease>      \if@nobreak
383 <latexrelease>      \else
384 <latexrelease>        \ifdim\lastskip=\z@
385 <latexrelease>        \penalty#1\relax
386 <latexrelease>        \else
387 <latexrelease>          \@tempskipb\lastskip
388 <latexrelease>          \vskip -\lastskip
389 <latexrelease>          \penalty#1%
390 <latexrelease>          \vskip\@tempskipb
391 <latexrelease>        \fi
392 <latexrelease>      \fi
393 <latexrelease>    \fi
394 <latexrelease>  \else
395 <latexrelease>    \@noitemerr

```



```

396 <latexrelease> \fi}%
397 <latexrelease>\EndIncludeInRelease
398 <*2ekernel>

```

(End of definition for \addpenalty.)

\vspace
\@vspace
\@vspacer

The new code for these commands depends on the following facts:

- The value of prevdepth is changed only when a box or rule is created and added to a vertical list;
- The value of prevdepth is used only when a box is created and added to a vertical list;
- The value of prevdepth is always local to the building of one vertical list.

```

399 \DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}
400 </2ekernel>
401 <*2ekernel | latexrelease>
402 <latexrelease>\IncludeInRelease{2020/10/01}%
403 <latexrelease> \{\@vspace\}{Support calc in \vspace}%

```

We support calc syntax in the argument and therefore use \setlength.

```

404 \def\@vspace #1{%
405   \ifvmode
406     \@vspace@calcify{#1}%
407     \vskip\z@skip
408   \else
409     \@bsphack
410     \vadjust{\@restorepar
411               \@vspace@calcify{#1}%
412               \vskip\z@skip
413             }%
414     \@esphack
415   \fi}
416 \def\@vspacer#1{%
417   \ifvmode
418     \dimen@\prevdepth
419     \hrule \@height\z@
420     \nobreak
421     \@vspace@calcify{#1}%
422     \vskip\z@skip
423     \prevdepth\dimen@
424   \else
425     \@bsphack
426     \vadjust{\@restorepar
427               \hrule \@height\z@
428               \nobreak
429               \@vspace@calcify{#1}%
430               \vskip\z@skip}%
431     \@esphack
432   \fi}
433 </2ekernel | latexrelease>
434 <latexrelease>\EndIncludeInRelease
435 <latexrelease>\IncludeInRelease{0000/00/00}%

```

```

436 <latexrelease>                {\@vspace}{Support calc in \vspace}%
437 <latexrelease>
438 <latexrelease>\def\@vspace #1{%
439 <latexrelease>  \ifvmode
440 <latexrelease>    \vskip #1
441 <latexrelease>    \vskip\z@skip
442 <latexrelease>  \else
443 <latexrelease>    \@bsphack
444 <latexrelease>    \vadjust{\@restorepar
445 <latexrelease>                \vskip #1
446 <latexrelease>                \vskip\z@skip
447 <latexrelease>            }%
448 <latexrelease>    \@esphack
449 <latexrelease>  \fi}
450 <latexrelease>\def\@vspacer#1{%
451 <latexrelease>  \ifvmode
452 <latexrelease>    \dimen@\prevdepth
453 <latexrelease>    \hrule \@height\z@
454 <latexrelease>    \nobreak
455 <latexrelease>    \vskip #1
456 <latexrelease>    \vskip\z@skip
457 <latexrelease>    \prevdepth\dimen@
458 <latexrelease>  \else
459 <latexrelease>    \@bsphack
460 <latexrelease>    \vadjust{\@restorepar
461 <latexrelease>                \hrule \@height\z@
462 <latexrelease>                \nobreak
463 <latexrelease>                \vskip #1
464 <latexrelease>                \vskip\z@skip}%
465 <latexrelease>    \@esphack
466 <latexrelease>  \fi}
467 <latexrelease>\EndIncludeInRelease
468 <*2ekernel>

```

(End of definition for \vspace, \@vspace, and \@vspacer.)

```

\smallskip
\medskip 469 \def\smallskip{\vspace\smallskipamount}
\bigskip 470 \def\medskip{\vspace\medskipamount}
         471 \def\bigskip{\vspace\bigskipamount}

```

(End of definition for \smallskip, \medskip, and \bigskip.)

```

\smallskipamount
\medskipamount 472 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 473 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
               474 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

```

(End of definition for \smallskipamount, \medskipamount, and \bigskipamount.)

1.6 Horizontal space (and breaks)

`\nobreakdashes` This idea is borrowed from the `amsmath` package but here we define a robust command. This command is a low-level command designed for use only before hyphens or dashes (such as `-`, `--`, or `---`).

It could probably be better implemented: it may need its own private token register and temporary command.

Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

Note that even if it is not followed by a ‘-’, it still leaves vmode and sets the spacefactor; so use it carefully!

```

475 \DeclareRobustCommand{\nobreakdashes}{%
476   \leavevmode
477   \toks@{}%
478   \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
479                     \futurelet\@let@token \reserved@b}%
480   \def\reserved@b   {\ifx\@let@token -%
481                     \expandafter\reserved@a
482                     \else
483                     \setbox\z@ \hbox{\the\toks@\nobreak}%
484                     \unhbox\z@
485                     \spacefactor\sfcode'\-
486                     \fi}%
487   \futurelet\@let@token \reserved@b
488 }

```

(End of definition for `\nobreakdashes`.)

`\nobreakspace` This is a robust command that produces a horizontal space at which, in paragraph-mode, a line-break is not possible. We then define an active `~` to expand to it since this is the documented behaviour of `~`. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the L^AT_EX internal commands.

The braces in the definition of `~` are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep `\@xobeysp` as it is widely used; so here it is let to the non-robust command `\nobreakspace`.

The fragile version of `~` needs a brace group after `\nobreakspace` to prevent loss of spaces if it occurs in an expansion context. That’s not an issue with the updated `\protected` definition, so we keep the code shorter and avoid that.

```

489 \DeclareRobustCommand{\nobreakspace}{%
490   \leavevmode\nobreak\ }
491 \catcode '\~ =13
492 \</2ekernel>
493 \<latexrelease>\IncludeInRelease{2023/11/01}%
494 \<latexrelease>          {\tilde}{Protected tilde}%
495 \*2ekernel | latexrelease>
496 \protected\edef~{%
497   \noexpand\ifincsname\noexpand\expandafter\string~%
498   \noexpand\else
499     \noexpand\expandafter\noexpand\nobreakspace
500   \noexpand\fi
501 }
502 \</2ekernel | latexrelease>
503 \<latexrelease>\EndIncludeInRelease
504 \<latexrelease>\IncludeInRelease{0000/00/00}%

```

```

505 <latexrelease> \tildel{Protected tilde}%
506 <latexrelease>\def~{\nobreakspace{}}
507 <latexrelease>\EndIncludeInRelease
508 <*2ekernel>
509 \expandafter\let\expandafter\xobeysp\csname nobreakspace \endcsname

```

(End of definition for \nobreakspace and \xobeysp.)

\xobeytab Equivalent to the space case with the default settings.

```

510 </2ekernel>
511 <latexrelease>\IncludeInRelease{2023/11/01}%
512 <latexrelease> \xobeytab{Obeyed tabs}%
513 <*2ekernel | latexrelease>
514 \let\xobeytab\xobeysp
515 </2ekernel | latexrelease>
516 <latexrelease>\EndIncludeInRelease
517 <latexrelease>\IncludeInRelease{0000/00/00}%
518 <latexrelease> \xobeytab{Obeyed tabs}%
519 <latexrelease>\let\xobeytab\undefined
520 <latexrelease>\EndIncludeInRelease
521 <*2ekernel>

```

(End of definition for \xobeytab.)

\@ Placed before a '.', makes it a sentence-ending period. Does the right thing for other punctuation marks as well. Does this by setting spacefactor to 1000.

```

522 </2ekernel>
523 <latexrelease>\IncludeInRelease{2015/01/01}%
524 <latexrelease> \@{Space after \@}%
525 <*2ekernel | latexrelease>
526 \def\@{\spacefactor\@m}%
527 </2ekernel | latexrelease>
528 <latexrelease>\EndIncludeInRelease
529 <latexrelease>\IncludeInRelease{0000/00/00}%
530 <latexrelease> \@{Space after \@}%
531 <latexrelease>\def\@{\spacefactor\@m}%
532 <latexrelease>\EndIncludeInRelease
533 <*2ekernel>

```

(End of definition for \@.)

\hspace

```

534 \DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}

```

(End of definition for \hspace.)

\@hspace

```

535 </2ekernel>
536 <*2ekernel | latexrelease>
537 <latexrelease>\IncludeInRelease{2020/10/01}%
538 <latexrelease> \@hspace{Support calc with \hspace}%

```

We use a private register to calculate the space (if calc is used). Previously we used a group but that results in `\everypar` etc. being executed inside the group if the `\hspace` starts a paragraph. This is a bug fix so we do not provide rollback to the incorrect intermediate version.

```

539 \newskip\sp@ce@skip
540 \def\@hspace#1{\setlength\sp@ce@skip{#1}\hskip\sp@ce@skip}
541 \</2ekernel|latexrelease>
542 \<latexrelease>\EndIncludeInRelease

543 \<latexrelease>\IncludeInRelease{0000/00/00}%
544 \<latexrelease>          {\@hspace}{Support calc with \hspace}%
545
546 \<latexrelease>
547 \<latexrelease>\def\@hspace#1{\hskip #1\relax}
548 \<latexrelease>\EndIncludeInRelease
549 \<*2ekernel>

```

(End of definition for \@hspace.)

\@hspacer Extra `\hskip` Opt added 1985/17/12 to guard against a following `\unskip \relax` added 13 Oct 88 for usual T_EX lossage replaced both changes by `\hskip\z@skip` 27 Nov 91

```

550 \def\@hspacer#1{\vrule \width\z@\nobreak
551          \@hspace{#1}\hskip \z@skip}

```

(End of definition for \@hspacer.)

\fill

```

552 \newskip\fill
553 \fill = Opt plus 1fill

```

(End of definition for \fill.)

\stretch

```

554 \def\stretch#1{\z@ \@plus #1fill\relax}

```

(End of definition for \stretch.)

```

555 \</2ekernel>
556 \<*2ekernel|latexrelease>
557 \<latexrelease>\IncludeInRelease{2018/12/01}%
558 \<latexrelease>          {\thinspace}{Start LR-mode}%

```

\enspace

```

559 \DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }

```

(End of definition for \enspace.)

\leavevmode@ifvmode Leave vmode but only if we are really in vmode, otherwise the expansion is empty (which is not the case with the default definition).

```

560 \protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}

```

```

(End of definition for \leavevmode@ifvmode.)

561 </2ekernel | latexrelease>
562 <latexrelease>\EndIncludeInRelease
563 <latexrelease>\IncludeInRelease{0000/00/00}%
564 <latexrelease>{\thinspace}{Start LR-mode}%
565 <latexrelease>\def\thinspace{\kern .16667em }
566 <latexrelease>\def\negthinspace{\kern-.16667em }
567 <latexrelease>\def\enspace{\kern.5em }
568 <latexrelease>\let\leavevmode@ifvmode\@undefined
569 <latexrelease>\EndIncludeInRelease
570 <*2ekernel>

\enskip
\quad 571 \def\enskip{\hskip.5em\relax}
\qquad 572 \def\quad{\hskip1em\relax}
573 \def\qquad{\hskip2em\relax}

(End of definition for \enskip, \quad, and \qquad.)
For Unicode engines, make the Unicode soft hyphen an active character defined as
\-.

574 \ifx\Umathcode\@undefined\else
575 \catcode "AD=13
576 \def^^ad{\-}
577 \fi

\obeycr The following definitions will probably get deleted or moved to compatibility mode soon.
\restorecr
578 {\catcode'\^M=13 \gdef\obeycr{\catcode'\^M13 \def^^M{\\relax}%
579 \@gobblecr}%
580 {\catcode'\^M=13 \gdef\@gobblecr{\@ifnextchar
581 \@gobble\ignorespaces}}
582 \gdef\restorecr{\catcode'\^M5 }}

(End of definition for \obeycr and \restorecr.)

583 </2ekernel>

```

File 19

ltlogos.dtx

1 Logos

Various logos are defined here.

`\TeX` The $\TeX$ logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 <*/2kernel>
2 \DeclareRobustCommand\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(End of definition for \TeX.)

`\LaTeX` The $\LaTeX$ logo.

```
3 \DeclareRobustCommand{\LaTeX}{L\kern-.36em%
4   {\sbox\z@ T%
5     \vbox to\ht\z@{\hbox{\check@mathfonts
6                           \fontsize\sf@size\z@
7                           \math@fontsfalse\selectfont
8                           A}%
9                           \vss}%
10  }%
11  \kern-.15em%
12  \TeX}
```

(End of definition for \LaTeX.)

`\LaTeXe` The $\LaTeX 2_{\epsilon}$ logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
14   \if b\expandafter\@car\@series\@nil\boldmath\fi
15   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
16 </2kernel>
```

(End of definition for \LaTeXe.)

File 20

ltfiles.dtx

1 File Handling

The following user commands are defined in this part:

<code>\document</code>	<code>(ie \begin{document})</code>
	Reads in the .AUX files and <code>\catcode</code> 's @ to 12.
<code>\nofiles</code>	
	Suppresses all file output by setting <code>\@files</code> false.
<code>\includeonly</code>	<code>{\NAME1, ... ,NAMEn}</code>
	Causes only parts NAME1, ... ,NAMEn to be read by their <code>\include</code> commands. Works by setting <code>partsw</code> true and setting <code>\@partlist</code> to NAME1, ... ,NAMEn.
<code>\include</code>	<code>{\NAME}</code>
	Does an <code>\input</code> NAME unless <code>\@partsw</code> is true and NAME is not in <code>\@partlist</code> . If <code>\@files</code> is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.
<code>\input</code>	<code>{\NAME}</code>
	The same as TeX's <code>\input</code> , except it allows optional braces around the file name. In $\text{\LaTeX 2}_{\epsilon}$, it also avoids the primitive 'missing file' error, if the file can not be found.
<code>\IfFileExists</code>	<code>{\NAME}{\then}{\else}</code>
	If the file exists on the system, execute <i>then</i> otherwise execute <i>else</i> .
<code>\InputIfFileExists</code>	<code>{\NAME}{\then}{\else}</code>
	If the file exists on the system, execute <i>then</i> and input NAME otherwise execute <i>else</i> . <i>Historical $\text{\LaTeX 2.09}$ comments (not necessarily accurate any more):</i>

```
1 <*2kernel>
2 \message{files,}
```

VARIABLES, SWITCHES AND INTERNAL COMMANDS:

<code>\@mainaux</code>	: Output file number for main .AUX file.
<code>\@partaux</code>	: Output file number for current part's .AUX file.
<code>\@auxout</code>	: Either <code>\@mainout</code> or <code>\@partout</code> , depending on which .AUX file output goes to.
<code>\@input{foo}</code>	: If file foo exists, then <code>\input</code> 's it, otherwise types a warning message.
<code>@files</code>	: Switch – set false if no .AUX, .TOC, .IDX etc files are to be written
<code>@partsw</code>	: Set true by a <code>\includeonly</code> command.
<code>\@partlist</code>	: Set to the argument of the <code>\includeonly</code> command.
<code>\cp@FOO</code>	: The checkpoint for <code>\include</code> 'd file FOO.TEX, written by <code>\@writeckpt</code> at the end of file FOO.AUX

```
\includeonly{FILELIST} ==
BEGIN
```



```

\@partsw := T
\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
\clearpage
if \@filesw = T
then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
fi
if \@partsw = T
then \@tempswa := F
\reserved@a := FILE
for \reserved@a := \@partlist
do if eval(\reserved@a) = eval(\reserved@b)
then \@tempswa := T fi
od
fi

if \@tempswa = T
then \@auxout := \@partaux
if \@filesw = T
then \immediate\openout\@partaux{FILE.AUX}
\immediate\write\@partaux{\relax}
fi
\@input{FILE.TEX}
\clearpage
\@writeckpt{FILE}
if @filesw then \closeout \@partaux fi
\@auxout := \@mainaux
else \cp@FILE
fi
END

\@writeckpt{FILE} ==
BEGIN
if \@filesw = T
\immediate\write on file \@partaux:
\@setckpt{FILE}{
%% }
for \reserved@a := \cl@ckpt
do \immediate\write on file \@partaux:
\global\string\setcounter
{eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
od
%% {
\immediate\write on file \@partaux: }
fi
END

\@setckpt{FILE}{LIST} ==
BEGIN

```

```
G \cp@FILE := LIST
END
```

```
INITIALIZATION
\@tempswa := T
```

End of historical L^AT_EX 2.09 comments.

```
\@mainaux
\@partaux 3 \newwrite\@mainaux
           4 \newwrite\@partaux
```

(End of definition for \@mainaux and \@partaux.)

```
\if@files
\if@partsw 5 \newif\if@files \@filestrue
           6 \newif\if@partsw \@partswfalse
```

(End of definition for \if@files and \if@partsw.)

\@clubpenalty This stores the current normal (non-infinite) value of `\clubpenalty`; it should therefore be reset whenever the normal value is changed (as in the bibliography in the standard styles).

```
7 \newcount\@clubpenalty
8 \@clubpenalty \clubpenalty
```

(End of definition for \@clubpenalty.)

```
\document
```

```
9 \</2ekernel>
10 \<latexrelease>\IncludeInRelease{2020/10/01}%
11 \<latexrelease> {\document}{Added hook to load l3backend code}%
12 \<*2ekernel | latexrelease>
13 \def\document{%
```

We do cancel the grouping as part of the `\begin` handling (this is now done inside `\begin` instead) so that the `env/<env>/begin` hook is not hidden inside `\begingroup ... \endgroup`.

```
14 % \endgroup
15 \UseOneTimeHook{begindocument/before}%
16 \<kernel@after@begindocument@before
```

Added hook to load l3backend code:

```
17 \<expl@sys@load@backend@@
18 \ifx\@unusedoptionlist\@empty\else
19 \<latex@warning@no@line{Unused global option(s):^^J%
20 \spaces[\@unusedoptionlist]}%
21 \fi
22 \<colht\textheight
23 \<colroom\textheight \vsize\textheight
24 \<columnwidth\textwidth
25 \<clubpenalty\clubpenalty
26 \if@twocolumn
27 \advance\columnwidth -\columnsep
```

```

28     \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
29 \fi
30 \hsize\columnwidth \linewidth\hsize
31 \begingroup\@floatplacement\@dblfloatplacement
32     \makeatletter\let\@writefile\@gobbletwo
33
34     \global \let \@multiplelabels \relax
35     \input{\jobname.aux}%
36 \endgroup
37 \if@files
38     \immediate\openout\@mainaux\jobname.aux
39     \immediate\write\@mainaux{\relax}%
40 \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old lfonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

40 \process@table
41 \let\glb@currsize\@empty % Force math initialization.
42
43 \normalsize
44 \everypar{}%

```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L^AT_EX 2.0x and plain T_EX.)

```

44 \ifx\normalsfcodes\@empty
45     \ifnum\sfcode'\.=\@m
46         \let\normalsfcodes\frenchspacing
47     \else
48         \let\normalsfcodes\nonfrenchspacing
49 \fi
50 \fi

```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```

51 \ifx\document@default@language\m@ne
52     \chardef\document@default@language\language
53 \fi

```

Way back in 1991 (08/26) FMi & RmS set the `@noskipsec` switch to true in the preamble and to false here. This was done to trap lists and related text in the preamble but it does not catch everything; hence Change 1.1g was introduced.

```

54 \noskipsecfalse
55
56 \let \@refundefined \relax

```

Just before disabling the preamble commands we execute the begin document hook which contains any code contributed by `\AtBeginDocument`. Also disable the gathering of the file list, if no `\listfiles` has been issued. `\AtBeginDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

56 \@kernel@before@begindocument
57 \UseOneTimeHook{begindocument}%
58 \@kernel@after@begindocument

```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, i.e. starts are group in `\begin{document}`.

Since a value of exactly 0pt for `\topskip` causes `\twocolumn[]` to misbehave, we add this check, hoping that it will not cause any problems elsewhere.

```

59 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
60 \global\@maxdepth\maxdepth
61 \global\let\@begindocumenthook\@undefined
62 \ifx\@listfiles\@undefined
63   \global\let\@filelist\relax
64   \global\let\@addtofilelist\@gobble
65 \fi

```

At the very end we disable all preamble commands. This has to happen after the begin document hooks was executed so that this hook can still use such commands.

```

66 \gdef\do##1{\global\let ##1\@notprerr}%
67 \@preamblecmds

```

The next line saves tokens and also allows `\@nodocument` to be used directly to trap preamble errors.

```

68 \global\let \@nodocument \relax

```

The next line is a pure safety measure in case a do list is ever expanded at the wrong place. In addition it will save a few tokens to get rid of the above definition.

```

69 \global\let\do\noexpand
70 \UseOneTimeHook{begindocument/end}%

```

Use of the hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```

71 \ignorespaces}

```

Provide a global definition for `\do` as well, so that it is already defined in the preamble and not late as `\begin{document}` overwriting some definition given by the unsuspecting user in the preamble.

```

72 \let\do\noexpand

```

The `begindocument` hook already existed in the kernel since 1994 under the name `\atbegindocumenthook` the additional ones are originally from the `etoolbox` package under the names `\@endpreamblehook` `\afterpreamble`.

```

73 \NewHook{begindocument}
74 \NewHook{begindocument/before}
75 \NewHook{begindocument/end}

```

Above we used two kernel only hooks to be run after the public `begindocument/before` and after `begindocument` hooks.

```

\@kernel@after@begindocument@before
\@kernel@before@begindocument
\@kernel@after@begindocument

```

In `\@kernel@after@begindocument@before` we already place one action: drop the fast execution code for the `env/document/begin` hook. That hook marks the end of the preamble and should therefore only be run once. In a normal document that is anyway the case (so the code would just sit there taking up space afterwards, which these days is rather harmless), however, in more complicated scenarios where several full documents are combined to a single document it might get applied several times with harmful effects.

We therefore explicitly drop it at this point. the coding is somewhat obscure due to the name of the macro which requires constructing.

```

76 \edef \@kernel@after@begindocument@before {%
77   \let\expandafter\noexpand\cename
78   __hook env/document/begin\endcename
79   \noexpand\@empty}

```

These internal hooks are already declared earlier (in `ltxpl`) so that other modules could write to them.

```

80 %\let \@kernel@before@begindocument \@empty
81 %\let \@kernel@after@begindocument \@empty

82 </2kernel | latexrelease>
83 <latexrelease>\EndIncludeInRelease

84 <latexrelease>\IncludeInRelease{2017/04/15}%
85 <latexrelease> {\document}{Save language for hyphenation}%
86 <latexrelease>
87 <latexrelease>\def\document{\endgroup
88 <latexrelease> \ifx\@unusedoptionlist\@empty\else
89 <latexrelease> \@latex@warning@no@line{Unused global option(s):^^J%
90 <latexrelease> \spaces[\@unusedoptionlist]}}%
91 <latexrelease> \fi
92 <latexrelease> \@colht\textheight
93 <latexrelease> \@colroom\textheight \vsize\textheight
94 <latexrelease> \columnwidth\textwidth
95 <latexrelease> \@clubpenalty\clubpenalty
96 <latexrelease> \if@twocolumn
97 <latexrelease> \advance\columnwidth -\columnsep
98 <latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
99 <latexrelease> \fi
100 <latexrelease> \hsize\columnwidth \linewidth\hsize
101 <latexrelease> \begingroup\@floatplacement\@dblfloatplacement
102 <latexrelease> \makeatletter\let\@writefile\@gobbletwo
103 <latexrelease> \global \let \@multiplelabels \relax
104 <latexrelease> \@input{\jobname.aux}%
105 <latexrelease> \endgroup
106 <latexrelease> \if@filesw
107 <latexrelease> \immediate\openout\@mainaux\jobname.aux
108 <latexrelease> \immediate\write\@mainaux{\relax}%
109 <latexrelease> \fi
110 <latexrelease> \process@table
111 <latexrelease> \let\glb@currsize\@empty % Force math initialization.
112 <latexrelease> \normalsize
113 <latexrelease> \everypar{}%
114 <latexrelease> \ifx\normalsfcodes\@empty
115 <latexrelease> \ifnum\sfcode'\.=\@m
116 <latexrelease> \let\normalsfcodes\frenchspacing
117 <latexrelease> \else
118 <latexrelease> \let\normalsfcodes\nonfrenchspacing
119 <latexrelease> \fi
120 <latexrelease> \fi
121 <latexrelease> \ifx\document@default@language@m@ne
122 <latexrelease> \chardef\document@default@language\language
123 <latexrelease> \fi

```

```

124 <latexrelease> \noskipsecfalse
125 <latexrelease> \let \@refundefined \relax
126 <latexrelease> \let \AtBeginDocument \@firstofone
127 <latexrelease> \@begindocumenthook
128 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
129 <latexrelease> \global\@maxdepth\maxdepth
130 <latexrelease> \global\let\@begindocumenthook\@undefined
131 <latexrelease> \ifx\@listfiles\@undefined
132 <latexrelease> \global\let\@filelist\relax
133 <latexrelease> \global\let\@addtofilelist\@gobble
134 <latexrelease> \fi
135 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
136 <latexrelease> \@preamblecmds
137 <latexrelease> \global\let \@nodocument \relax
138 <latexrelease> \global\let\do\noexpand
139 <latexrelease> \ignorespaces}
140 <latexrelease> \EndIncludeInRelease
141 <latexrelease>
142 <latexrelease> \IncludeInRelease{0000/00/00}%
143 <latexrelease> {\document}{Save language for hyphenation}
144 <latexrelease> \def\document{\endgroup
145 <latexrelease> \ifx\@unusedoptionlist\@empty\else
146 <latexrelease> \@latex@warning@no@line{Unused global option(s):^^J%
147 <latexrelease> \@spaces[\@unusedoptionlist]}}%
148 <latexrelease> \fi
149 <latexrelease> \@colht\textheight
150 <latexrelease> \@colroom\textheight \vsize\textheight
151 <latexrelease> \columnwidth\textwidth
152 <latexrelease> \@clubpenalty\clubpenalty
153 <latexrelease> \if@twocolumn
154 <latexrelease> \advance\columnwidth -\columnsep
155 <latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth
156 <latexrelease> \@firstcolumntrue
157 <latexrelease> \fi
158 <latexrelease> \hsize\columnwidth \linewidth\hsize
159 <latexrelease> \begingroup\@floatplacement\@dblfloatplacement
160 <latexrelease> \makeatletter\let\@writefile\@gobbletwo
161 <latexrelease> \global \let \@multiplelabels \relax
162 <latexrelease> \@input{\jobname.aux}%
163 <latexrelease> \endgroup
164 <latexrelease> \if@filesw
165 <latexrelease> \immediate\openout\@mainaux\jobname.aux
166 <latexrelease> \immediate\write\@mainaux{\relax}%
167 <latexrelease> \fi
168 <latexrelease> \process@table
169 <latexrelease> \let\glb@currsz\@empty
170 <latexrelease> \normalsize
171 <latexrelease> \everypar{}%
172 <latexrelease> \ifx\normalsfcodes\@empty
173 <latexrelease> \ifnum\sfcode'\.=\@m
174 <latexrelease> \let\normalsfcodes\frenchspacing
175 <latexrelease> \else
176 <latexrelease> \let\normalsfcodes\nonfrenchspacing
177 <latexrelease> \fi

```

```

178 <latexrelease> \fi
179 <latexrelease> \@noskipsecfalse
180 <latexrelease> \let \@refundefined \relax
181 <latexrelease> \let \@AtBeginDocument \@firstofone
182 <latexrelease> \@begindocumenthook
183 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
184 <latexrelease> \global\@maxdepth\maxdepth
185 <latexrelease> \global\let \@begindocumenthook \@undefined
186 <latexrelease> \ifx \@listfiles \@undefined
187 <latexrelease> \global\let \@filelist \relax
188 <latexrelease> \global\let \@addtofilelist \@gobble
189 <latexrelease> \fi
190 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
191 <latexrelease> \@preamblecmds
192 <latexrelease> \global\let \@nodocument \relax
193 <latexrelease> \global\let \do\noexpand
194 <latexrelease> \ignorespaces}
195 <latexrelease> \EndIncludeInRelease
196 <*2ekernel>
197 \@onlypreamble\document

```

(End of definition for \document and others.)

\normalsfcodes The setting of `\@empty` is just a flag. This command may be defined in a class or package file. If it is still `\@empty` at `\begin{document}` it will be defined to be `\frenchspacing` or `\nonfrenchspacing`, depending on which of those appears to be in effect at that point.

```
198 \let\normalsfcodes\@empty
```

(End of definition for \normalsfcodes.)

\nofiles Set `\@fileswfalse` which suppresses the places where L^AT_EX makes `\immediate` writes. The `\makeindex` and `\makeglossary` are disabled. `\protected@write` is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a *<whatsit>* node is still created, and so spacing is not affected by the `\nofiles` command; to ensure this more generally, the `\if@nobreak` test is needed.

```

199 \def\nofiles{%
200   \@fileswfalse
201   \typeout{No auxiliary output files.^^J}%
202   \long\def\protected@write##1##2##3%
203     {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
204   \let\makeindex\relax
205   \let\makeglossary\relax}
206 \@onlypreamble\nofiles

```

(End of definition for \nofiles.)

\protected@write This takes three arguments: an output stream, some initialization code, and some text to write. It then writes this, with appropriate handling of `\protect` and `\thepage`.

```

207 </2ekernel>
208 <*2ekernel | latexrelease>
209 <latexrelease> \IncludeInRelease{2025/06/01}%
210 <latexrelease> \def\protected@write#1#2#3%
211 \long\def \protected@write#1#2#3%

```

```

212 \begingroup
213 \let\thepage\relax
214 #2%
215 \let\protect\@unexpandable@protect
216 \edef\reserved@a{\endgroup\write#1{#3}}%
217 \reserved@a
218 \if@nobreak\ifvmode\nobreak\fi\fi
219 }
220 </2ekernel | latexrelease>
221 <latexrelease>\EndIncludeInRelease
222 <latexrelease>\IncludeInRelease{0000/00/00}%
223 <latexrelease> \protected@write}{Allow for immediate write}%
224 <latexrelease>\long\def \protected@write#1#2#3{%
225 <latexrelease> \begingroup
226 <latexrelease> \let\thepage\relax
227 <latexrelease> #2%
228 <latexrelease> \let\protect\@unexpandable@protect
229 <latexrelease> \edef\reserved@a{\write#1{#3}}%
230 <latexrelease> \reserved@a
231 <latexrelease> \endgroup
232 <latexrelease> \if@nobreak\ifvmode\nobreak\fi\fi
233 <latexrelease>}
234 <latexrelease>\EndIncludeInRelease
235 <*2ekernel>

```

(End of definition for \protected@write.)

```

236 \let\@auxout=\@mainaux

```

\include In the definition of **\include**, **\def\reserved@b** changed to **\edef\reserved@b** to be consistent with the **\edef** in **\includeonly**. (Suggested by Rainer Schöpf & Frank Mittelbach. Change made 20 Jul 88.)

\includeonly Changed definition of **\include** to allow space at end of file name — otherwise, typing **\include{foo }** would cause L^AT_EX to overwrite **foo.tex**. Change made 24 May 89, suggested by Rainer Schöpf and Frank Mittelbach

Made **\include** check for being used inside an **\include**'d file, as this will not work and cause surprising results.

```

237 </2ekernel>
238 <*2ekernel | latexrelease>
239 <latexrelease>\IncludeInRelease{2020/10/01}%
240 <latexrelease> \includeonly}{Spaces in file names}%
241 \def\include#1{\relax
242 \ifnum\@auxout=\@partaux
243 \latex@error{\string\include\space cannot be nested}\@eha
244 \else

```

Here the normalization will add **.tex** for all files, (it uses the same normalization as the hooks), so we need to remove that manually. **\@strip@tex@ext** does that.

```

245 \set@curr@file{#1}%
246 \edef\@curr@file{\@strip@tex@ext\@curr@file}%

```

For historical reasons **\@include** expects an argument delimited by a space. This is kept (though unnecessary now) to avoid errors in other packages that use **\@include** directly.

```

247 \expandafter\@include\expandafter{\@curr@file} % deliberate space
248 \fi}

```


Here in `\includeonly` we also need to strip `.tex` after normalization:

```

249 \def\includeonly#1{%
250   \@partswtrue
251   \let\@partlist\@empty
252   \@for\reserved@a:=#1 \do
253     {%
254       \expandafter\set@curr@file\expandafter{\reserved@a}%
255       \ifx\@partlist\@empty
256         \edef\@partlist{\@strip@tex@ext\@curr@file}%
257       \else
258         \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%
259       \fi
260     }%
261   }
262 \@onlypreamble\includeonly

```

(End of definition for `\include` and `\includeonly`.)

`\@strip@tex@ext`
`\@strip@tex@ext@aux`

These macros take a (`\detokenized` file name and remove any `.tex` extension). Extra care is taken to not remove the string `.tex` from the middle of a file name: it is only removed if it's the very last thing in the file name.

```

263 \def\reserved@a#1{%
264   \def\@strip@tex@ext##1{%
265     \expandafter\@strip@tex@ext@aux
266     ##1\@nil\@nil
267     #1\@nil\relax\@nnil}
268 \def\@strip@tex@ext@aux##1#1\@nil##2\@nnil{%
269   \ifx\relax##2\@empty
270     \expandafter\@cdr\expandafter\@empty\@cdr{ }##1%
271   \else##1\fi}}%
272 \expandafter\reserved@a
273 \expandafter{\detokenize{.tex}}
274 </2ekernel | latexrelease>

```

(End of definition for `\@strip@tex@ext` and `\@strip@tex@ext@aux`.)

```

275 <latexrelease>\EndIncludeInRelease
276 <latexrelease>\IncludeInRelease{2019/10/01}%
277 <latexrelease>          {\includeonly}{Spaces in file names}%
278 <latexrelease>
279 <latexrelease>\def\includeonly#1{%
280 <latexrelease>  \@partswtrue
281 <latexrelease>  \set@curr@file{\zap@space#1 \@empty}%
282 <latexrelease>  \let\@partlist\@curr@file
283 <latexrelease>  }
284 <latexrelease>
285 <latexrelease>\def\include#1{\relax
286 <latexrelease>  \ifnum\@auxout=\@partaux
287 <latexrelease>    \@latex@error{\string\include\space cannot be nested}\@eha
288 <latexrelease>  \else

```

```

289 <latexrelease> \set@curr@file{#1 }%
290 <latexrelease> \expandafter\@include\@curr@file
291 <latexrelease> \fi}
292 <latexrelease>
293 <latexrelease>\let\@strip@tex@ext\@undefined
294 <latexrelease>\let\@strip@tex@ext@aux\@undefined
295 <latexrelease>
296 <latexrelease>\EndIncludeInRelease

297 <latexrelease>\IncludeInRelease{0000/00/00}%
298 <latexrelease> \{\includeonly\}{Spaces in file names}%
299 <latexrelease>\def\includeonly#1{%
300 <latexrelease> \@partswtrue
301 <latexrelease> \edef\@partlist{\zap@space#1 \@empty}}
302 <latexrelease>
303 <latexrelease>\def\include#1{\relax
304 <latexrelease> \ifnum\@auxout=\@partaux
305 <latexrelease> \@latex@error{\string\include\space cannot be nested}\@eha
306 <latexrelease> \else \@include#1 \fi}
307 <latexrelease>
308 <latexrelease>\EndIncludeInRelease
309 <*2ekernel>

```

\@include

```

310 </2ekernel>
311 <*2ekernel | latexrelease>
312 <latexrelease>\IncludeInRelease{2022/06/01}%
313 <latexrelease> \{\@include\}{Spaces in file names and hooks}%

314 \def\@include#1 {%
315 \ifx\@nodocument\relax

316 \clearpage
317 \if@filesw
318 \immediate\write\@mainaux{\string\@input{#1.aux}}%
319 \fi
320 \@tempwattrue
321 \if@partsw
322 \@tempswafalse
323 \edef\reserved@a{\#1}%
324 \@for\reserved@a:=\@partlist\do
325 {\ifx\reserved@a\reserved@b\@tempwattrue\fi}%
326 \fi
327 \if@tempswa
328 \let\@auxout\@partaux
329 \if@filesw
330 \immediate\openout\@partaux "#1.aux"
331 \immediate\write\@partaux{\relax}%
332 \fi

```

Now before going to the hooks we need to set \CurrentFile:

```

333 %-----
334 \@filehook@set@CurrentFile

```

Execute the `before` hooks just after we switched the `.aux` file ...

```

335 \UseHook{include/before}%
336 \UseOneTimeHook{include/#1/before}%
337 %-----
338 \input@{#1.tex}%
339 %-----

```

... then end hooks ...

```

340 \UseOneTimeHook{include/#1/end}%
341 \UseHook{include/end}%
342 %-----
343 \clearpage
344 %-----

```

... and after the `\clearpage` the after hooks followed by another `\clearpage` just in case new material got added (after all we need to be in well defined state after the `\include`).

```

345 \UseOneTimeHook{include/#1/after}%
346 \UseHook{include/after}%
347 \clearpage
348 %-----
349 \@writeckpt{#1}%
350 \if@filesw
351 \immediate\closeout\@partaux
352 \fi
353 \else

```

If the file is not included, reset `\deadcycles`, so that a long list of non-included files does not generate an ‘Output loop’ error.

```

354 \deadcycles\z@
355 \@nameuse{cp@#1}%

```

We also execute a hook in this case, first a general used for every include file that is exclude and then a specific one that contains the name of the include file.

```

356 %-----
357 \UseHook{include/excluded}%
358 \UseOneTimeHook{include/#1/excluded}%
359 %-----
360 \fi
361 \let\@auxout\@mainaux
362 \else
363 \@latex@warning{%
364 \noexpand\include should only be used after \string\begin{document}}%
365 \input@{#1}%
366 \fi}

```

Now declare the non-generic `include` hooks used above:

```

367 \NewHook{include/before}
368 \NewReversedHook{include/end}
369 \NewReversedHook{include/after}
370 \NewHook{include/excluded}
371 \<latexrelease> \EndIncludeInRelease
372 \</2ekernel | latexrelease>

```

```

373 <latexrelease>\IncludeInRelease{2020/10/01}%
374 <latexrelease>          {\@include}{Spaces in file names and hooks}%
375 <latexrelease>\EndIncludeInRelease
376 <latexrelease>\def\@include#1 {%
377 <latexrelease>\ifx\@nodocument\relax
378 <latexrelease>  \clearpage
379 <latexrelease>  \if@filesw
380 <latexrelease>    \immediate\write\@mainaux{\string\@input{#1.aux}}%
381 <latexrelease>  \fi
382 <latexrelease>  \@tempswatrue
383 <latexrelease>  \if@partsw
384 <latexrelease>    \@tempswafalse
385 <latexrelease>    \edef\reserved@b{#1}%
386 <latexrelease>    \@for\reserved@a:=\@partlist\do
387 <latexrelease>      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
388 <latexrelease>  \fi
389 <latexrelease>  \if@tempswa
390 <latexrelease>    \let\@auxout\@partaux
391 <latexrelease>    \if@filesw
392 <latexrelease>      \immediate\openout\@partaux "#1.aux"
393 <latexrelease>      \immediate\write\@partaux{\relax}%
394 <latexrelease>    \fi
395 <latexrelease>    \@filehook@set@CurrentFile
396 <latexrelease>    \UseHook{include/before}%
397 <latexrelease>    \UseOneTimeHook{include/#1/before}%
398 <latexrelease>    \@input@{#1.tex}%
399 <latexrelease>    \UseOneTimeHook{include/#1/end}%
400 <latexrelease>    \UseHook{include/end}%
401 <latexrelease>    \clearpage
402 <latexrelease>    \UseOneTimeHook{include/#1/after}%
403 <latexrelease>    \UseHook{include/after}%
404 <latexrelease>    \clearpage
405 <latexrelease>    \@writeckpt{#1}%
406 <latexrelease>    \if@filesw
407 <latexrelease>      \immediate\closeout\@partaux
408 <latexrelease>    \fi
409 <latexrelease>  \else
410 <latexrelease>    \deadcycles\z@
411 <latexrelease>    \@nameuse{cp@#1}%
412 <latexrelease>  \fi
413 <latexrelease>  \let\@auxout\@mainaux
414 <latexrelease>\else
415 <latexrelease>\@latex@warning{%
416 <latexrelease>  \noexpand\include should only be used after \string\begin{document}}%
417 <latexrelease>\@input@{#1}%
418 <latexrelease>\fi}
419 <latexrelease>\NewHook{include/before}
420 <latexrelease>\NewReversedHook{include/end}
421 <latexrelease>\NewReversedHook{include/after}
422 <latexrelease>\IncludeInRelease{0000/00/00}%
423 <latexrelease>          {\@include}{Spaces in file names and hooks}%
424 <latexrelease>\def\@include#1 {%
425 <latexrelease>  \clearpage
426 <latexrelease>  \if@filesw

```

```

427 <latexrelease> \immediate\write\@mainaux{\string\@input{#1.aux}}%
428 <latexrelease> \fi
429 <latexrelease> \@tempswatrue
430 <latexrelease> \if@partsw
431 <latexrelease> \@tempswafalse
432 <latexrelease> \edef\reserved@b{#1}%
433 <latexrelease> \@for\reserved@a:=\@partlist\do
434 <latexrelease> {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
435 <latexrelease> \fi
436 <latexrelease> \if@tempswa
437 <latexrelease> \let\@auxout\@partaux
438 <latexrelease> \if@filesw
439 <latexrelease> \immediate\openout\@partaux #1.aux
440 <latexrelease> \immediate\write\@partaux{\relax}%
441 <latexrelease> \fi
442 <latexrelease> \@input@{#1.tex}%
443 <latexrelease> \clearpage
444 <latexrelease> \@writeckpt{#1}%
445 <latexrelease> \if@filesw
446 <latexrelease> \immediate\closeout\@partaux
447 <latexrelease> \fi
448 <latexrelease> \else
449 <latexrelease> \deadcycles\z@
450 <latexrelease> \@nameuse{cp@#1}%
451 <latexrelease> \fi
452 <latexrelease> \let\@auxout\@mainaux}
453 <latexrelease>
454 <latexrelease>\EndIncludeInRelease
455 <*2ekernel>

```

(End of definition for \@include.)

\@writeckpt

```

456 \def\@writeckpt#1{%
457   \if@filesw
458     \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
459     {\let\@elt\@wckptelt \cl@ckpt}%
460     \immediate\write\@partaux{\@charrb}%
461   \fi}

```

(End of definition for \@writeckpt.)

\@wckptelt

```

462 \def\@wckptelt#1{%
463   \immediate\write\@partaux{%
464     \string\setcounter{#1}{\the\@nameuse{c@#1}}}}

```

(End of definition for \@wckptelt.)

\@setckpt RmS 93/08/31: introduced \@setckpt

```

465 \def\@setckpt#1{\global\@namedef{cp@#1}}

```

(End of definition for \@setckpt.)

`\@charlb` The following defines `\@charlb` and `\@charrb` to be { and }, respectively with `\catcode`
`\@charrb` 11.

```

466 {\catcode' [=1 \catcode']=2
467 \catcode' {=11 \catcode'}=11
468 \gdef\@charlb[{
469 \gdef\@charrb[]
470 ]% }brace matching

```

(End of definition for \@charlb and \@charrb.)

1.1 Safe Input Macros

`\@curr@file` File name handling is done by generating a csname from the provided file name (which
`\set@curr@file` means that UTF-8 octets gets turned into strings as this is what happens if they appear
in a csname due to the code in `utf8.def`). By setting `\escapechar` to -1 we ensure that
we don't get a backslash in front. As a result we end up with all characters as catcode
12 (plus spaces). We then sometimes add quotes around the construct (removing any
existing inner quotes). Sometimes we only remove the quotes if they have been supplied
by the user. There is clearly some room for improvement.

A side effect of the new code is that we will see quotes around file name displays
where there haven't been any before.

For compatibility with existing code using `{abc}.tex` or `{one.two}.png`, an initial
brace group is discarded before expansion and `\string` is applied. The content of the
brace group is discarded. This means that a leading space will be lost unless protected
(by { } or " " or `\space`) but filenames with a space are hopefully rare.

The definition below is from 2019 and only used during kernel bootstrapping, later
on in `ltfilehook.dtx` it will get overwritten.

```

471 \def\set@curr@file#1{%
472   \begingroup
473   \escapechar\m@ne
474   \xdef\@curr@file{%
475     \expandafter\expandafter\expandafter\unquote@name
476     \expandafter\expandafter\expandafter{%
477       \expandafter\string
478       \csname\@firstofone#1\@empty\endcsname}}%
479   \endgroup
480 }

```

(End of definition for \@curr@file and \set@curr@file.)

`\quote@name` Quoting spaces
`\quote@@name`
`\unquote@name`

<code>a b c</code>	<code>-> "a b c"</code>
<code>"a b c"</code>	<code>-> "a b c"</code>
<code>a" "b" "c</code>	<code>-> "a b c"</code>
	<code>-> ""</code>

```

481 </2kernel>
482 <*2kernel | latexrelease>
483 <latexrelease> \IncludeInRelease{2019/10/01}%
484 <latexrelease> { \quote@name } { Quote file names }%
485 \def\quote@name#1{ "\quote@@name#1\@gobble" }
486 \def\quote@@name#1{ #1\quote@@name }

```

and removing quotes ...

```
487 \def\unquote@name#1{\quote@@name#1@gobble"}
```

(End of definition for \quote@name, \quote@@name, and \unquote@name.)

\IfFileExists

```
488 \DeclareRobustCommand\IfFileExists[1]{%
489   \set@curr@file{#1}%
490   \expandafter\IfFileExists@\expandafter{\@curr@file}}

(End of definition for \IfFileExists.)

491 </2ekernel | latexrelease>
492 <latexrelease>\EndIncludeInRelease
493 <latexrelease>\IncludeInRelease{0000/00/00}%
494 <latexrelease>           {\quote@name}{Quote file names}%
495 <latexrelease>
496 <latexrelease>\let\quote@name\@undefined
497 <latexrelease>\let\quote@@name\@undefined
498 <latexrelease>\let\unquote@name\@undefined
499 <latexrelease>
500 <latexrelease>\long\def \IfFileExists#1#2#3{%
501 <latexrelease>  \openin\@inputcheck#1 %
502 <latexrelease>  \ifeof\@inputcheck
503 <latexrelease>    \ifx\input@path\@undefined
504 <latexrelease>    \def\reserved@a{#3}%
505 <latexrelease>    \else
506 <latexrelease>    \def\reserved@a{\@iffilenamepath{#1}{#2}{#3}}%
507 <latexrelease>    \fi
508 <latexrelease>  \else
509 <latexrelease>    \closein\@inputcheck
510 <latexrelease>    \edef\@filef@und{#1 }%
511 <latexrelease>    \def\reserved@a{#2}%
512 <latexrelease>    \fi
513 <latexrelease>    \reserved@a}
514 <latexrelease>
515 <latexrelease>\EndIncludeInRelease
516 <*2ekernel>
```

\IfFileExists@ Argument #1 is \@curr@file so catcode 12 string with no quotes.

\IfFileExists@@ The original definition picked up arguments #2 and #3 in a way that they couldn't contain unbalanced conditionals. A better implementation would have been not to pick up the arguments at all but instead use the usual \@firstoftwo and \@secondoftwo. However, that changes how # is interpreted and so we can't do that nowadays without invalidating a lot of code. Therefore the somewhat curious construction near the end.

Earlier versions used \openin here, but this led to two code paths, one in expl3 and one here. To avoid that, and as the expl3 approach works by expansion, we use that here. As we need the file name to include the path, the actual expl3 function used is not the file existence test!

```
517 </2ekernel>
518 <*2ekernel | latexrelease>
519 <latexrelease>\IncludeInRelease{2023/06/01}%
520 <latexrelease>           {\IfFileExists@}{\IfFileExists}
521 \long\def \IfFileExists@#1#2#3{%
522   \edef\@filef@und{\IfFileExists@@{#1}}%
```

The `expl3` function regards an empty argument as nothing at all, but the L^AT_EX 2_ε convention is that this is equal to the special `.tex` file.

```

523 \ifx\@filef@und\@empty
524 \if\relax\detokenize{#1}\relax
525 \let\reserved@a\@firstoftwo
526 \def\@filef@und{" .tex" }%
527 \else
528 \let\reserved@a\@secondoftwo
529 \fi
530 \else
531 \let\reserved@a\@firstoftwo
532 \edef\@filef@und{"\@filef@und" }%
533 \fi

```

This is just there so that any `#` inside `#2` or `#3` needs doubling (as that was the case in the past).

```

534 \expandafter\def\expandafter\reserved@a
535 \expandafter{\reserved@a{#2}{#3}}%
536 \reserved@a}

```

Pipes are not really files, but at the document level they are supported. To quickly trim of any leading spaces, we use a blank test and `\use:n` rather than `\tl_trim_spaces:n` for speed as we don't care about the end of the input.

```

537 \ExplSyntaxOn
538 \cs_new:Npn \IfFileExists@@ #1
539 {
540   \tl_if_blank:nF {#1}
541   {
542     \tl_if_head_eq_charcode:oNTF { \use:n #1 } |
543     {#1}
544     { \file_full_name:n {#1} }
545   }
546 }
547 \cs_generate_variant:Nn \tl_if_head_eq_charcode:nNTF { o }
548 \ExplSyntaxOff
549 </2ekernel | latexrelease>
550 <latexrelease>\EndIncludeInRelease
551 <latexrelease>\IncludeInRelease{2021/06/01}%
552 <latexrelease>          {\IfFileExists@}{\IfFileExists}
553 <latexrelease>
554 <latexrelease>\long\def \IfFileExists@#1#2#3{%
555 <latexrelease> \openin\@inputcheck"#1" %
556 <latexrelease> \ifeof\@inputcheck
557 <latexrelease> \ifx\input@path\@undefined
558 <latexrelease> \let\reserved@a\@secondoftwo
559 <latexrelease> \else
560 <latexrelease> \def\reserved@a{\@iffileonpath{#1}}%
561 <latexrelease> \fi
562 <latexrelease> \else
563 <latexrelease> \closein\@inputcheck
564 <latexrelease> \edef\@filef@und{"#1" }%
565 <latexrelease> \let\reserved@a\@firstoftwo
566 <latexrelease> \fi
567 <latexrelease> \expandafter\def\expandafter\reserved@a

```



```

568 <latexrelease> \expandafter{\reserved@a{#2}{#3}}%
569 <latexrelease>\reserved@a}
570 <latexrelease>\let\IfFileExists@@\undefined
571 <latexrelease>\EndIncludeInRelease
572 <latexrelease>
573 <latexrelease>\IncludeInRelease{2019/10/01}%
574 <latexrelease> \IfFileExists@{\IfFileExists}
575 <latexrelease>
576 <latexrelease>\long\def \IfFileExists@#1#2#3{%
577 <latexrelease> \openin\@inputcheck"#1" %
578 <latexrelease> \ifeof\@inputcheck
579 <latexrelease> \ifx\input@path\@undefined
580 <latexrelease> \def\reserved@a{#3}%
581 <latexrelease> \else
582 <latexrelease> \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
583 <latexrelease> \fi
584 <latexrelease> \else
585 <latexrelease> \closein\@inputcheck
586 <latexrelease> \edef\@filef@und{"#1" }%
587 <latexrelease> \def\reserved@a{#2}%
588 <latexrelease> \fi
589 <latexrelease> \reserved@a}
590 <latexrelease>\EndIncludeInRelease
591 <latexrelease>\IncludeInRelease{0000/00/00}%
592 <latexrelease> \IfFileExists@{\IfFileExists}
593 <latexrelease>
594 <latexrelease>\let\IfFileExists@\@undefined
595 <latexrelease>
596 <latexrelease>
597 <latexrelease>\EndIncludeInRelease
598 <*2ekernel>

```

(End of definition for \IfFileExists@ and \IfFileExists@@.)

\@iffileonpath If the file is not found by \openin, and \input@path is defined, look in all the directories specified in \input@path.

```

599 </2ekernel>
600 <*2ekernel | latexrelease>
601 <latexrelease>\IncludeInRelease{2019/10/01}%
602 <latexrelease> \@iffileonpath{\Quote file names}
603 \long\def\@iffileonpath#1{%
604 \let\reserved@a\@secondoftwo
605 \expandafter\@tfor\expandafter\reserved@b\expandafter
606 : \expandafter=\input@path\do{%
607 \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
608 \ifeof\@inputcheck\else
609 \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
610 \let\reserved@a\@firstoftwo%
611 \closein\@inputcheck
612 \@break@tfor
613 \fi}%
614 \reserved@a}

```

(End of definition for \@iffileonpath.)

```

615 </2ekernel | latexrelease>
616 <latexrelease>\EndIncludeInRelease
617 <latexrelease>\IncludeInRelease{0000/00/00}%
618 <latexrelease>          {\quote@name}{Quote file names}
619 <latexrelease>
620 <latexrelease>\long\def\@iffileonpath#1{%
621 <latexrelease>  \let\reserved@a\@secondoftwo
622 <latexrelease>  \expandafter\@tfor\expandafter\reserved@b\expandafter
623 <latexrelease>          :\expandafter=\input@path\do{%
624 <latexrelease>    \openin\@inputcheck\reserved@b#1 %
625 <latexrelease>    \ifeof\@inputcheck\else
626 <latexrelease>    \edef\@filef@und{\reserved@b#1 }%
627 <latexrelease>    \let\reserved@a\@firstoftwo%
628 <latexrelease>    \closein\@inputcheck
629 <latexrelease>    \@break@tfor
630 <latexrelease>    \fi}%
631 <latexrelease>  \reserved@a}
632 <latexrelease>
633 <latexrelease>\EndIncludeInRelease
634 <*2ekernel>

```

\InputIfFileExists Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute ‘#3’.

This here is a temporary definition for the kernel. The real one comes somewhat later in the file `ltxfilehook.dtx`.

```

635 \DeclareRobustCommand \InputIfFileExists[2]{%
636   \IfFileExists{#1}%
637   {%
638     \expandafter\@swaptwoargs\expandafter
639     {\@filef@und}{#2\@addtofilelist{#1}\@input}}

```

(End of definition for \InputIfFileExists.)

\@swaptwoargs Swap two arguments and return them unbraced (like `\@firstoftwo` etc).

```

640 </2ekernel>
641 <*2ekernel | latexrelease>
642 <latexrelease>\IncludeInRelease{2019/10/01}%
643 <latexrelease>  {\@swaptwoargs}{Don't lose the file name}%
644 <long\def\@swaptwoargs#1#2{#2#1}

645 </2ekernel | latexrelease>
646 <latexrelease>\EndIncludeInRelease
647 <latexrelease>\IncludeInRelease{0000/00/00}%
648 <latexrelease>  {\@swaptwoargs}{Don't lose the file name}%
649 <latexrelease>\let\@swaptwoargs\undefined
650 <latexrelease>\EndIncludeInRelease
651 <*2ekernel>

```

(End of definition for \@swaptwoargs.)

\input Input a file: if the argument is given in braces use safe input macros, otherwise use TeX’s primitive `\input` command (which is called `\@input` in L^AT_EX).

```

652 \def\input{\@ifnextchar\bgroup\@input\@input}

```

(End of definition for \input.)

`\@input` Define `\@input` (i.e., `\input`) in terms of `\InputIfFileExists`.
Changes to `\@input`: adapt to the changes to `\@missingfileerror`.

```

653 </2ekernel>
654 <*2ekernel | latexrelease>
655 <latexrelease> \IncludeInRelease{2020/10/01}%
656 <latexrelease>           {\@input}{Change in file error handling}%
657 \def\@input#1{%
658   \InputIfFileExists{#1}{}%
659   {\filename@parse\@curr@file
660    \edef\reserved@a{\noexpand\@missingfileerror
661     {\filename@area\filename@base}%
662     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%

```

This line now just sets `\@missingfile@<part>`:

```

663   \reserved@a

```

Now here we have to use it. The file here is guaranteed to exist, because `\@missingfileerror` ensures so, but we have to use `\InputIfFileExists` because it executes the file hooks.

```

664   \edef\reserved@a{\noexpand\@input{%
665    \@missingfile@area\@missingfile@base.\@missingfile@ext}}%
666   \reserved@a}}
667 </2ekernel | latexrelease>
668 <latexrelease> \EndIncludeInRelease
669 <latexrelease> \IncludeInRelease{2019/10/01}%
670 <latexrelease>           {\@input}{Quote file names}%
671 <latexrelease>
672 <latexrelease> \def\@input#1{%
673 <latexrelease>   \InputIfFileExists{#1}{}%
674 <latexrelease>   {\filename@parse\@curr@file
675 <latexrelease>   \edef\reserved@a{\noexpand\@missingfileerror
676 <latexrelease>     {\filename@area\filename@base}%
677 <latexrelease>     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
678 <latexrelease>   \reserved@a}}
679 <latexrelease> \EndIncludeInRelease
680 <latexrelease> \IncludeInRelease{0000/00/00}%
681 <latexrelease>           {\@input}{Quote file names}%
682 <latexrelease> \def\@input#1{%
683 <latexrelease>   \InputIfFileExists{#1}{}%
684 <latexrelease>   {\filename@parse{#1}%
685 <latexrelease>   \edef\reserved@a{\noexpand\@missingfileerror
686 <latexrelease>     {\filename@area\filename@base}%
687 <latexrelease>     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
688 <latexrelease>   \reserved@a}}
689 <latexrelease> \EndIncludeInRelease
690 <*2ekernel>

```

(End of definition for \@input.)

`\@input` Define `\@input` in terms of `\IfFileExists`. So this is a ‘safe input’ command, but the files input are not listed by `\listfiles`.

We don’t want `.aux`, `.toc` files etc be listed by `\listfiles`. However, something like `.bbl` probably should be listed and thus should be implemented not by `\@input`.

```

691 \def\@input#1{%
692   \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}

```

(End of definition for \@input.)

\@input@ Version of \@input that does add the file to \@filelist.

```
693 \def\@input@#1{\InputIfFileExists{#1}{\typeout{No file #1.}}}
```

(End of definition for \@input@.)

\@missingfileerror This ‘error’ command avoids T_EX’s primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

Changes to \@missingfileerror: rather than trying to input the file by force, now \@missingfileerror just returns three \@missingfile@<part> and the caller macro is responsible for doing the right thing with it.

```
694 </2ekernel>
695 <*2ekernel|latexrelease>
696 <latexrelease>\IncludeInRelease{2020/10/01}%
697 <latexrelease>{\@missingfileerror}{Do not load missing file immediately}%
698 \gdef\@missingfileerror#1#2{%
699   \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
700   Type X to quit or <RETURN> to proceed,^^J%
701   or enter new name. (Default extension: #2)^^J}%
702   \message{Enter file name: }%
703   {\endlinechar\m@ne
704   \global\read\m@ne to\@gtempa}%
705   \ifx\@gtempa\@empty
```

If the user answers with <return>, fallback to the .tex file (previously it did nothing).

```
706   \let\@missingfile@area\@empty
707   \let\@missingfile@base\@empty
708   \def\@missingfile@ext{tex}%
709   \else
```

Use \batchmode\read-1 to <t1> to end the T_EX run, same as expl3 does (it was \batchmode\@@end before).

```
710   \def\reserved@b{\batchmode\read-1 to \reserved@a}%
711   \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
712   \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
713   \filename@parse\@gtempa
714   \edef\filename@ext{%
715     \ifx\filename@ext\relax#2\else\filename@ext\fi}%
716   \edef\reserved@a{%
```

Only check \IfFileExists (it was \InputIfFileExists).

```
717   \noexpand\IfFileExists
718     {\filename@area\filename@base.\filename@ext}%
```

If the file exists, define \@missingfile@<part>.

```
719     {\def\noexpand\@missingfile@area{\filename@area}%
720     \def\noexpand\@missingfile@base{\filename@base}%
721     \def\noexpand\@missingfile@ext {\filename@ext}}%
722     {\noexpand\@missingfileerror
723     {\filename@area\filename@base}{\filename@ext}}}%
724   \reserved@a
725   \fi
726 }
727 </2ekernel|latexrelease>
728 <latexrelease>\EndIncludeInRelease
```

```

729 <latexrelease>\IncludeInRelease{0000/00/00}%
730 <latexrelease>      {\@missingfileerror}{Do not load missing file immediately}%
731 <latexrelease>
732 <latexrelease>\gdef\@missingfileerror#1#2{%
733 <latexrelease>      \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
734 <latexrelease>      Type X to quit or <RETURN> to proceed,^^J%
735 <latexrelease>      or enter new name. (Default extension: #2)^^J}%
736 <latexrelease>      \message{Enter file name: }%
737 <latexrelease>      {\endlinechar\m@ne
738 <latexrelease>      \global\read\m@ne to\@gtempa}%
739 <latexrelease>      \ifx\@gtempa\@empty
740 <latexrelease>      \else
741 <latexrelease>          \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@end\fi
742 <latexrelease>          \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@end\fi
743 <latexrelease>          \filename@parse\@gtempa
744 <latexrelease>          \edef\filename@ext{%
745 <latexrelease>              \ifx\filename@ext\relax#2\else\filename@ext\fi}%
746 <latexrelease>          \edef\reserved@a{%
747 <latexrelease>              \noexpand\InputIfFileExists
748 <latexrelease>              {\filename@area\filename@base.\filename@ext}%
749 <latexrelease>              }%
750 <latexrelease>          {\noexpand\@missingfileerror
751 <latexrelease>              {\filename@area\filename@base}{\filename@ext}}}%
752 <latexrelease>      \reserved@a
753 <latexrelease>      \fi}
754 <latexrelease>
755 <latexrelease>\EndIncludeInRelease
756 <*2ekernel>

```

(End of definition for \@missingfileerror.)

\@obsoletefile For compatibility with L^AT_EX 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```

757 \def\@obsoletefile#1#2{%
758     \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
759 \@onlypreamble\@obsoletefile

```

1.2 Listing files

A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```

760 \let\@filelist\@gobble

```

Add to the list of files input so far. This ‘real’ definition is only used for ‘cfg’ files during intex. An initial definition of `\@gobble` has already been set.

```

761 %\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}

```

```

\if@listfiles@hashes 762 \ExplSyntaxOn
\if@listfiles@sizes 763 \keys_define:nn { __kernel / listfiles }
764 {
765     hashes .legacy_if_set:n = @listfiles@hashes ,

```

```

766     sizes .legacy_if_set:n = @listfiles@sizes
767   }
768 \ExplSyntaxOff

```

A preamble command to cause `\end{document}` to list files input from the main file.

```

\listfiles 769 \NewDocumentCommand\listfiles{0{}}{%
770   \SetKeys[_kernel/listfiles]{#1}%
771   \let\listfiles\relax
772   \def\@listfiles##1##2##3##4##5##6##7##8##9\@{%
773     \def\reserved@d{\}%
774     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
775       \ifx\reserved@c\reserved@d
776         \edef\filename@area{ \filename@area}%
777       \fi}}%
778   \def\@dofilelist{%
779     \typeout{^^J *File List*}%
780     \@for\@currname:=\@filelist\do{%
781       \filename@parse\@currname
782       \edef\reserved@a{%
783         \filename@base.%
784         \ifx\filename@ext\relax tex\else\filename@ext\fi}%
785       \expandafter\let\expandafter\reserved@b
786         \csname ver@\reserved@a\endcsname

```

Packages that `\relax` their `\ver@...` string to allow for multiple loading (e.g., fontenc) can use `\ver@@...` to store the version information instead.

```

787     \ifx\reserved@b\relax
788       \expandafter\let\expandafter\reserved@b
789         \csname ver@@\reserved@a\endcsname
790     \fi
791     \expandafter\expandafter\expandafter\@listfiles\expandafter
792       \filename@area\filename@base\////////////////////////////////\@
793     \typeout{%
794       \filename@area\reserved@a
795       \ifx\reserved@b\relax\else\@spaces\reserved@b\fi

```

Now we add the additional information if requested.

```

796     \ifnum0%
797       \if@listfiles@hashes1\fi
798       \if@listfiles@sizes1\fi
799       >0 %
800       ^^J\@spaces
801       (%
802         \if@listfiles@sizes
803           size \@dofilelist@size\@currname
804           \if@listfiles@hashes
805             , %
806           \fi
807         \fi
808         \if@listfiles@hashes
809           hash \@dofilelist@hash\@currname
810         \fi
811       )%
812     \fi

```

```

813     }}%
814     \typeout{ *****^~J}}

815 \ExplSyntaxOn
\@dofilelist@hash 816 \cs_new_eq:NN \@dofilelist@hash \file_md5five_hash:n
\@dofilelist@size 817 \cs_new_eq:NN \@dofilelist@size \file_size:n
818 \ExplSyntaxOff

    The \@filelist will be de-activated if \listfiles does not appear in the preamble.
    \begin{document} contains code equivalent to the following:

\AtBeginDocument{%
    \ifx\@listfiles\@undefined
        \let\@filelist\relax
        \let\@addtofilelist\@gobble
    \fi}

819 \@onlypreamble\listfiles

\@dofilelist 820 \let\@dofilelist\relax

821 </2ekernel>

(End of definition for \@obsoletefile and others.)

```

File 21

ltoutenc.dtx

1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OML, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{<command>}{<encoding>}
[<number>][<default>]{<commands>}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{\@xxxii l}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{<command>}{<encoding>}
[<number>][<default>]{<commands>}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{<command>}{<encoding>}{<slot>}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{<command>}{<encoding>}{<slot>}
```


This command declares a text accent. The commands:

```
\DeclareTextAccent{"}{OT1}{127}
\DeclareTextCommand{"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{<command>}
                               {<encoding>}{<argument>}{<slot>}
```

This command declares a composite letter, for example in the T1 encoding `\'a` is slot 225, which is declared by:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{<command>}
                               {<encoding>}{<argument>}{<text>}
```

For example, in the OT1 encoding Å has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
{\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
 \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

```
\UseTextSymbol{<encoding>}{<command>}
```

For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

```
\UseTextAccent{<encoding>}{<command>}{<text>}
```

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{\'}{a}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\'{\fontencoding{OT2}\selectfont a}}
```

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{<command>}{<definition>}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{<command>}{<definition>}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{<command>}{<encoding>}
\DeclareTextAccentDefault{<command>}{<encoding>}
```

are short for:

```
\DeclareTextCommandDefault{<command>}
      {\UseTextSymbol{<encoding>}{<command>}}
\DeclareTextCommandDefault[1]{<command>}
      {\UseTextAccent{<encoding>}{<command>}{#1}}
```

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in OT1 is a hack since \$ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., TS1) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L^AT_EX will still find the encoding specific-definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L^AT_EX to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar}{T1}
\DeclareTextCommandDefault{\textdollar}
{\UseTextSymbol{TS1}\textdollaroldstyle}
```

1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

1.3 Docstrip modules

This `.dtx` file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

T1	generates <code>t1enc.def</code> for the Cork encoding.
TS1	generates <code>ts1enc.def</code> for the Text Companion encoding.
TS1sty	generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.
OT1	generates <code>ot1enc.def</code> for Knuth's CM encoding.
OMS	generates <code>omsenc.def</code> for Knuth's math symbol encoding.
OML	generates <code>omlenc.def</code> for Knuth's math letters encoding.
OT4	generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
TU	generates <code>tuenc.def</code> for Unicode font encoding.
package	generates <code>fontenc.sty</code> for selecting encodings.
2ekernel	for the kernel commands.

1.4 Definitions for the kernel

1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in .def files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 <*2ekernel>
2 \message{font encodings,}
```

Far too many macros in one block here!

```
\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
  \@dec@text@cmd
\chardef@text@cmd
  \@changed@cmd
  \@changed@x
\TextSymbolUnavailable
  \@inmathwarn
```

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
3 \def\DeclareTextCommand{%
4   \@dec@text@cmd\newcommand}
5 \def\ProvideTextCommand{%
6   \@dec@text@cmd\providecommand}
7 \def\@enc@info#1{%
8   \GenericInfo{(Encoding)\@spaces\@spaces\@spaces\space\space}%
9   {LaTeX Encoding Info: \space\space\space#1}}%
10 \def\@dec@text@cmd#1#2#3{%
```

For logging, there are a number of different cases to cater for.

```
11 \ifcsname #3\string#2\endcsname
12 \@enc@info{%
13   \ifx#1\providecommand
14     Ignoring declaration for
15   \else
16     Redeclaring
17   \fi
18   text
```

This test distinguishes between commands and symbols:

```
19 \ifx#1\chardef@text@cmd
20   symbol
21 \else
22   command
23 \fi
24 \string#2 (encoding #3)}%
25 \fi
26 \expandafter\def\expandafter#2%
27   \expandafter{%
28     \csname#3-cmd\expandafter\endcsname
29     \expandafter#2%
30     \csname#3\string#2\endcsname
31   }%
32 \let\@ifdefinable\@rc@ifdefinable
33 \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `\@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\@ifdefinable` (following its disabling in `\@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```

34 \def\chardef@text@cmd{%
35   \let\@ifdefinable\@ifdefinable
36   \chardef
37 }
38 \def\DeclareTextSymbol#1#2#3{%
39   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
40 }
```

The declarations are only available before `\begin{document}`.

```

41 \@onlypreamble\DeclareTextCommand
42 \@onlypreamble\DeclareTextSymbol
```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If `\protect` is `\@typeset@protect` and `\cf@encoding` is `T1`, then we execute `\T1\foo`. This should be the normal behaviour, and is optimized for speed.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, and `\OT1\foo` is defined, then we execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in text mode, and `\OT1\foo` is undefined, then we define `\OT1\foo` to be the default value of `\foo`, and execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in math mode, and `\OT1\foo` is undefined, then we execute the default value of `\foo`. (This is necessary so that things like `$X_{copyright}` work properly.)
- If `\protect` is not `\@typeset@protect` then we execute `\noexpand\foo`. For example, if we are writing to a file, then this results in `\foo` being written. If we are in a `\mark`, then `\foo` will be put in the mark—since `\foo` is robust, it will then survive all the things which may happen to it whilst it's a `\mark`.

So after all that, we will either execute the appropriate definition of `\foo` for the current encoding, or we will execute `\noexpand\foo`.

The default value of `\foo` is `\?\foo` if it is defined, and an error message otherwise.

When the encoding is changed from `T1` to `OT1`, `\T1-cmd` is defined to be `\@changed@cmd` and `\OT1-cmd` is defined to be `\@current@cmd`. This means that the test for what the current encoding is can be performed quickly.

```

43 \def\@current@cmd#1{%
44   \ifx\protect\@typeset@protect
45     \inmathwarn#1%
46   \else
47     \noexpand#1\expandafter\@gobble
48   \fi}
```

```

49 \def\@changed@cmd#1#2{%
50   \ifx\protect\@typeset@protect
51     \@inmathwarn#1%
52     \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
53       \expandafter\ifx\csname ?\string#1\endcsname\relax
54         \expandafter\def\csname ?\string#1\endcsname{%
55           \TextSymbolUnavailable#1%
56         }%
57       \fi
58     \global\expandafter\let
59       \csname\cf@encoding\string#1\expandafter\endcsname
60       \csname ?\string#1\endcsname
61     \fi
62     \csname\cf@encoding\string#1%
63       \expandafter\endcsname
64   \else
65     \noexpand#1%
66   \fi}

67 \gdef\TextSymbolUnavailable#1{%
68   \@latex@error{%
69     Command \protect#1 unavailable in encoding \cf@encoding%
70   }\@eha}

```

The command `\@inmathwarn` produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call `\relax` before the `\ifmmode`. This means that it is possible for the warning to fail to be issued at the beginning of a row of an `halign` whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a `\relax` would break ligatures and kerning between text symbols.

A more efficient solution would be to make `\@inmathwarn` and `\@inmatherr` equal to `\@empty` and `\relax` by default, and to have `\everymath` reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```

71 \def\@inmathwarn#1{%
72   \ifmmode
73     \@latex@warning{Command \protect#1 invalid in math mode}%
74   \fi}

```

(End of definition for `\DeclareTextCommand` and others.)

`\DeclareTextCommandDefault`
`\ProvideTextCommandDefault`

These define commands with encoding ?.

Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc .def files, so is allowed anywhere.

```

75 \def\DeclareTextCommandDefault#1{%
76   \DeclareTextCommand#1?}

77 \def\ProvideTextCommandDefault#1{%
78   \ProvideTextCommand#1?}

79 \@onlypreamble\DeclareTextCommandDefault
80 %\@onlypreamble\ProvideTextCommandDefault

```

They require `\? -cmd` to be initialized as `\@changed@cmd`.

```

81 \expandafter\let\csname?\-cmd\endcsname\@changed@cmd

```

(End of definition for `\DeclareTextCommandDefault` and `\ProvideTextCommandDefault`.)

`\DeclareTextAccent` This is just a disguise for defining a T_EX `\accent` command.

```
82 \def\DeclareTextAccent#1#2#3{%  
83   \DeclareTextCommand#1{#2}{\add@accent{#3}}}  
84 \@onlypreamble\DeclareTextAccent
```

(End of definition for `\DeclareTextAccent`.)

`\add@accent` To save space this code is shared between all text accents that are set using the `\accent` primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve `\setbox` operations which also inhibit the mechanism of the `\accent` primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.

```
85 \def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
86   \let\hmode@start@before@group\@firstofone  
87   \setbox\@tempboxa\hbox{#2%
```

When presetting the argument in a box we record its `\spacefactor` for later use after the accent got typeset. This way something like `\A` gets the spacefactor of `A` (i.e., 999) rather than the default value of 1000.

```
88     \global\mathchardef\accent@spacefactor\spacefactor}%
```

The accent primitive doesn't allow things `\begingroup` to interfere between accent and base character. Therefore we need to avoid that (they are some hidden inside `\maybe@load@fontshape`). As we don't have to load the fontshape in this case (as that already happened in the box above, if necessary), we simply disable that part of the code temporarily. We also ignore `\ignorespaces` which has the same issue and may show up as part of `\normalfont` if that is used.

```
89   \let\maybe@load@fontshape\relax  
90   \let\ignorespaces\relax  
91   \accent#1 #2\egroup\ifmmode\else\spacefactor\accent@spacefactor\fi}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
92 \let\accent@spacefactor\relax
```

(End of definition for `\add@accent`.)

`\hmode@bgroup`

```
93 \def\hmode@bgroup{\leavevmode\bgroup}
```

(End of definition for `\hmode@bgroup`.)

`\DeclareTextCompositeCommand` Another amusing game to play with `\expandafter`, `\csname`, and `\string`. When you say `\DeclareTextCompositeCommand{\foo}{T1}{a}{bar}`, we look to see if the expansion of `\T1\foo` begins with `\@text@composite`, and if it doesn't, we redefine `\T1\foo` to be:

```

\DeclareTextComposite
  \@text@composite
  \@text@composite@x
    \@strip@args
      #1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}

```

where ... is the previous definition of `\T1\foo`. Finally, we define `\T1\foo-a` to expand to `bar`.

```

94 </2ekernel>
95 <latexrelease>\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
96 <latexrelease>                                     {test for undeclared accent}%
97 <*2ekernel | latexrelease>
98 \def\DeclareTextCompositeCommand#1#2#3#4{%
99   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
100   \ifx\reserved@a\relax
101     \DeclareTextCommand#1{#2}{%
102       \@latex@error{Text composite \string#1 undeclared in encoding #2}\@eha}%
103       \@enc@info{Text composite with undeclared \string#1 in encoding #2}%
104       \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
105       \fi
106       \expandafter\expandafter\expandafter\ifx
107       \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
108       \edef\reserved@b##1{%
109         \def\expandafter\noexpand
110           \csname#2\string#1\endcsname####1{%
111           \noexpand\@text@composite
112             \expandafter\noexpand\csname#2\string#1\endcsname
113             ####1\noexpand\@empty\noexpand\@text@composite
114             {##1}}}%
115       \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
116       \fi
117       \expandafter\def\csname\expandafter\string\csname
118         #2\endcsname\string#1-\string#3\@empty\endcsname{#4}%
119     }
120 </2ekernel | latexrelease>
121 <latexrelease>\EndIncludeInRelease
122 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareTextCompositeCommand}
123 <latexrelease>                                     {test for undeclared accent}%
124 <latexrelease>\def\DeclareTextCompositeCommand#1#2#3#4{%
125 <latexrelease>   \expandafter\let\expandafter\reserved@a
126 <latexrelease>                                     \csname#2\string#1\endcsname
127 <latexrelease>   \expandafter\expandafter\expandafter\ifx
128 <latexrelease>   \expandafter\@car\reserved@a\relax\relax\@nil
129 <latexrelease>                                     \@text@composite \else
130 <latexrelease>     \edef\reserved@b##1{%
131 <latexrelease>       \def\expandafter\noexpand
132 <latexrelease>         \csname#2\string#1\endcsname####1{%
133 <latexrelease>         \noexpand\@text@composite
134 <latexrelease>           \expandafter\noexpand\csname#2\string#1\endcsname
135 <latexrelease>           ####1\noexpand\@empty\noexpand\@text@composite
136 <latexrelease>           {##1}}}%
137 <latexrelease>     \expandafter\reserved@b\expandafter{\reserved@a{##1}}%

```



```

138 <latexrelease> \fi
139 <latexrelease> \expandafter\def\csname\expandafter\string\csname
140 <latexrelease> #2\endcsname\string#1-\string#3\@empty\endcsname{#4}}
141 <latexrelease>\EndIncludeInRelease
142 <*2ekernel>

143 \@onlypreamble\DeclareTextCompositeCommand

```

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\T1\foo-A` if `\T1\foo-A` has been defined, and `{...}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the `\csname`. This is so that `\'\{\textit{e}\}` will work—it checks whether `\T1\'-\textit` is defined (which presumably it isn't) and so expands to `\{\accent 1 \textit{e}\}`.

This trick won't always work, for example `\'\{\itshape e\}` will expand to (with spaces added for clarity):

```
\csname \string \T1\' - \string {\itshape e} \@empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use `\csname` lookups as a fast way of accessing composites.

This has an unfortunate 'misfeature' though, which is that in the T1 encoding, `\'\{aa\}` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn't affect performance too badly.

Finally, it's worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\'\{\}` then this looks up `\T1\'-\@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'\{\}` would expand to:

```
\csname \string \T1\' - \string \endcsname
```

so the `\endcsname` would be `\string'ed` and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```

144 \def\@text@composite#1#2#3\@text@composite{%
145   \expandafter\@text@composite@x
146   \csname\string#1-\string#2\endcsname}

```

Originally the `\@text@composite@x` macro had two arguments and if `#1` was not `\relax` it was executed, otherwise `#2` was executed. All this happened within the `\ifx` code so that neither `#1` nor `#2` could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo / \@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```

147 \def\@text@composite@x#1{%
148   \ifx#1\relax
149     \expandafter\@secondoftwo
150   \else
151     \expandafter\@firstoftwo
152   \fi
153   #1}

```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```

154 \catcode\z@=11\relax
155 \def\DeclareTextComposite#1#2#3#4{%
156   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
157   \bgroup
158     \lccode\z@#4%
159     \lowercase{%
160       \egroup
161       \reserved@a ^~@}}
162 \catcode\z@=15\relax
163 \@onlypreamble\DeclareTextComposite

```

(End of definition for `\DeclareTextCompositeCommand` and others.)

```

164 </2ekernel>
165 <*2ekernel | latexrelease>
166 <latexrelease>\IncludeInRelease{2019/10/01}%
167 <latexrelease>           {\UseTextAccent}{Make commands robust}%

```

`\UseTextAccent` These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

`\UseTextSymbol`

`\@use@text@encoding`

For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.

```

168 \DeclareRobustCommand*\UseTextAccent[3]{%
169   \hmode@start@before@group
170   {%

```

Turn off the group in `\UseTextSymbol` in case this is used inside the arguments of `\UseTextAccent`.

```

171     \let\hmode@start@before@group\@firstofone
172     \let\@curr@enc\cf@encoding
173     \@use@text@encoding{#1}%
174     #2{\@use@text@encoding\@curr@enc#3}%
175   }}

```

```

176 \DeclareRobustCommand*\UseTextSymbol[2]{%
177   \hmode@start@before@group
178   {%
179     \def\@wrong@font@char{\MessageBreak
180       for \noexpand\symbol'\string#2'}%
181     \@use@text@encoding{#1}%
182     #2%
183   }%
184 }

```

```

185 </2ekernel | latexrelease>
186 <latexrelease>\EndIncludeInRelease
187 <latexrelease>\IncludeInRelease{0000/00/00}%
188 <latexrelease>           {\UseTextAccent}{Make commands robust}%
189 <latexrelease>
190 <latexrelease>\kernel@make@fragile\UseTextAccent

```

```

191 <latexrelease>\kernel@make@fragile\UseTextSymbol
192 <latexrelease>
193 <latexrelease>\EndIncludeInRelease
194 <*2ekernel>

```

Switch to a different text encoding without any grouping for use in `\UseTextAccent` or `\UseTextSymbol` (and for `\oldstylenums`).

```

195 \def\@use@text@encoding#1{%
196   \edef\f@encoding{#1}%
197   \xdef\font@name{%
198     \csname\curr@fontshape/\f@size\endcsname}%
199   \pickup@font
200   \font@name
201   \@enc@update}

```

(End of definition for \UseTextAccent, \UseTextSymbol, and \@use@text@encoding.)

`\hmode@start@before@group` The `\hmode@start@before@group` starts hmode and should be immediately followed by an explicit `{...}`. Its purpose is to ensure that hmode is started before this group is opened. Inside `\add@accent` and `\UseTextAccent` it is redefined to remove this group so that it doesn't conflict with the `\accent` primitive.

For a detailed discussion see pr/3160.

```

202 \let\hmode@start@before@group\leavevmode

```

(End of definition for \hmode@start@before@group.)

`\DeclareTextSymbolDefault` Some syntactic sugar. Again, these should probably be optimized for speed.
`\DeclareTextAccentDefault`

```

203 \def\DeclareTextSymbolDefault#1#2{%
204   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
205 \def\DeclareTextAccentDefault#1#2{%
206   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
207 \@onlypreamble\DeclareTextSymbolDefault
208 \@onlypreamble\DeclareTextAccentDefault

```

(End of definition for \DeclareTextSymbolDefault and \DeclareTextAccentDefault.)

`\UndeclareTextCommand` This command safely removes an encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.

```

209 \def\UndeclareTextCommand#1#2{%

```

If there is no declaration for the current encoding do nothing.

```

210   \ifcsname #2\string#1\endcsname

```

Else: throw away that declaration.

```

211     \@enc@info{Undeclare text command \string#1 (encoding #2)}%
212     \global\expandafter\let\csname#2\string#1\endcsname
213     \undefined

```

But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command `\foo` would look similar to `\T1-cmd \foo \T1\foo` (three tokens).

Of course, instead of `T1` one could see a different encoding name; which one depends the encoding for which `\foo` was declared last.

Now assume we have just removed the declaration for `\foo` in `T1` and the top-level of `\foo` expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset `\foo` within `T1` instead of getting the default definition for `\foo`. And what is the best way to change that top-level definition? Well, the only “encoding” we know for sure will still be around is the default encoding denoted by `?`.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like `\?-cmd \foo \?\foo` which is done by the following (readable?) code:

```

214     \expandafter\expandafter\expandafter
215     \ifx\expandafter\@thirdofthree#1\@undefined
216     \expandafter\gdef\expandafter#1\expandafter
217         {\csname ?-cmd\expandafter\endcsname\expandafter
218          #1\csname?\string#1\endcsname}%
219     \fi
220 \else
221
222     \@enc@info{Text command \string#1 (encoding #2) is not declared}%
223 \fi
224 \onlypreamble\UndeclareTextCommand

```

(End of definition for \UndeclareTextCommand.)

1.4.2 Hyphenation

<code>\patterns</code> <code>\@patterns</code> <code>\hyphenation</code> <code>\@hyphenation</code>	<p>We redefine <code>\patterns</code> and <code>\hyphenation</code> to allow the use of commands declared with <code>\DeclareText*</code> to be used inside them.</p> <pre> 225 %\let\@patterns\patterns 226 %\let\@hyphenation\hyphenation 227 %\def\patterns{% 228 % \bgroup 229 % \let\protect\@empty 230 % \let\@typeset@protect\@empty 231 % \let\@changed@x\@changed@x@mouth 232 % \afterassignment\egroup 233 % \@patterns 234 %} 235 %\def\hyphenation{% 236 % \bgroup 237 % \let\protect\@empty 238 % \let\@typeset@protect\@empty 239 % \let\@changed@x\@changed@x@mouth 240 % \afterassignment\egroup 241 % \@hyphenation 242 %} </pre>
--	--

(End of definition for \patterns and others.)

1.4.3 Miscellanea

`\a` The `\a` command is used to access the accent commands even when they have been redefined (for example by the `tabbing` environment). Its internal name is `\@tabacckludge`.

The `\string` within the `\csname` guards against something like `'` being active at the point of use.

```
243 \def\@tabacckludge#1{\expandafter\@changed@cmd
244                               \csname\string#1\endcsname\relax}
245 \let\a=\@tabacckludge
```

(End of definition for \a.)

1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the T_EX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
246 \DeclareTextAccentDefault{\"}{OT1}
247 \DeclareTextAccentDefault{\'}{OT1}
248 \DeclareTextAccentDefault{\.}{OT1}
249 \DeclareTextAccentDefault{\=}{OT1}
250 \DeclareTextAccentDefault{\H}{OT1}
251 \DeclareTextAccentDefault{\^}{OT1}
252 \DeclareTextAccentDefault{\'}{OT1}
253 \DeclareTextAccentDefault{\b}{OT1}
254 \DeclareTextAccentDefault{\c}{OT1}
255 \DeclareTextAccentDefault{\d}{OT1}
256 \DeclareTextAccentDefault{\r}{OT1}
257 \DeclareTextAccentDefault{\u}{OT1}
258 \DeclareTextAccentDefault{\v}{OT1}
259 \DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
260 %\DeclareTextSymbolDefault{\AA}{OT1}
261 \DeclareTextSymbolDefault{\AE}{OT1}
262 \DeclareTextSymbolDefault{\L}{OT1}
263 \DeclareTextSymbolDefault{\OE}{OT1}
264 \DeclareTextSymbolDefault{\O}{OT1}
265 %\DeclareTextSymbolDefault{\aa}{OT1}
266 \DeclareTextSymbolDefault{\ae}{OT1}
267 \DeclareTextSymbolDefault{\i}{OT1}
```

```

268 \DeclareTextSymbolDefault{\j}{OT1}
269 \DeclareTextSymbolDefault{\ij}{OT1}
270 \DeclareTextSymbolDefault{\IJ}{OT1}
271 \DeclareTextSymbolDefault{\l}{OT1}
272 \DeclareTextSymbolDefault{\oe}{OT1}
273 \DeclareTextSymbolDefault{\o}{OT1}
274 \DeclareTextSymbolDefault{\ss}{OT1}
275 \DeclareTextSymbolDefault{\textdollar}{OT1}
276 \DeclareTextSymbolDefault{\textemdash}{OT1}
277 \DeclareTextSymbolDefault{\textendash}{OT1}
278 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
279 %\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
280 %\DeclareTextSymbolDefault{\texthyphen}{OT1}
281 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
282 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
283 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
284 \DeclareTextSymbolDefault{\textquoteleft}{OT1}
285 \DeclareTextSymbolDefault{\textquoteright}{OT1}
286 \DeclareTextSymbolDefault{\textsterling}{OT1}

```

Some symbols from OMS:

```

287 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
288 \DeclareTextSymbolDefault{\textbackslash}{OMS}
289 \DeclareTextSymbolDefault{\textbar}{OMS}
290 \DeclareTextSymbolDefault{\textbardbl}{OMS}
291 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
292 \DeclareTextSymbolDefault{\textbraceright}{OMS}
293 \DeclareTextSymbolDefault{\textbullet}{OMS}
294 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
295 \DeclareTextSymbolDefault{\textdagger}{OMS}
296 \DeclareTextSymbolDefault{\textparagraph}{OMS}
297 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
298 \DeclareTextSymbolDefault{\textsection}{OMS}
299 \DeclareTextAccentDefault{\textcircled}{OMS}

```

Some symbols from OML:

```

300 \DeclareTextSymbolDefault{\textless}{OML}
301 \DeclareTextSymbolDefault{\textgreater}{OML}
302 \DeclareTextAccentDefault{\t}{OML}

```

Some defaults we can fake.

The interface for defining `\copyright` changed, it used to use `\expandafter` to add braces at the appropriate points.

```

303 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
304 % \expandafter\def\expandafter
305 % \copyright\expandafter{\expandafter\copyright}
306 \DeclareTextCommandDefault{\textasciicircum}{\~{}}
307 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
308 \DeclareTextCommandDefault{\textunderscore}{%
309 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}

```

There is no good reason anymore to fake `\textcompwordmark`.

```

310 %\DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
311 \DeclareTextSymbolDefault{\textcompwordmark}{T1}

```

```

312 \DeclareTextCommandDefault{\textvisiblespace}{%
313   \mbox{\kern.06em\vrule \@height.3ex}%
314   \vbox{\hrule \@width.3em}%
315   \hbox{\vrule \@height.3ex}}

```

Using `\fontdimen3` in the next definition is some sort of a kludge (since it is the interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since there it is zero).

```

316 \DeclareTextCommandDefault{\textellipsis}{%
317   .\kern\fontdimen3\font
318   .\kern\fontdimen3\font
319   .\kern\fontdimen3\font}

320 %\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
321 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
322   \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
323 \DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
324 \DeclareTextCommandDefault{\SS}{SS}

325 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
326 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}

```

1.4.5 Math material

Some commands can be used in both text and math mode:

```

327 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}

```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```

328 \protected\def{\ifmmode\lbrace\else\textbraceleft\fi}
329 \protected\def\}{\ifmmode\rbrace\else\textbraceright\fi}

330 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
331 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
332 \DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
333 \DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}

```

For historical reasons `\copyright` needs `{}` around the definition in maths.

```

334 \DeclareRobustCommand{\_}{%
335   \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
336 \DeclareRobustCommand{\copyright}{%
337   \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
338 \DeclareRobustCommand{\pounds}{%
339   \ifmmode\mathsterling\else\textsterling\fi}

340 \DeclareRobustCommand{\dots}{%
341   \ifmmode\mathellipsis\else\textellipsis\fi}

342 \let\ldots\dots

```

Default definition of the commabelow accent.

```

343 </2ekernel>
344 <latexrelease>\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
345 <*2ekernel|latexrelease>
346 \DeclareTextCommandDefault\textcommabelow[1
347   {\hmode\bgroup\oalign{\null#1\crcr\hidewidth\raise-.31ex
348   \hbox{\check@mathfonts\fontsize\ssf@size\z@

```

```

349 \math@fontsfalse\selectfont,}\hidewidth}\egroup}
350 <latexrelease>\EndIncludeInRelease
351 </2ekernel| latexrelease>
352 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
353 <latexrelease>\let\textcommabelow\@undefined
354 <latexrelease>\expandafter
355 <latexrelease> \let\csname\string\T1\string\c-G\endcsname\@undefined
356 <latexrelease>\expandafter
357 <latexrelease> \let\csname\string\T1\string\c-K\endcsname\@undefined
358 <latexrelease>\expandafter
359 <latexrelease> \let\csname\string\T1\string\c-k\endcsname\@undefined
360 <latexrelease>\expandafter
361 <latexrelease> \let\csname\string\T1\string\c-L\endcsname\@undefined
362 <latexrelease>\expandafter
363 <latexrelease> \let\csname\string\T1\string\c-l\endcsname\@undefined
364 <latexrelease>\expandafter
365 <latexrelease> \let\csname\string\T1\string\c-N\endcsname\@undefined
366 <latexrelease>\expandafter
367 <latexrelease> \let\csname\string\T1\string\c-n\endcsname\@undefined
368 <latexrelease>\expandafter
369 <latexrelease> \let\csname\string\T1\string\c-R\endcsname\@undefined
370 <latexrelease>\expandafter
371 <latexrelease> \let\csname\string\T1\string\c-r\endcsname\@undefined
372 <latexrelease>\EndIncludeInRelease
    Default definition of the commaabove accent(E.G.).
373 <latexrelease>\IncludeInRelease{2016/02/01}{\textcommaabove}{comma above}%
374 <*2ekernel| latexrelease>
375 \DeclareTextCommandDefault\textcommaabove[1]{%
376 \hmode\bgroup
377 \ooalign{%
378 \hidewidth
379 \raise.7ex\hbox{%
380 \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont‘%
381 }%
382 \hidewidth\crrc
383 \null#1\crrc
384 }%
385 \egroup
386 }
387 <latexrelease>\EndIncludeInRelease
388 </2ekernel| latexrelease>
389 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommaabove}{comma above}%
390 <latexrelease>\let\textcommaabove\@undefined
391 <latexrelease>\expandafter
392 <latexrelease> \let\csname\string\OT1\string\c-g\endcsname\@undefined
393 <latexrelease>\expandafter
394 <latexrelease> \let\csname\string\T1\string\c-g\endcsname\@undefined
395 <latexrelease>\EndIncludeInRelease

```

1.5 Definitions for the OT1 encoding

The definitions for the ‘T_EX text’ (OT1) encoding.

Declare the encoding.


```

396 <*OT1>
397 \DeclareFontEncoding{OT1}{-}{}

```

Providing font substitution is essential if accents are missing in a font and substitution tries to find it elsewhere. Without it you might receive a “Corrupted NFSS tables” error.

```

398 \DeclareFontSubstitution{OT1}{cmr}{m}{n}

```

Declare the accents.

```

399 \DeclareTextAccent{"}{OT1}{127}
400 \DeclareTextAccent{'}{OT1}{19}
401 \DeclareTextAccent{.}{OT1}{95}
402 \DeclareTextAccent{=}{OT1}{22}
403 \DeclareTextAccent{^}{OT1}{94}
404 \DeclareTextAccent{'}{OT1}{18}
405 \DeclareTextAccent{\~}{OT1}{126}
406 \DeclareTextAccent{\H}{OT1}{125}
407 \DeclareTextAccent{\u}{OT1}{21}
408 \DeclareTextAccent{\v}{OT1}{20}
409 \DeclareTextAccent{\r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\ooalign` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from `plain.tex` since that now has two incompatible definitions.

```

410 \DeclareTextCommand{\b}{OT1}[1]
411   {\hmode\bgroup\o@lign{\relax#1\crrcr\hidewidth\ltx@sh@ft{-3ex}%
412     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
413 \DeclareTextCommand{\c}{OT1}[1]
414   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
415     \else\ooalign{\unhbox\z@\crrcr\hidewidth\char24\hidewidth}\fi}
416 \DeclareTextCommand{\d}{OT1}[1]
417   {\hmode\bgroup
418     \o@lign{\relax#1\crrcr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}

```

Declare the text symbols.

```

419 \DeclareTextSymbol{\AE}{OT1}{29}
420 \DeclareTextSymbol{\OE}{OT1}{30}
421 \DeclareTextSymbol{\O}{OT1}{31}
422 \DeclareTextSymbol{\ae}{OT1}{26}
423 \DeclareTextSymbol{\i}{OT1}{16}
424 \DeclareTextSymbol{\j}{OT1}{17}
425 \DeclareTextSymbol{\oe}{OT1}{27}
426 \DeclareTextSymbol{\o}{OT1}{28}
427 \DeclareTextSymbol{\ss}{OT1}{25}
428 \DeclareTextSymbol{\textendash}{OT1}{124}
429 \DeclareTextSymbol{\textendash}{OT1}{123}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

430 \DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hskip\z@}
431 \DeclareTextCommand{\textfiguredash}{OT1}{\textendash}
432 \DeclareTextCommand{\texthorizontalbar}{OT1}{\textendash}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

433 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}

```

```

434 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
435 \DeclareTextCommand{\textexclamdown}{OT1}{!'}
436 \DeclareTextCommand{\textquestiondown}{OT1}{?' }
437 %\DeclareTextSymbol{\textthyphenchar}{OT1}{'\-}
438 %\DeclareTextSymbol{\textthyphen}{OT1}{'\-}
439 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
440 \DeclareTextSymbol{\textquotedblright}{OT1}{'\"}
441 \DeclareTextSymbol{\textquoteleft}{OT1}{'\'}
442 \DeclareTextSymbol{\textquoteright}{OT1}{'\'}

```

Some symbols which are faked from others:

```

443 % \DeclareTextCommand{\aa}{OT1}
444 %   {\accent23a}
445 \DeclareTextCommand{\L}{OT1}
446   {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
447 \DeclareTextCommand{\l}{OT1}
448   {\hmode\bgroup\@xxxii l\egroup}
449 % \DeclareTextCommand{\AA}{OT1}
450 %   {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
451 %     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

452 \DeclareTextCompositeCommand{\r}{OT1}{A}
453   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
454   \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```

455 \DeclareTextCommand{\ij}{OT1}{%
456   \nobreak\hskip\z@skip i\kern-0.02em\nobreak\hskip\z@skip j}
457 \DeclareTextCommand{\IJ}{OT1}{%
458   \nobreak\hskip\z@skip I\kern-0.02em\nobreak\hskip\z@skip J}

```

In the OT1 encoding, £ and \$ share a slot.

```

459 \DeclareTextCommand{\textdollar}{OT1}{\hmode\bgroup
460   \ifdim \fontdimen\@ne\font >\z@
461     \slshape
462   \else
463     \upshape
464   \fi
465   \char'\$\egroup}
466 \DeclareTextCommand{\textsterling}{OT1}{\hmode\bgroup
467   \ifdim \fontdimen\@ne\font >\z@
468     \itshape
469   \else
470     \fontshape{ui}\selectfont
471   \fi
472   \char'\$\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true L^AT_EX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

473 \DeclareTextComposite{\.}{OT1}{i}{'\i}
474 \DeclareTextComposite{\.}{OT1}{i}{'\i}
475 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
476 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
477 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}
478 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}
479 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}

T1 encoding is given more extensive set of overloads for \c But here we just adjust
\c{g}.
480 \ifx\textcommaabove\undefined\else
481 \DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
482 \fi
483 \</OT1>

```

1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T_EX text’ (T1) encoding.

Declare the encoding.

```

484 \<*T1>
485 \DeclareFontEncoding{T1}{}{}
486 \DeclareFontSubstitution{T1}{cmr}{m}{n}

```

Declare the accents.

```

487 \DeclareTextAccent{\'}{T1}{0}
488 \DeclareTextAccent{\'}{T1}{1}
489 \DeclareTextAccent{\^}{T1}{2}
490 \DeclareTextAccent{\^}{T1}{3}
491 \DeclareTextAccent{\^}{T1}{4}
492 \DeclareTextAccent{\^}{T1}{5}
493 \DeclareTextAccent{\^}{T1}{6}
494 \DeclareTextAccent{\^}{T1}{7}
495 \DeclareTextAccent{\^}{T1}{8}
496 \DeclareTextAccent{\^}{T1}{9}
497 \DeclareTextAccent{\^}{T1}{10}

```

Some accents have to be built by hand. Note that \ooalign and \o@lign must be inside a group. In these definitions we no longer use the helper function \sh@ft from plain.tex since that now has two incompatible definitions.

```

498 \DeclareTextCommand{\b}{T1}[1]
499   {\hmode\bgroup\o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-3ex}%
500    \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
501 \DeclareTextCommand{\c}{T1}[1]
502   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
503    \else\ooalign{\unhbox\z@\cr
504     \hidewidth\char11\hidewidth}\fi}
505 \DeclareTextCommand{\d}{T1}[1]
506   {\hmode\bgroup
507    \o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
508 \DeclareTextCommand{\k}{T1}[1]
509   {\hmode\bgroup\ooalign{\null#1\cr\hidewidth\char12}\egroup}
510 \DeclareTextCommand{\textogonekcentered}{T1}[1]
511   {\hmode\bgroup\ooalign{%
512    \null#1\cr\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```

513 \DeclareTextCommand{\textperthousand}{T1}
514   {\%\char 24 }           % space or ‘relax as delimiter?
515 \DeclareTextCommand{\textpertenthousand}{T1}
516   {\%\char 24\char 24 } % space or ‘relax as delimiter?

```

For Maltese, \Hwithstroke and \hwithstroke are needed.

```

517 \DeclareTextCommand{\Hwithstroke}{T1}
518   {%
519     \hmode\bgroup
520     \vphantom{H}%
521     \sbox\z@{H}%
522     \oalign{%
523       H\cr
524       \hidewidth
525       \vrule
526         height \dimexpr 0.7\ht\z@+0.1ex\relax
527         depth  -0.7\ht\z@
528         width   0.8\wd\z@
529       \hidewidth\cr
530     }%
531   \egroup
532 }
533 \DeclareTextCommand{\hwithstroke}{T1}
534   {%
535     \hmode\bgroup
536     \vphantom{h}%
537     \sbox\z@{h}%
538     \oalign{%
539       h\cr
540       \kern0.075\wd\z@
541       \vrule
542         height \dimexpr 0.7\ht\z@+0.1ex\relax
543         depth  -0.7\ht\z@
544         width   0.4\wd\z@
545       \hidewidth\cr
546     }%
547   \egroup
548 }

```

Declare the text symbols.

```

549 %\DeclareTextSymbol{\AA}{T1}{197}
550 \DeclareTextSymbol{\AE}{T1}{198}
551 \DeclareTextSymbol{\DH}{T1}{208}
552 \DeclareTextSymbol{\DJ}{T1}{208}
553 \DeclareTextSymbol{\L}{T1}{138}
554 \DeclareTextSymbol{\NG}{T1}{141}
555 \DeclareTextSymbol{\OE}{T1}{215}
556 \DeclareTextSymbol{\O}{T1}{216}
557 \DeclareTextSymbol{\SS}{T1}{223}
558 \DeclareTextSymbol{\TH}{T1}{222}
559 %\DeclareTextSymbol{\aa}{T1}{229}
560 \DeclareTextSymbol{\ae}{T1}{230}

```

```

561 \DeclareTextSymbol{\dh}{T1}{240}
562 \DeclareTextSymbol{\dj}{T1}{158}

563 \DeclareTextSymbol{\guillemetleft}{T1}{19}
564 \DeclareTextSymbol{\guillemetright}{T1}{20}
565 % old Adobe names
566 \DeclareTextSymbol{\guillemotleft}{T1}{19}
567 \DeclareTextSymbol{\guillemotright}{T1}{20}

568 \DeclareTextSymbol{\guilsinglleft}{T1}{14}
569 \DeclareTextSymbol{\guilsinglright}{T1}{15}
570 \DeclareTextSymbol{\i}{T1}{25}
571 \DeclareTextSymbol{\j}{T1}{26}
572 \DeclareTextSymbol{\ij}{T1}{188}
573 \DeclareTextSymbol{\IJ}{T1}{156}
574 \DeclareTextSymbol{\l}{T1}{170}
575 \DeclareTextSymbol{\ng}{T1}{173}
576 \DeclareTextSymbol{\oe}{T1}{247}
577 \DeclareTextSymbol{\o}{T1}{248}
578 \DeclareTextSymbol{\quotedblbase}{T1}{18}
579 \DeclareTextSymbol{\quotesinglbase}{T1}{13}
580 \DeclareTextSymbol{\ss}{T1}{255}
581 \DeclareTextSymbol{\textasciicircum}{T1}{‘^’}
582 \DeclareTextSymbol{\textasciitilde}{T1}{‘~’}
583 \DeclareTextSymbol{\textbackslash}{T1}{‘\’}
584 \DeclareTextSymbol{\textbar}{T1}{‘|’}
585 \DeclareTextSymbol{\textbraceleft}{T1}{‘{’}
586 \DeclareTextSymbol{\textbraceright}{T1}{‘}’}
587 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
588 \DeclareTextSymbol{\textdollar}{T1}{‘$’}
589 \DeclareTextSymbol{\textendash}{T1}{22}
590 \DeclareTextSymbol{\textendash}{T1}{21}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

591 \DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
592 \DeclareTextCommand{\textfiguredash}{T1}{\textendash}
593 \DeclareTextCommand{\texthorizontalbar}{T1}{\textendash}

594 \DeclareTextSymbol{\textexclamdown}{T1}{189}
595 \DeclareTextSymbol{\textgreater}{T1}{‘>’}
596 %\DeclareTextSymbol{\textthyphenchar}{T1}{127}
597 %\DeclareTextSymbol{\textthyphen}{T1}{‘-’}
598 \DeclareTextSymbol{\textless}{T1}{‘<’}
599 \DeclareTextSymbol{\textquestiondown}{T1}{190}
600 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
601 \DeclareTextSymbol{\textquotedblright}{T1}{17}
602 \DeclareTextSymbol{\textquotedbl}{T1}{‘”’}
603 \DeclareTextSymbol{\textquoteleft}{T1}{‘‘’}
604 \DeclareTextSymbol{\textquoteright}{T1}{‘’’}
605 \DeclareTextSymbol{\textsection}{T1}{159}
606 \DeclareTextSymbol{\textsterling}{T1}{191}
607 \DeclareTextSymbol{\textunderscore}{T1}{95}
608 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
609 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

610 \DeclareTextComposite{\.}{T1}{i}{'\i}
611 \DeclareTextComposite{\.}{T1}{\i}{'\i}
"80 = 128
612 \DeclareTextComposite{\u}{T1}{A}{128}
613 \DeclareTextComposite{\k}{T1}{A}{129}
614 \DeclareTextComposite{\'}{T1}{C}{130}
615 \DeclareTextComposite{\v}{T1}{C}{131}
616 \DeclareTextComposite{\v}{T1}{D}{132}
617 \DeclareTextComposite{\v}{T1}{E}{133}
618 \DeclareTextComposite{\k}{T1}{E}{134}
619 \DeclareTextComposite{\u}{T1}{G}{135}
"88 = 136
620 \DeclareTextComposite{\'}{T1}{L}{136}
621 \DeclareTextComposite{\v}{T1}{L}{137}
622 \DeclareTextComposite{\'}{T1}{N}{139}
623 \DeclareTextComposite{\v}{T1}{N}{140}
624 \DeclareTextComposite{\H}{T1}{O}{142}
625 \DeclareTextComposite{\'}{T1}{R}{143}
"90 = 144
626 \DeclareTextComposite{\v}{T1}{R}{144}
627 \DeclareTextComposite{\'}{T1}{S}{145}
628 \DeclareTextComposite{\v}{T1}{S}{146}
629 \DeclareTextComposite{\c}{T1}{S}{147}
630 \DeclareTextComposite{\v}{T1}{T}{148}
631 \DeclareTextComposite{\c}{T1}{T}{149}
632 \DeclareTextComposite{\H}{T1}{U}{150}
633 \DeclareTextComposite{\r}{T1}{U}{151}
"98 = 152
634 \DeclareTextComposite{\"}{T1}{Y}{152}
635 \DeclareTextComposite{\'}{T1}{Z}{153}
636 \DeclareTextComposite{\v}{T1}{Z}{154}
637 \DeclareTextComposite{\.}{T1}{Z}{155}
638 \DeclareTextComposite{\.}{T1}{I}{157}
"A0 = 160
639 \DeclareTextComposite{\u}{T1}{a}{160}
640 \DeclareTextComposite{\k}{T1}{a}{161}
641 \DeclareTextComposite{\'}{T1}{c}{162}
642 \DeclareTextComposite{\v}{T1}{c}{163}
643 \DeclareTextComposite{\v}{T1}{d}{164}
644 \DeclareTextComposite{\v}{T1}{e}{165}
645 \DeclareTextComposite{\k}{T1}{e}{166}
646 \DeclareTextComposite{\u}{T1}{g}{167}
"A8 = 168
647 \DeclareTextComposite{\'}{T1}{l}{168}
648 \DeclareTextComposite{\v}{T1}{l}{169}
649 \DeclareTextComposite{\'}{T1}{n}{171}
650 \DeclareTextComposite{\v}{T1}{n}{172}
651 \DeclareTextComposite{\H}{T1}{o}{174}
652 \DeclareTextComposite{\'}{T1}{r}{175}

```

```

"B0 = 176
653 \DeclareTextComposite{\v}{T1}{r}{176}
654 \DeclareTextComposite{\'}{T1}{s}{177}
655 \DeclareTextComposite{\v}{T1}{s}{178}
656 \DeclareTextComposite{\c}{T1}{s}{179}
657 \DeclareTextComposite{\v}{T1}{t}{180}
658 \DeclareTextComposite{\c}{T1}{t}{181}
659 \DeclareTextComposite{\H}{T1}{u}{182}
660 \DeclareTextComposite{\r}{T1}{u}{183}
"B8 = 184
661 \DeclareTextComposite{\"}{T1}{y}{184}
662 \DeclareTextComposite{\'}{T1}{z}{185}
663 \DeclareTextComposite{\v}{T1}{z}{186}
664 \DeclareTextComposite{\.}{T1}{z}{187}
"C0 = 192
665 \DeclareTextComposite{\'}{T1}{A}{192}
666 \DeclareTextComposite{\'}{T1}{A}{193}
667 \DeclareTextComposite{\^}{T1}{A}{194}
668 \DeclareTextComposite{\~}{T1}{A}{195}
669 \DeclareTextComposite{\"}{T1}{A}{196}
670 \DeclareTextComposite{\r}{T1}{A}{197}
671 \DeclareTextComposite{\c}{T1}{C}{199}
"C8 = 200
672 \DeclareTextComposite{\'}{T1}{E}{200}
673 \DeclareTextComposite{\'}{T1}{E}{201}
674 \DeclareTextComposite{\^}{T1}{E}{202}
675 \DeclareTextComposite{\"}{T1}{E}{203}
676 \DeclareTextComposite{\'}{T1}{I}{204}
677 \DeclareTextComposite{\'}{T1}{I}{205}
678 \DeclareTextComposite{\^}{T1}{I}{206}
679 \DeclareTextComposite{\"}{T1}{I}{207}
"D0 = 208
680 \DeclareTextComposite{\~}{T1}{N}{209}
681 \DeclareTextComposite{\'}{T1}{O}{210}
682 \DeclareTextComposite{\'}{T1}{O}{211}
683 \DeclareTextComposite{\^}{T1}{O}{212}
684 \DeclareTextComposite{\~}{T1}{O}{213}
685 \DeclareTextComposite{\"}{T1}{O}{214}
"D8 = 216
686 \DeclareTextComposite{\'}{T1}{U}{217}
687 \DeclareTextComposite{\'}{T1}{U}{218}
688 \DeclareTextComposite{\^}{T1}{U}{219}
689 \DeclareTextComposite{\"}{T1}{U}{220}
690 \DeclareTextComposite{\'}{T1}{Y}{221}
"E0 = 224
691 \DeclareTextComposite{\'}{T1}{a}{224}
692 \DeclareTextComposite{\'}{T1}{a}{225}
693 \DeclareTextComposite{\^}{T1}{a}{226}
694 \DeclareTextComposite{\~}{T1}{a}{227}
695 \DeclareTextComposite{\"}{T1}{a}{228}

```

```

696 \DeclareTextComposite{\r}{T1}{a}{229}
697 \DeclareTextComposite{\c}{T1}{c}{231}
"E8 = 232
698 \DeclareTextComposite{\'}{T1}{e}{232}
699 \DeclareTextComposite{\'}{T1}{e}{233}
700 \DeclareTextComposite{\^}{T1}{e}{234}
701 \DeclareTextComposite{\"}{T1}{e}{235}
702 \DeclareTextComposite{\'}{T1}{i}{236}
703 \DeclareTextComposite{\'}{T1}{i}{236}
704 \DeclareTextComposite{\'}{T1}{i}{237}
705 \DeclareTextComposite{\'}{T1}{i}{237}
706 \DeclareTextComposite{\^}{T1}{i}{238}
707 \DeclareTextComposite{\^}{T1}{i}{238}
708 \DeclareTextComposite{\"}{T1}{i}{239}
709 \DeclareTextComposite{\"}{T1}{i}{239}
"F0 = 240
710 \DeclareTextComposite{\~}{T1}{n}{241}
711 \DeclareTextComposite{\'}{T1}{o}{242}
712 \DeclareTextComposite{\'}{T1}{o}{243}
713 \DeclareTextComposite{\^}{T1}{o}{244}
714 \DeclareTextComposite{\~}{T1}{o}{245}
715 \DeclareTextComposite{\"}{T1}{o}{246}
"F8 = 248
716 \DeclareTextComposite{\'}{T1}{u}{249}
717 \DeclareTextComposite{\'}{T1}{u}{250}
718 \DeclareTextComposite{\^}{T1}{u}{251}
719 \DeclareTextComposite{\"}{T1}{u}{252}
720 \DeclareTextComposite{\'}{T1}{y}{253}

721 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
722 \DeclareTextCompositeCommand{\k}{T1}{0}{\textogonekcentered{0}}

723 \ifx\textcommaabove\undefined\else
724 \DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
725 \fi
726 \ifx\textcommabelow\undefined\else
727 \DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
728 \DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
729 \DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
730 \DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
731 \DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
732 \DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
733 \DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
734 \DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
735 \DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
736 \fi

One oddity.
737 \DeclareTextCompositeCommand{\=}{T1}{i}{\=i}
738 \end{document}

```


1.7 Definitions for the OMS encoding

The definitions for the ‘ $\TeX$ math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard $\LaTeX$ text symbols.

Declare the encoding.

```
739 ⟨*OMS⟩
740 \DeclareFontEncoding{OMS}{-}{-}

Declare the symbols. Note that slot 13 has in places been named \Orb: please root out
and destroy this impolity wherever you find it!

741 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3}      % "03
742 \DeclareTextSymbol{\textbackslash}{OMS}{110}          % "6E
743 \DeclareTextSymbol{\textbar}{OMS}{106}                 % "6A
744 \DeclareTextSymbol{\textbardbl}{OMS}{107}              % "6B
745 \DeclareTextSymbol{\textbraceleft}{OMS}{102}           % "66
746 \DeclareTextSymbol{\textbraceright}{OMS}{103}          % "67
747 \DeclareTextSymbol{\textbullet}{OMS}{15}               % "0F
748 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122}           % "7A
749 \DeclareTextSymbol{\textdagger}{OMS}{121}              % "79
750 \DeclareTextSymbol{\textparagraph}{OMS}{123}           % "7B
751 \DeclareTextSymbol{\textperiodcentered}{OMS}{1}        % "01
752 \DeclareTextSymbol{\textsection}{OMS}{120}             % "78
753 \DeclareTextSymbol{\textbigcircle}{OMS}{13}            % "0D
754 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode\bgroup
755   \oalign{%
756     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
757     \char 13 % "0D
758   }%
759 \egroup}
760 ⟨/OMS⟩
```

1.8 Definitions for the OML encoding

The definitions for the ‘ $\TeX$ math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard $\LaTeX$ text symbols.

Declare the encoding.

```
761 ⟨*OML⟩
762 \DeclareFontEncoding{OML}{-}{-}

Declare the symbols.

763 \DeclareTextSymbol{\textless}{OML}{‘\<}
764 \DeclareTextSymbol{\textgreater}{OML}{‘\>}
765 \DeclareTextAccent{\t}{OML}{127} % "7F
766 ⟨/OML⟩
```

1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ $\TeX$ text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The $\LaTeX$ support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:
 Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;
 Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.
 Declare the encoding.

```

767 <*OT4>
768 \DeclareFontEncoding{OT4}{-}{-}
769 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```

770 \DeclareTextAccent{"}{OT4}{127}
771 \DeclareTextAccent{'}{OT4}{19}
772 \DeclareTextAccent{.}{OT4}{95}
773 \DeclareTextAccent{=}{OT4}{22}
774 \DeclareTextAccent{^}{OT4}{94}
775 \DeclareTextAccent{'}{OT4}{18}
776 \DeclareTextAccent{~}{OT4}{126}
777 \DeclareTextAccent{H}{OT4}{125}
778 \DeclareTextAccent{u}{OT4}{21}
779 \DeclareTextAccent{v}{OT4}{20}
780 \DeclareTextAccent{r}{OT4}{23}

```

The ogonek accent is available only under a e A & E. But we have to provide some definition for \k. Some other accents have to be built by hand as in OT1:

```

781 \DeclareTextCommand{\k}{OT4}[1]{%
782   \TextSymbolUnavailable{\k{#1}}#1}

```

In these definitions we no longer use the helper function \sh@ft from plain.tex since that now has two incompatible definitions.

```

783 \DeclareTextCommand{\b}{OT4}[1]
784   {\hmode\bgroup\o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-3ex}%
785     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
786 \DeclareTextCommand{\c}{OT4}[1]
787   {\leavevmode\setbox\z@{\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
788     \else\o@lign{\unhbox\z@\cr\hidewidth\char24\hidewidth}}\fi}
789 \DeclareTextCommand{\d}{OT4}[1]
790   {\hmode\bgroup
791     \o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}

```

Declare the text symbols.

```

792 \DeclareTextSymbol{\AE}{OT4}{29}
793 \DeclareTextSymbol{\OE}{OT4}{30}
794 \DeclareTextSymbol{\O}{OT4}{31}
795 \DeclareTextSymbol{\L}{OT4}{138}
796 \DeclareTextSymbol{\ae}{OT4}{26}

797 \DeclareTextSymbol{\guillemetleft}{OT4}{174}
798 \DeclareTextSymbol{\guillemetright}{OT4}{175}
799 % old Adobe names
800 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
801 \DeclareTextSymbol{\guillemotright}{OT4}{175}

802 \DeclareTextSymbol{\i}{OT4}{16}
803 \DeclareTextSymbol{\j}{OT4}{17}
804 \DeclareTextSymbol{\l}{OT4}{170}
805 \DeclareTextSymbol{\o}{OT4}{28}
806 \DeclareTextSymbol{\oe}{OT4}{27}

```

```

807 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
808 \DeclareTextSymbol{\ss}{OT4}{25}
809 \DeclareTextSymbol{\textemdash}{OT4}{124}
810 \DeclareTextSymbol{\textendash}{OT4}{123}
811 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
812 %\DeclareTextSymbol{\textthyphenchar}{OT4}{'\-}
813 %\DeclareTextSymbol{\textthyphen}{OT4}{'\-}
814 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
815 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
816 \DeclareTextSymbol{\textquotedblright}{OT4}{'\'}
817 \DeclareTextSymbol{\textquoteleft}{OT4}{'\'}
818 \DeclareTextSymbol{\textquoteright}{OT4}{'\'}

```

Definition for Å as in OT1:

```

819 \DeclareTextCompositeCommand{\r}{OT4}{A}
820   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
821     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT4 encoding, £ and \$ share a slot.

```

822 \DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
823   \ifdim \fontdimen\@ne\font >\z@
824     \slshape
825   \else
826     \upshape
827   \fi
828   \char'\$ \egroup}
829 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
830   \ifdim \fontdimen\@ne\font >\z@
831     \itshape
832   \else
833     \fontshape{ui}\selectfont
834   \fi
835   \char'\$ \egroup}

```

Declare the composites.

```

836 \DeclareTextComposite{\k}{OT4}{A}{129}
837 \DeclareTextComposite{\'}{OT4}{C}{130}
838 \DeclareTextComposite{\k}{OT4}{E}{134}
839 \DeclareTextComposite{\'}{OT4}{N}{139}
840 \DeclareTextComposite{\'}{OT4}{S}{145}
841 \DeclareTextComposite{\'}{OT4}{Z}{153}
842 \DeclareTextComposite{\.}{OT4}{Z}{155}
843 \DeclareTextComposite{\k}{OT4}{a}{161}
844 \DeclareTextComposite{\'}{OT4}{c}{162}
845 \DeclareTextComposite{\k}{OT4}{e}{166}
846 \DeclareTextComposite{\'}{OT4}{n}{171}
847 \DeclareTextComposite{\'}{OT4}{s}{177}
848 \DeclareTextComposite{\'}{OT4}{z}{185}
849 \DeclareTextComposite{\.}{OT4}{z}{187}
850 \DeclareTextComposite{\'}{OT4}{0}{211}
851 \DeclareTextComposite{\'}{OT4}{o}{243}
852 </OT4>

```

1.10 Definitions for the TS1 encoding

```
853 <*TS1>
854 \DeclareFontEncoding{TS1}{-}{-}
855 \DeclareFontSubstitution{TS1}{cmr}{m}{n}
```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group.

```
856 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
857   {\hmode\bgroup
858     \ooalign{\null#1\crrc\hidewidth\char11\hidewidth}\egroup}
859 \DeclareTextCommand{\capitalogonek}{TS1}[1]
860   {\hmode\bgroup
861     \ooalign{\null#1\crrc\hidewidth\char12\hidewidth}\egroup}
```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type

```
\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}
```

None of the latter definitions are provided by default, since they are probably rarely used.

"00 = 0

```
862 \DeclareTextAccent{\capitalgrave}{TS1}{0}
863 \DeclareTextAccent{\capitalacute}{TS1}{1}
864 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}
865 \DeclareTextAccent{\capitaltilde}{TS1}{3}
866 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
867 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
868 \DeclareTextAccent{\capitalring}{TS1}{6}
869 \DeclareTextAccent{\capitalcaron}{TS1}{7}
```

"08 = 8

```
870 \DeclareTextAccent{\capitalbreve}{TS1}{8}
871 \DeclareTextAccent{\capitalmacron}{TS1}{9}
872 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}
```

Tie accents.

The tie accent was borrowed from the `cmmi` font. The `tc` fonts now provide four tie accents, the first two are done in the classical way with asymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

" =

```
873 \DeclareTextAccent{\t}{TS1}{26}
874 \DeclareTextAccent{\capitaltie}{TS1}{27}
875 \DeclareTextAccent{\newtie}{TS1}{28}
876 \DeclareTextAccent{\capitalnewtie}{TS1}{29}
```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```
877 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
878 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}
```

The text companion symbols.

```
879 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
```

"10 = 16

```
880 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
881 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
882 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

"18 = 24

```
883 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
884 \DeclareTextSymbol{\textrightarrow}{TS1}{25}
```

"20 = 32

```
885 \DeclareTextSymbol{\textblank}{TS1}{32}
886 \DeclareTextSymbol{\textdollar}{TS1}{36}
887 \DeclareTextSymbol{\textquotesingle}{TS1}{39}
```

"28 = 40

The symbol `\textasteriskcentered` “*” is supposed to be always available in TS1 and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man’s version by using a normal “*” slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```
888 %\DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that’s wanted
889 \DeclareTextCommand \textasteriskcentered{TS1}{%      % and that’s needed
890   \iffontchar\font 42 \char42 \else
891   \begingroup\fontencoding{T1}%
892     \fontsize
893     {\the\dimexpr1.3\dimexpr\fontsize pt\relax}%
894     {\f@baselineskip}%
895     \selectfont
896     \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
897   \endgroup
898   \fi
899 }
```

Note that ’054 is a comma and ’056 is a full stop: these make numbers using oldstyle digits easier to input.

```
900 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
901 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}
```

Oldstyle digits.

"30 = 48

```
902 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
903 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
904 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
905 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
906 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
907 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
908 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
909 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
```

"38 = 56

```
910 \DeclareTextSymbol{\teightoldstyle}{TS1}{56}
911 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
```

More text companion symbols.

```
912 \DeclareTextSymbol{\textlangle}{TS1}{60}
913 \DeclareTextSymbol{\textminus}{TS1}{61}
914 \DeclareTextSymbol{\textrangle}{TS1}{62}
```

"48 = 72

```
915 \DeclareTextSymbol{\textmho}{TS1}{77}
```

The big circle is here to define the command `\textcircled`. Formerly it was taken from the `cmsy` font.

```
916 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
917 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode\bgroup
918   \oalign{%
919     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
920     \char 79   % '117 = "4F
921   }%
922 \egroup}
```

More text companion symbols.

"50 = 80

```
923 \DeclareTextSymbol{\textohm}{TS1}{87}
```

"58 = 88

```
924 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
925 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
926 \DeclareTextSymbol{\textuparrow}{TS1}{94}
927 \DeclareTextSymbol{\textdownarrow}{TS1}{95}
```

"60 = 96

```
928 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
929 \DeclareTextSymbol{\textborn}{TS1}{98}
930 \DeclareTextSymbol{\textdivorced}{TS1}{99}
931 \DeclareTextSymbol{\textdied}{TS1}{100}
```

"68 = 104

```
932 \DeclareTextSymbol{\textleaf}{TS1}{108}
933 \DeclareTextSymbol{\textmarried}{TS1}{109}
934 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}
```

"78 = 120

```
935 \DeclareTextSymbol{\texttildebelow}{TS1}{126}
```

This glyph, `\textdblhyphenchar` is hanging, like the `hyphenchar` of the `ec` fonts.

```
936 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
```

"80 = 128

```
937 \DeclareTextSymbol{\textasciibreve}{TS1}{128}
938 \DeclareTextSymbol{\textasciicaron}{TS1}{129}
```

This next glyph is *not* the same as `\textquotedbl`.

```
939 \DeclareTextSymbol{\textacutedbl}{TS1}{130}
940 \DeclareTextSymbol{\textgravedbl}{TS1}{131}
941 \DeclareTextSymbol{\textdagger}{TS1}{132}
942 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
943 \DeclareTextSymbol{\textbardbl}{TS1}{134}
944 \DeclareTextSymbol{\textperthousand}{TS1}{135}
```

"88 = 136

```
945 \DeclareTextSymbol{\textbullet}{TS1}{136}
946 \DeclareTextSymbol{\textcelsius}{TS1}{137}
947 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
948 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
949 \DeclareTextSymbol{\textflorin}{TS1}{140}
950 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
951 \DeclareTextSymbol{\textwon}{TS1}{142}
952 \DeclareTextSymbol{\textnaira}{TS1}{143}
```

"90 = 144

```
953 \DeclareTextSymbol{\textguarani}{TS1}{144}
954 \DeclareTextSymbol{\textpeso}{TS1}{145}
955 \DeclareTextSymbol{\textlira}{TS1}{146}
956 \DeclareTextSymbol{\textrecipe}{TS1}{147}
957 \DeclareTextSymbol{\textinterrobang}{TS1}{148}
958 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
959 \DeclareTextSymbol{\textdong}{TS1}{150}
960 \DeclareTextSymbol{\texttrademark}{TS1}{151}
```

"98 = 152

```
961 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}
962 \DeclareTextSymbol{\textpilcrow}{TS1}{153}
963 \DeclareTextSymbol{\textbaht}{TS1}{154}
964 \DeclareTextSymbol{\textnumero}{TS1}{155}
```

This next name may change. For the following sign we know only a german name, which is abzüglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is \textdiscount.

```
965 \DeclareTextSymbol{\textdiscount}{TS1}{156}
966 \DeclareTextSymbol{\textestimated}{TS1}{157}
967 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
968 \DeclareTextSymbol{\textservicemark}{TS1}{159}
```

"A0 = 160

```
969 \DeclareTextSymbol{\textlquill}{TS1}{160}
970 \DeclareTextSymbol{\textrquill}{TS1}{161}
971 \DeclareTextSymbol{\textcent}{TS1}{162}
972 \DeclareTextSymbol{\textsterling}{TS1}{163}
973 \DeclareTextSymbol{\textcurrency}{TS1}{164}
974 \DeclareTextSymbol{\textyen}{TS1}{165}
975 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
976 \DeclareTextSymbol{\textsection}{TS1}{167}
```

"A8 = 168

```
977 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
978 \DeclareTextSymbol{\textcopyright}{TS1}{169}
979 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
980 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
981 \DeclareTextSymbol{\textlnot}{TS1}{172}
```

The meaning of the circled-P is “sound recording copyright”.

```
982 \DeclareTextSymbol{\textcircledP}{TS1}{173}
983 \DeclareTextSymbol{\textregistered}{TS1}{174}
984 \DeclareTextSymbol{\textasciimacron}{TS1}{175}
```

```

"B0 = 176
985 \DeclareTextSymbol{\textdegree}{TS1}{176}
986 \DeclareTextSymbol{\textpm}{TS1}{177}
987 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
988 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
989 \DeclareTextSymbol{\textasciicute}{TS1}{180}
990 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
991 \DeclareTextSymbol{\textparagraph}{TS1}{182}
992 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}
"B8 = 184
993 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
994 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
995 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
996 \DeclareTextSymbol{\textsurd}{TS1}{187}
997 \DeclareTextSymbol{\textonequarter}{TS1}{188}
998 \DeclareTextSymbol{\textonehalf}{TS1}{189}
999 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
1000 \DeclareTextSymbol{\texteuro}{TS1}{191}
"E0 = 208
1001 \DeclareTextSymbol{\texttimes}{TS1}{214}
"F0 = 240
1002 \DeclareTextSymbol{\textdiv}{TS1}{246}
1003 </TS1>

```

1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here Cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted from the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to L^AT_EX LICR commands.

A few additional names for character which were supported in the original `fontspec` version of this file have also been added, even though they are not currently in the default `inputenc` UTF-8 declarations.

```

1004 < *TU>

```

In the base interface the Unicode encoding is always known as TU But we parameterize the encoding name to allow for modelling differences in Unicode support by different fonts.

```

1005 \providecommand\UnicodeEncodingName{TU}

```


TU is only currently available with XeTeX, HiTeX or LuaTeX, we detect these engines first, and make adjustments for the differing font loading syntax. For other engines, we issue a warning then abort this file, switching back to T1 encoding. Use `\Umathcode` as the marker for TU support (as in `fonttext.ltx`). Here we actually test for `\tex_Umathcode:D` as `\Umathcode` is redefined locally in the L^AT_EX test suite.

```
1006 \expandafter\ifx\csname tex_Umathcode:D\endcsname\relax
1007   \expandafter\let\csname tex_Umathcode:D\endcsname\undefined
```

Not HiTeX, LuaTeX or XeTeX, abort with a warning.

```
1008   \PackageWarningNoLine{fontenc}
1009     {\UnicodeEncodingName\space
1010      encoding is only available with\MessageBreak
1011      HiTeX, XeTeX and LuaTeX.\MessageBreak
1012      Defaulting to T1 encoding}
1013   \def\encodingdefault{T1}
1014   \expandafter\endinput
1015 \fi
```

LuaTeX.

```
1016 \ifx\directlua\@undefined\else
```

For LuaTeX 1.10+, define a Lua function to disable any handling by the font code. Otherwise we reload the font without TeX ligatures.

```
1017   \def\UnicodeFontTeXLigatures{+tlig;}
1018   \ifnum\luatexversion<110
1019     \def\reserved@a#1{%
1020       \def\@remove@tlig##1{\@remove@tlig@##1\@nil#1\@nil\relax}
1021       \def\@remove@tlig@##1#1{\@remove@tlig@@##1}}
1022     \edef\reserved@b{\detokenize{+tlig;}}
1023     \expandafter\reserved@a\expandafter{\reserved@b}
1024     \def\@remove@tlig@@#1\@nil#2\relax{#1}
1025     \def\remove@tlig#1{%
1026       \begingroup
1027       \font\remove@tlig
1028       \expandafter\@remove@tlig\expandafter{\fontname\font}%
1029       \remove@tlig
1030       \char#1\relax
1031       \endgroup
1032     }
1033   \else
1034     \newprotectedluacmd\@remove@tlig@@@
```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```
1035   \now@and@everyjob{\directlua{
1036     local rawchar_func = token.create'\@remove@tlig@@@'.index
1037     local forcehmode = tex.forcehmode
1038     local put_next = token.put_next
1039     local glyph_id = node.id'glyph'
1040     local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
1041     lua.get_functions_table()[rawchar_func] = function()
1042       local mode = tex.nest.top.mode
```

```

1043         if mode == 1 or mode == -1 then
1044             put_next(rawchar_token)
1045             return forcehmode(true)
1046         end
1047         local n = node.new(glyph_id, 256)
1048         n.font = font.current()
1049         n.char = token.scan_int()
1050         return node.write(n)
1051     end
1052 }

```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```

1053     \def\remove@tlig#1{\@remove@tlig@@@#1\relax}
1054     \fi
1055 \fi
1056
1057 XeTeX
1058 \ifx\XeTeXversion\@undefined\else
1059     \def\UnicodeFontTeXLigatures{mapping=tex-text;}
1060     \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1061 \fi
1062
1063 HiTeX
1064 \ifx\HiTeXversion\@undefined\else
1065     \def\UnicodeFontTeXLigatures{+tlig;}
1066     \def\remove@tlig#1{\mbox{\Uchar#1}}
1067 \fi
1068
1069 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1070 \def\UnicodeFontName#1#2{"#1:#2"}
1071
1072 Declare the encoding
1073 \DeclareFontEncoding\UnicodeEncodingName{}{}
1074
1075 Declare accent command to use a postpended combining character rather than the
1076 TeX \accent primitive
1077 \def\add@unicode@accent#1#2{%
1078     \if\relax\detokenize{#2}\relax~^a0\else#2\fi
1079     \char#1\relax}

```

In its original implementation `\DeclareUnicodeAccent` was given 3 arguments (with second the “Unicode encoding” a.k.a., `\UnicodeEncodingName`) while in other places, e.g., `\DeclareUnicodeComposite`, we always made encoding implicit. So we now change it here to implicit too so that the interfaces become a bit more consistent. To avoid making that a breaking change (even though it only affects two packages on CTAN) we test for `#2` being `\UnicodeEncodingName`. This would not catch if somebody used `\DeclareUnicodeAccent{=}{TU-sub}{“0304}` but that fortunately hasn’t happened. With the implicit argument you would need to change `\UnicodeEncodingName` instead, as you have to do anyway for the other interface commands.

```

1070 \def\DeclareUnicodeAccent#1#2{%
1071     \edef\reserved@a{#2}%
1072     \edef\reserved@b{\UnicodeEncodingName}%
1073     \ifx\reserved@a\reserved@b
1074         \def\reserved@a{\DeclareUnicodeAccent@{#1}}%

```

```

1075 \else
1076 \def\reserved@a{\DeclareUnicodeAccent@#1\UnicodeEncodingName}%
1077 \fi
1078 \reserved@a{#2}%
1079 }
1080 \def\DeclareUnicodeAccent@#1#2#3{%
1081 \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
1082 }

```

Wrapper around `\DeclareTextCompositeCommand` that uses the declared composite if it exists in the current font or falls back to the default definition for the TU accent if not.

```

1083 {
1084 \catcode\z@=11\relax
1085 \gdef\DeclareUnicodeComposite#1#2#3{%
1086 \def\reserved@a##1##2{%
1087 \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1088 \iffontchar\font#3 ##2%
1089 \else ##1\fi}}%
1090 \expandafter\expandafter\expandafter\extract@default@composite
1091 \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1092 \bgroup
1093 \lccode\z@#3 %
1094 \lowercase{\egroup
1095 \expandafter\reserved@a\expandafter{\reserved@b{^^@}}}%
1096 }
1097 \def\extract@default@composite#1{%
1098 \ifx\@text@composite#1%
1099 \expandafter\extract@default@composite@a
1100 \else
1101 \expandafter\extract@default@composite@b\expandafter#1%
1102 \fi}
1103 \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1104 \def\reserved@b{#2}}
1105 \def\extract@default@composite@b#1#2\@nil{%
1106 \def\reserved@b{#1#2}}

```

Next two commands are simply syntactic sugar to go with the other `\DeclareUnicode...` declarations.

```

1107 \def\DeclareUnicodeSymbol#1{\DeclareTextSymbol{#1}{\UnicodeEncodingName}}
1108 \def\DeclareUnicodeCommand#1{\DeclareTextCommand{#1}{\UnicodeEncodingName}}
1109 \DeclareUnicodeCommand\textquotesingle {\remove@tlig{"0027}}
1110 \DeclareUnicodeCommand\textasciigrave {\remove@tlig{"0060}}
1111 \DeclareUnicodeCommand\textquotedbl {\remove@tlig{"0022}}
1112 \DeclareUnicodeSymbol{\textdollar} {"0024}
1113 \DeclareUnicodeSymbol{\textless} {"003C}
1114 \DeclareUnicodeSymbol{\textgreater} {"003E}
1115 \DeclareUnicodeSymbol{\textbackslash} {"005C}
1116 \DeclareUnicodeSymbol{\textasciicircum} {"005E}
1117 \DeclareUnicodeSymbol{\textunderscore} {"005F}
1118 \DeclareUnicodeSymbol{\textbraceleft} {"007B}
1119 \DeclareUnicodeSymbol{\textbar} {"007C}
1120 \DeclareUnicodeSymbol{\textbraceright} {"007D}

```

1121	\DeclareUnicodeSymbol{\textasciitilde}	{"007E}
1122	\DeclareUnicodeSymbol{\textexclamdown}	{"00A1}
1123	\DeclareUnicodeSymbol{\textcent}	{"00A2}
1124	\DeclareUnicodeSymbol{\textsterling}	{"00A3}
1125	\DeclareUnicodeSymbol{\textcurrency}	{"00A4}
1126	\DeclareUnicodeSymbol{\textyen}	{"00A5}
1127	\DeclareUnicodeSymbol{\textbrokenbar}	{"00A6}
1128	\DeclareUnicodeSymbol{\textsection}	{"00A7}
1129	\DeclareUnicodeSymbol{\textasciidieresis}	{"00A8}
1130	\DeclareUnicodeSymbol{\textcopyright}	{"00A9}
1131	\DeclareUnicodeSymbol{\textordfeminine}	{"00AA}
1132	\DeclareUnicodeSymbol{\guillemetleft}	{"00AB}
1133	% old Adobe name	
1134	\DeclareUnicodeSymbol{\guillemotleft}	{"00AB}
1135	\DeclareUnicodeSymbol{\textlnot}	{"00AC}
1136	\DeclareUnicodeSymbol{\textregistered}	{"00AE}
1137	\DeclareUnicodeSymbol{\textasciimacron}	{"00AF}
1138	\DeclareUnicodeSymbol{\textdegree}	{"00B0}
1139	\DeclareUnicodeSymbol{\textpm}	{"00B1}
1140	\DeclareUnicodeSymbol{\texttwosuperior}	{"00B2}
1141	\DeclareUnicodeSymbol{\textthreesuperior}	{"00B3}
1142	\DeclareUnicodeSymbol{\textasciiacute}	{"00B4}
1143	\DeclareUnicodeSymbol{\textmu}	{"00B5}
1144	\DeclareUnicodeSymbol{\textparagraph}	{"00B6}
1145	\DeclareUnicodeSymbol{\textperiodcentered}	{"00B7}
1146	\DeclareUnicodeSymbol{\textonesuperior}	{"00B9}
1147	\DeclareUnicodeSymbol{\textordmasculine}	{"00BA}
1148	\DeclareUnicodeSymbol{\guillemetright}	{"00BB}
1149	% old Adobe name	
1150	\DeclareUnicodeSymbol{\guillemotright}	{"00BB}
1151	\DeclareUnicodeSymbol{\textonequarter}	{"00BC}
1152	\DeclareUnicodeSymbol{\textonehalf}	{"00BD}
1153	\DeclareUnicodeSymbol{\textthreequarters}	{"00BE}
1154	\DeclareUnicodeSymbol{\textquestiondown}	{"00BF}
1155	\DeclareUnicodeSymbol{\AE}	{"00C6}
1156	\DeclareUnicodeSymbol{\DH}	{"00D0}
1157	\DeclareUnicodeSymbol{\texttimes}	{"00D7}
1158	\DeclareUnicodeSymbol{\O}	{"00D8}
1159	\DeclareUnicodeSymbol{\TH}	{"00DE}
1160	\DeclareUnicodeSymbol{\ss}	{"00DF}
1161	\DeclareUnicodeSymbol{\ae}	{"00E6}
1162	\DeclareUnicodeSymbol{\dh}	{"00F0}
1163	\DeclareUnicodeSymbol{\textdiv}	{"00F7}
1164	\DeclareUnicodeSymbol{\o}	{"00F8}
1165	\DeclareUnicodeSymbol{\th}	{"00FE}
1166	\DeclareUnicodeSymbol{\DJ}	{"0110}
1167	\DeclareUnicodeSymbol{\dj}	{"0111}
1168	\DeclareUnicodeSymbol{\i}	{"0131}
1169	\DeclareUnicodeSymbol{\IJ}	{"0132}
1170	\DeclareUnicodeSymbol{\ij}	{"0133}
1171	\DeclareUnicodeSymbol{\L}	{"0141}
1172	\DeclareUnicodeSymbol{\l}	{"0142}

```

1173 \DeclareUnicodeSymbol{\NG} {"014A}
1174 \DeclareUnicodeSymbol{\ng} {"014B}
1175 \DeclareUnicodeSymbol{\OE} {"0152}
1176 \DeclareUnicodeSymbol{\oe} {"0153}
1177 \DeclareUnicodeSymbol{\textflorin} {"0192}
1178 \DeclareUnicodeSymbol{\j} {"0237}
1179 \DeclareUnicodeSymbol{\textasciicaron} {"02C7}
1180 \DeclareUnicodeSymbol{\textasciibreve} {"02D8}
1181 \DeclareUnicodeSymbol{\textacutedbl} {"02DD}
1182 \DeclareUnicodeSymbol{\textgravedbl} {"02F5}
1183 \DeclareUnicodeSymbol{\texttildelow} {"02F7}
1184 \DeclareUnicodeSymbol{\textbaht} {"0E3F}
1185 \DeclareUnicodeSymbol{\SS} {"1E9E}
1186 \DeclareUnicodeSymbol{\textcompwordmark} {"200C}

1187 %\DeclareUnicodeSymbol{\textnonbreakinghyphen} {"2011}
1188 %\DeclareUnicodeSymbol{\textfiguredash} {"2012}
1189 \DeclareUnicodeSymbol{\textendash} {"2013}
1190 \DeclareUnicodeSymbol{\textemdash} {"2014}
1191 %\DeclareUnicodeSymbol{\texthorizontalbar} {"2015}

```

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the L^AT_EX default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for OT1 and T1.

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

1192 \DeclareUnicodeCommand{\textnonbreakinghyphen}
1193     {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hskip\z@ \fi}
1194 \DeclareUnicodeCommand{\textfiguredash}
1195     {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1196 \DeclareUnicodeCommand{\texthorizontalbar}
1197     {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1198 \DeclareUnicodeSymbol{\textbardbl} {"2016}
1199 \DeclareUnicodeSymbol{\textquoteleft} {"2018}
1200 \DeclareUnicodeSymbol{\textquoteright} {"2019}
1201 \DeclareUnicodeSymbol{\quotesinglbase} {"201A}
1202 \DeclareUnicodeSymbol{\textquotedblleft} {"201C}
1203 \DeclareUnicodeSymbol{\textquotedblright} {"201D}
1204 \DeclareUnicodeSymbol{\quotedblbase} {"201E}
1205 \DeclareUnicodeSymbol{\textdagger} {"2020}
1206 \DeclareUnicodeSymbol{\textdaggerdbl} {"2021}
1207 \DeclareUnicodeSymbol{\textbullet} {"2022}
1208 \DeclareUnicodeSymbol{\textellipsis} {"2026}
1209 \DeclareUnicodeSymbol{\textperthousand} {"2030}
1210 \DeclareUnicodeSymbol{\textpertenthousand} {"2031}
1211 \DeclareUnicodeSymbol{\guilsinglleft} {"2039}
1212 \DeclareUnicodeSymbol{\guilsinglright} {"203A}
1213 \DeclareUnicodeSymbol{\textreferencemark} {"203B}
1214 \DeclareUnicodeSymbol{\textinterrobang} {"203D}
1215 \DeclareUnicodeSymbol{\textfractionsolidus} {"2044}
1216 \DeclareUnicodeSymbol{\textlquill} {"2045}
1217 \DeclareUnicodeSymbol{\textrquill} {"2046}
1218 \DeclareUnicodeSymbol{\textdiscount} {"2052}
1219 \DeclareUnicodeSymbol{\textcolonmonetary} {"20A1}

```

```

1220 \DeclareUnicodeSymbol{\textlira} {"20A4}
1221 \DeclareUnicodeSymbol{\textnaira} {"20A6}
1222 \DeclareUnicodeSymbol{\textwon} {"20A9}
1223 \DeclareUnicodeSymbol{\textdong} {"20AB}
1224 \DeclareUnicodeSymbol{\texteuro} {"20AC}
1225 \DeclareUnicodeSymbol{\textpeso} {"20B1}
1226 \DeclareUnicodeSymbol{\textcelsius} {"2103}
1227 \DeclareUnicodeSymbol{\textnumero} {"2116}
1228 \DeclareUnicodeSymbol{\textcircledP} {"2117}
1229 \DeclareUnicodeSymbol{\textrecipe} {"211E}
1230 \DeclareUnicodeSymbol{\textservicemark} {"2120}
1231 \DeclareUnicodeSymbol{\texttrademark} {"2122}
1232 \DeclareUnicodeSymbol{\textohm} {"2126}
1233 \DeclareUnicodeSymbol{\textmho} {"2127}
1234 \DeclareUnicodeSymbol{\textestimated} {"212E}
1235 \DeclareUnicodeSymbol{\textleftarrow} {"2190}
1236 \DeclareUnicodeSymbol{\textuparrow} {"2191}
1237 \DeclareUnicodeSymbol{\textrightarrow} {"2192}
1238 \DeclareUnicodeSymbol{\textdownarrow} {"2193}
1239 \DeclareUnicodeSymbol{\textminus} {"2212}
1240

```

```

1241 \DeclareUnicodeSymbol{\Hwithstroke} {"0126}
1242 \DeclareUnicodeSymbol{\hwithstroke} {"0127}

```

Not all fonts have U+2217 but using U+002A requires some adjustment.

```

1243 \DeclareUnicodeCommand{\textasteriskcentered}{%
1244   \iffontchar\font"2217 \char"2217 \else
1245     \begingroup
1246       \fontsize
1247       {\the\dimexpr1.3\dimexpr\fontsize pt\relax}%
1248       {\font\baselineskip}%
1249       \selectfont
1250       \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
1251     \endgroup
1252   \fi
1253 }

```

```

1254 \DeclareUnicodeSymbol{\textsurd} {"221A}
1255 \DeclareUnicodeSymbol{\textlangle} {"2329}
1256 \DeclareUnicodeSymbol{\textrangle} {"232A}
1257 \DeclareUnicodeSymbol{\textblank} {"2422}
1258 \DeclareUnicodeSymbol{\textvisiblespace} {"2423}
1259 \DeclareUnicodeSymbol{\textopenbullet} {"25E6}
1260 \DeclareUnicodeSymbol{\textbigcircle} {"25EF}
1261 \DeclareUnicodeSymbol{\textmusicalnote} {"266A}
1262 \DeclareUnicodeSymbol{\textmarried} {"26AD}
1263 \DeclareUnicodeSymbol{\textdivorced} {"26AE}
1264 \DeclareUnicodeSymbol{\textinterrobangdown} {"2E18}

```

Accents must be declared before the composites that use them.

```

1265 \DeclareUnicodeAccent{\' }{"0300}
1266 \DeclareUnicodeAccent{\' }{"0301}
1267 \DeclareUnicodeAccent{\^ }{"0302}
1268 \DeclareUnicodeAccent{\~ }{"0303}

```

```

1269 \DeclareUnicodeAccent{\=}{\0304}
1270 \DeclareUnicodeAccent{\u}{\0306}
1271 \DeclareUnicodeAccent{\.}{\0307}
1272 \DeclareUnicodeAccent{\'}{\0308}
1273 \DeclareUnicodeAccent{\r}{\030A}
1274 \DeclareUnicodeAccent{\H}{\030B}
1275 \DeclareUnicodeAccent{\v}{\030C}
1276 \DeclareUnicodeAccent{\b}{\0323}
1277 \DeclareUnicodeAccent{\d}{\0323}
1278 \DeclareUnicodeAccent{\c}{\0327}
1279 \DeclareUnicodeAccent{\k}{\0328}

The odd one out:

1280 \DeclareUnicodeCommand\textcommabelow[1]
1281   {\hmode\bgroup\ooalign{\null#1\crr\hidewidth\raise-.31ex
1282     \hbox{\check@mathfonts\fontsize\ssf@size\z@
1283       \math@fontsfalse\selectfont,}\hidewidth}\egroup}

1284 \DeclareUnicodeComposite{\^}      {}{\005E}
1285 \DeclareUnicodeComposite{\~}      {}{\007E}

1286 \DeclareUnicodeComposite{\'}      {A}{\00C0}
1287 \DeclareUnicodeComposite{\'}      {A}{\00C1}
1288 \DeclareUnicodeComposite{\^}      {A}{\00C2}
1289 \DeclareUnicodeComposite{\~}      {A}{\00C3}
1290 \DeclareUnicodeComposite{"}       {A}{\00C4}
1291 \DeclareUnicodeComposite{\r}       {A}{\00C5}
1292 \DeclareUnicodeComposite{\c}       {C}{\00C7}
1293 \DeclareUnicodeComposite{\'}       {E}{\00C8}
1294 \DeclareUnicodeComposite{\'}       {E}{\00C9}
1295 \DeclareUnicodeComposite{\^}       {E}{\00CA}
1296 \DeclareUnicodeComposite{"}       {E}{\00CB}
1297 \DeclareUnicodeComposite{\'}       {I}{\00CC}
1298 \DeclareUnicodeComposite{\'}       {I}{\00CD}
1299 \DeclareUnicodeComposite{\^}       {I}{\00CE}
1300 \DeclareUnicodeComposite{"}       {I}{\00CF}
1301 \DeclareUnicodeComposite{\~}       {N}{\00D1}
1302 \DeclareUnicodeComposite{\'}       {O}{\00D2}
1303 \DeclareUnicodeComposite{\'}       {O}{\00D3}
1304 \DeclareUnicodeComposite{\^}       {O}{\00D4}
1305 \DeclareUnicodeComposite{\~}       {O}{\00D5}
1306 \DeclareUnicodeComposite{"}       {O}{\00D6}
1307 \DeclareUnicodeComposite{\'}       {U}{\00D9}
1308 \DeclareUnicodeComposite{\'}       {U}{\00DA}
1309 \DeclareUnicodeComposite{\^}       {U}{\00DB}
1310 \DeclareUnicodeComposite{"}       {U}{\00DC}
1311 \DeclareUnicodeComposite{\'}       {Y}{\00DD}
1312 \DeclareUnicodeComposite{\'}       {a}{\00E0}
1313 \DeclareUnicodeComposite{\'}       {a}{\00E1}
1314 \DeclareUnicodeComposite{\^}       {a}{\00E2}
1315 \DeclareUnicodeComposite{\~}       {a}{\00E3}
1316 \DeclareUnicodeComposite{"}       {a}{\00E4}
1317 \DeclareUnicodeComposite{\r}       {a}{\00E5}
1318 \DeclareUnicodeComposite{\c}       {c}{\00E7}
1319 \DeclareUnicodeComposite{\'}       {e}{\00E8}
1320 \DeclareUnicodeComposite{\'}       {e}{\00E9}

```

1321	\DeclareUnicodeComposite{\~}	{e}{"00EA}
1322	\DeclareUnicodeComposite{"}	{e}{"00EB}
1323	\DeclareUnicodeComposite{'}	{i}{"00EC}
1324	\DeclareUnicodeComposite{\'}	{i}{"00EC}
1325	\DeclareUnicodeComposite{'}	{i}{"00ED}
1326	\DeclareUnicodeComposite{'}	{i}{"00ED}
1327	\DeclareUnicodeComposite{^}	{i}{"00EE}
1328	\DeclareUnicodeComposite{^}	{i}{"00EE}
1329	\DeclareUnicodeComposite{"}	{i}{"00EF}
1330	\DeclareUnicodeComposite{"}	{i}{"00EF}
1331	\DeclareUnicodeComposite{~}	{n}{"00F1}
1332	\DeclareUnicodeComposite{'}	{o}{"00F2}
1333	\DeclareUnicodeComposite{'}	{o}{"00F3}
1334	\DeclareUnicodeComposite{^}	{o}{"00F4}
1335	\DeclareUnicodeComposite{~}	{o}{"00F5}
1336	\DeclareUnicodeComposite{"}	{o}{"00F6}
1337	\DeclareUnicodeComposite{'}	{u}{"00F9}
1338	\DeclareUnicodeComposite{'}	{u}{"00FA}
1339	\DeclareUnicodeComposite{^}	{u}{"00FB}
1340	\DeclareUnicodeComposite{"}	{u}{"00FC}
1341	\DeclareUnicodeComposite{'}	{y}{"00FD}
1342	\DeclareUnicodeComposite{"}	{y}{"00FF}
1343	\DeclareUnicodeComposite{=}	{A}{"0100}
1344	\DeclareUnicodeComposite{=}	{a}{"0101}
1345	\DeclareUnicodeComposite{u}	{A}{"0102}
1346	\DeclareUnicodeComposite{u}	{a}{"0103}
1347	\DeclareUnicodeComposite{k}	{A}{"0104}
1348	\DeclareUnicodeComposite{k}	{a}{"0105}
1349	\DeclareUnicodeComposite{'}	{C}{"0106}
1350	\DeclareUnicodeComposite{'}	{c}{"0107}
1351	\DeclareUnicodeComposite{^}	{C}{"0108}
1352	\DeclareUnicodeComposite{^}	{c}{"0109}
1353	\DeclareUnicodeComposite{.}	{C}{"010A}
1354	\DeclareUnicodeComposite{.}	{c}{"010B}
1355	\DeclareUnicodeComposite{v}	{C}{"010C}
1356	\DeclareUnicodeComposite{v}	{c}{"010D}
1357	\DeclareUnicodeComposite{v}	{D}{"010E}
1358	\DeclareUnicodeComposite{v}	{d}{"010F}
1359	\DeclareUnicodeComposite{=}	{E}{"0112}
1360	\DeclareUnicodeComposite{=}	{e}{"0113}
1361	\DeclareUnicodeComposite{u}	{E}{"0114}
1362	\DeclareUnicodeComposite{u}	{e}{"0115}
1363	\DeclareUnicodeComposite{.}	{E}{"0116}
1364	\DeclareUnicodeComposite{.}	{e}{"0117}
1365	\DeclareUnicodeComposite{k}	{E}{"0118}
1366	\DeclareUnicodeComposite{k}	{e}{"0119}
1367	\DeclareUnicodeComposite{v}	{E}{"011A}
1368	\DeclareUnicodeComposite{v}	{e}{"011B}
1369	\DeclareUnicodeComposite{^}	{G}{"011C}
1370	\DeclareUnicodeComposite{^}	{g}{"011D}
1371	\DeclareUnicodeComposite{u}	{G}{"011E}
1372	\DeclareUnicodeComposite{u}	{g}{"011F}
1373	\DeclareUnicodeComposite{.}	{G}{"0120}
1374	\DeclareUnicodeComposite{.}	{g}{"0121}

1375	\DeclareUnicodeComposite{c}	{G}{0122}
1376	\DeclareUnicodeComposite{c}	{g}{0123}
1377	\DeclareUnicodeComposite{^}	{H}{0124}
1378	\DeclareUnicodeComposite{^}	{h}{0125}
1379	\DeclareUnicodeComposite{~}	{I}{0128}
1380	\DeclareUnicodeComposite{~}	{i}{0129}
1381	\DeclareUnicodeComposite{~}	{i}{0129}
1382	\DeclareUnicodeComposite{=}	{I}{012A}
1383	\DeclareUnicodeComposite{=}	{i}{012B}
1384	\DeclareUnicodeComposite{=}	{i}{012B}
1385	\DeclareUnicodeComposite{u}	{I}{012C}
1386	\DeclareUnicodeComposite{u}	{i}{012D}
1387	\DeclareUnicodeComposite{u}	{i}{012D}
1388	\DeclareUnicodeComposite{k}	{I}{012E}
1389	\DeclareUnicodeComposite{k}	{i}{012F}
1390	\DeclareUnicodeComposite{k}	{i}{012F}
1391	\DeclareUnicodeComposite{.}	{I}{0130}
1392	\DeclareUnicodeComposite{^}	{J}{0134}
1393	\DeclareUnicodeComposite{^}	{j}{0135}
1394	\DeclareUnicodeComposite{^}	{j}{0135}
1395	\DeclareUnicodeComposite{c}	{K}{0136}
1396	\DeclareUnicodeComposite{c}	{k}{0137}
1397	\DeclareUnicodeComposite{'}	{L}{0139}
1398	\DeclareUnicodeComposite{'}	{l}{013A}
1399	\DeclareUnicodeComposite{c}	{L}{013B}
1400	\DeclareUnicodeComposite{c}	{l}{013C}
1401	\DeclareUnicodeComposite{v}	{L}{013D}
1402	\DeclareUnicodeComposite{v}	{l}{013E}
1403	\DeclareUnicodeComposite{'}	{N}{0143}
1404	\DeclareUnicodeComposite{'}	{n}{0144}
1405	\DeclareUnicodeComposite{c}	{N}{0145}
1406	\DeclareUnicodeComposite{c}	{n}{0146}
1407	\DeclareUnicodeComposite{v}	{N}{0147}
1408	\DeclareUnicodeComposite{v}	{n}{0148}
1409	\DeclareUnicodeComposite{=}	{O}{014C}
1410	\DeclareUnicodeComposite{=}	{o}{014D}
1411	\DeclareUnicodeComposite{u}	{O}{014E}
1412	\DeclareUnicodeComposite{u}	{o}{014F}
1413	\DeclareUnicodeComposite{H}	{O}{0150}
1414	\DeclareUnicodeComposite{H}	{o}{0151}
1415	\DeclareUnicodeComposite{'}	{R}{0154}
1416	\DeclareUnicodeComposite{'}	{r}{0155}
1417	\DeclareUnicodeComposite{c}	{R}{0156}
1418	\DeclareUnicodeComposite{c}	{r}{0157}
1419	\DeclareUnicodeComposite{v}	{R}{0158}
1420	\DeclareUnicodeComposite{v}	{r}{0159}
1421	\DeclareUnicodeComposite{'}	{S}{015A}
1422	\DeclareUnicodeComposite{'}	{s}{015B}
1423	\DeclareUnicodeComposite{^}	{S}{015C}
1424	\DeclareUnicodeComposite{^}	{s}{015D}
1425	\DeclareUnicodeComposite{c}	{S}{015E}
1426	\DeclareUnicodeComposite{c}	{s}{015F}
1427	\DeclareUnicodeComposite{v}	{S}{0160}
1428	\DeclareUnicodeComposite{v}	{s}{0161}

1429	\DeclareUnicodeComposite{c}	{T}{0162}
1430	\DeclareUnicodeComposite{c}	{t}{0163}
1431	\DeclareUnicodeComposite{v}	{T}{0164}
1432	\DeclareUnicodeComposite{v}	{t}{0165}
1433	\DeclareUnicodeComposite{~}	{U}{0168}
1434	\DeclareUnicodeComposite{~}	{u}{0169}
1435	\DeclareUnicodeComposite{=}	{U}{016A}
1436	\DeclareUnicodeComposite{=}	{u}{016B}
1437	\DeclareUnicodeComposite{u}	{U}{016C}
1438	\DeclareUnicodeComposite{u}	{u}{016D}
1439	\DeclareUnicodeComposite{r}	{U}{016E}
1440	\DeclareUnicodeComposite{r}	{u}{016F}
1441	\DeclareUnicodeComposite{H}	{U}{0170}
1442	\DeclareUnicodeComposite{H}	{u}{0171}
1443	\DeclareUnicodeComposite{k}	{U}{0172}
1444	\DeclareUnicodeComposite{k}	{u}{0173}
1445	\DeclareUnicodeComposite{^}	{W}{0174}
1446	\DeclareUnicodeComposite{^}	{w}{0175}
1447	\DeclareUnicodeComposite{^}	{Y}{0176}
1448	\DeclareUnicodeComposite{^}	{y}{0177}
1449	\DeclareUnicodeComposite{"}	{Y}{0178}
1450	\DeclareUnicodeComposite{'}	{Z}{0179}
1451	\DeclareUnicodeComposite{'}	{z}{017A}
1452	\DeclareUnicodeComposite{.}	{Z}{017B}
1453	\DeclareUnicodeComposite{.}	{z}{017C}
1454	\DeclareUnicodeComposite{v}	{Z}{017D}
1455	\DeclareUnicodeComposite{v}	{z}{017E}
1456	\DeclareUnicodeComposite{v}	{A}{01CD}
1457	\DeclareUnicodeComposite{v}	{a}{01CE}
1458	\DeclareUnicodeComposite{v}	{I}{01CF}
1459	\DeclareUnicodeComposite{v}	{i}{01D0}
1460	\DeclareUnicodeComposite{v}	{i}{01D0}
1461	\DeclareUnicodeComposite{v}	{O}{01D1}
1462	\DeclareUnicodeComposite{v}	{o}{01D2}
1463	\DeclareUnicodeComposite{v}	{U}{01D3}
1464	\DeclareUnicodeComposite{v}	{u}{01D4}
1465	\DeclareUnicodeComposite{'}	\AE{01FC}
1466	\DeclareUnicodeComposite{'}	{E}{01FC}
1467	\DeclareUnicodeComposite{'}	\ae{01FD}
1468	\DeclareUnicodeComposite{'}	{æ}{01FD}
1469	\DeclareUnicodeComposite{=}	\AE{01E2}
1470	\DeclareUnicodeComposite{=}	{E}{01E2}
1471	\DeclareUnicodeComposite{=}	\ae{01E3}
1472	\DeclareUnicodeComposite{=}	{æ}{01E3}
1473	\DeclareUnicodeComposite{v}	{G}{01E6}
1474	\DeclareUnicodeComposite{v}	{g}{01E7}
1475	\DeclareUnicodeComposite{v}	{K}{01E8}
1476	\DeclareUnicodeComposite{v}	{k}{01E9}
1477	\DeclareUnicodeComposite{k}	{O}{01EA}
1478	\DeclareUnicodeComposite{k}	{o}{01EB}
1479	\DeclareUnicodeComposite{v}	{j}{01F0}
1480	\DeclareUnicodeComposite{v}	{j}{01F0}
1481	\DeclareUnicodeComposite{'}	{G}{01F4}

```

1482 \DeclareUnicodeComposite{\'} {g}{"01F5}
1483 \DeclareUnicodeComposite{\textcommabelow}{S}{"0218}
1484 \DeclareUnicodeComposite{\textcommabelow}{s}{"0219}
1485 \DeclareUnicodeComposite{\textcommabelow}{T}{"021A}
1486 \DeclareUnicodeComposite{\textcommabelow}{t}{"021B}
1487 \DeclareUnicodeComposite{\=} {Y}{"0232}
1488 \DeclareUnicodeComposite{\=} {y}{"0233}
1489 \DeclareUnicodeComposite{\.} {B}{"1E02}
1490 \DeclareUnicodeComposite{\.} {b}{"1E03}
1491 \DeclareUnicodeComposite{\d} {B}{"1E04}
1492 \DeclareUnicodeComposite{\d} {b}{"1E05}
1493 \DeclareUnicodeComposite{\d} {D}{"1E0C}
1494 \DeclareUnicodeComposite{\d} {d}{"1E0D}
1495 \DeclareUnicodeComposite{\=} {G}{"1E20}
1496 \DeclareUnicodeComposite{\=} {g}{"1E21}
1497 \DeclareUnicodeComposite{\d} {H}{"1E24}
1498 \DeclareUnicodeComposite{\d} {h}{"1E25}
1499 \DeclareUnicodeComposite{\d} {K}{"1E32}
1500 \DeclareUnicodeComposite{\d} {k}{"1E33}
1501 \DeclareUnicodeComposite{\d} {L}{"1E36}
1502 \DeclareUnicodeComposite{\d} {l}{"1E37}
1503 \DeclareUnicodeComposite{\d} {M}{"1E42}
1504 \DeclareUnicodeComposite{\d} {m}{"1E43}
1505 \DeclareUnicodeComposite{\d} {N}{"1E46}
1506 \DeclareUnicodeComposite{\d} {n}{"1E47}
1507 \DeclareUnicodeComposite{\d} {R}{"1E5A}
1508 \DeclareUnicodeComposite{\d} {r}{"1E5B}
1509 \DeclareUnicodeComposite{\d} {S}{"1E62}
1510 \DeclareUnicodeComposite{\d} {s}{"1E63}
1511 \DeclareUnicodeComposite{\d} {T}{"1E6C}
1512 \DeclareUnicodeComposite{\d} {t}{"1E6D}
1513 \DeclareUnicodeComposite{\d} {V}{"1E7E}
1514 \DeclareUnicodeComposite{\d} {v}{"1E7F}
1515 \DeclareUnicodeComposite{\d} {W}{"1E88}
1516 \DeclareUnicodeComposite{\d} {w}{"1E89}
1517 \DeclareUnicodeComposite{\d} {Z}{"1E92}
1518 \DeclareUnicodeComposite{\d} {z}{"1E93}
1519 \DeclareUnicodeComposite{\d} {A}{"1EA0}
1520 \DeclareUnicodeComposite{\d} {a}{"1EA1}
1521 \DeclareUnicodeComposite{\d} {E}{"1EB8}
1522 \DeclareUnicodeComposite{\d} {e}{"1EB9}
1523 \DeclareUnicodeComposite{\d} {I}{"1ECA}
1524 \DeclareUnicodeComposite{\d} {i}{"1ECB}
1525 \DeclareUnicodeComposite{\d} {O}{"1ECC}
1526 \DeclareUnicodeComposite{\d} {o}{"1ECD}
1527 \DeclareUnicodeComposite{\d} {U}{"1EE4}
1528 \DeclareUnicodeComposite{\d} {u}{"1EE5}
1529 \DeclareUnicodeComposite{\d} {Y}{"1EF4}
1530 \DeclareUnicodeComposite{\d} {y}{"1EF5}
1531 </TU>

```

2 Package files

This file now also contains some packages that provide access to the more specialized encodings.

2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding FOO, the package looks to see if the encoding FOO has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be FOO.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't want to rush it.

1532 `<*package>`

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
1533 \def\update@uclc@with@cyrillic{%
1534 \expandafter\def\expandafter\@uclclist\expandafter
1535 {\@uclclist
1536 \cyr\CYRA\cyrahch\CYRABHCH\cyrahchdsc\CYRABHCHDSC\cyrahhdze
1537 \CYRABHDZE\cyrahha\CYRABHHA\cyrac\CYRAE\cyrb\CYRB\cyrbys
1538 \CYREBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
1539 \CYRCHRDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1540 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1541 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
1542 \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1543 \cyrgchcrs\CYRGHCRS\cyrgkh\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1544 \cyrhdsdsc\CYRHDS\cyrhchcrs\CYRHCHCRS\cyrhkh\CYRHHK\cyrhhrdsn
1545 \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
1546 \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1547 \cyrkbeak\CYRKBEAK\cyrkdsdsc\CYRKDSC\cyrkchcrs\CYRKCHCRS\cyrkhh
1548 \CYRKHK\cyrkvcrs\CYRKVCRS\cyr\CYRL\cyrldsc\CYRLDSC\cyrllhk
1549 \CYRLHK\cyrllje\CYRLJE\cyr\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
1550 \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
1551 \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1552 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsdsc\CYRRDSC\cyrrhk
1553 \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
1554 \cyrschwa\CYRSCHWA\cyrsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
1555 \cyrstftsn\CYRSFTSN\cyrrsh\CYRSH\cyrrshch\CYRSHCH\cyrrshha\CYRSHHA
1556 \cyrt\CYRT\cyrtddsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtsh\CYRTSHE
1557 \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyr\CYRY
1558 \CYRYA\CYRYA\cyrat\CYRYAT\cyrhcrs\CYRYHCRS\cyrri\CYRYI\cyrro
1559 \CYRYO\cyrri\CYRYU\cyrrz\CYRZ\cyrrzdsdsc\CYRRZDSC\cyrrzh\CYRZH
1560 \cyrrzhdsc\CYRRZHDSC}%
1561 \let\update@uclc@with@cyrillic\relax
1562 }
```

Here we process each option:

```
1563 \DeclareOption*{%
1564 \let\encodingdefault\CurrentOption
```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look at whether `\T@encoding` is already defined. If not, we load it later (indicated by setting the switch `@tempswa` to true) and we always load if we are using an older format (or rather in a rollback situation).

```

1565 \@tempswafalse
1566 \@ifl@t@r\fmtversion{2020/02/02}%
1567     {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1568       \@tempswatrue\fi}%
1569     {\@tempswatrue}%

Load if necessary:

1570 \if@tempswa

1571     \edef\reserved@f{%
1572       \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
1573     \reserved@f
1574     \InputIfFileExists\reserved@f
1575     {\PackageError{fontenc}%
1576       {Encoding file '\reserved@f' not found.%
1577       \MessageBreak
1578       You might have misspelled the name of the encoding
1579       \MessageBreak
1580       or a required support package (e.g., cyrillic) is
1581       \MessageBreak
1582       missing in your installation}%
1583     {Necessary code for this encoding was not
1584     loaded.\MessageBreak
1585     Thus calling the encoding later on will
1586     produce further error messages.}}%
1587     \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

1588 \expandafter\in@\expandafter{\CurrentOption}%
1589                               {T2A,T2B,T2C,X2,LCY,OT2}%
1590 \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

1591 \expandafter\in@\expandafter\cyr@\expandafter
1592                               {\@uclclist}%
1593 \ifin@
1594 \else
1595     \update@uclc@with@cyrillic
1596 \fi
1597 \fi
1598 \fi
1599 }

1600 \ProcessOptions*

```

We select the new font encoding default (i.e., the last encoding specified in the option list). But this encoding may not work with the current `\f@shape`: e.g., LY1 is not defined for `cmr` and therefore packages switching to LY1 usually also change `\rmdefault`. But

that only applies at `\begin{document}` so we get a spurious warning if we use what L^AT_EX previously used:

```
1601 %\fontencoding\encodingdefault\selectfont
```

So instead we do this here:

```
1602 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```
1603 \let\update@uclc@with@cyrillic\relax
```

Finally we pretend that the fontenc package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```
1604 \let\@elt\relax
```

```
1605 \xdef\@fontenc@load@list{\@fontenc@load@list
```

```
1606   \@elt{\csname opt@fontenc.sty\endcsname}}}
```

```
1607 \global\expandafter\let\csname ver@@fontenc.sty\expandafter\endcsname
```

```
1608   \csname ver@fontenc.sty\endcsname
```

```
1609 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
```

```
1610 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
```

```
1611 \global\let\@ifl@ter@@\@ifl@ter
```

```
1612 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
```

```
1613 \endpackage
```

File 22

ltcounts.dtx

1 Counters and Lengths

Commands for defining and using counters. This file defines the following commands. In each case $\langle counter \rangle$ may be $*$ denoting the current counter as set by a previous $\backslash refstepcounter$.

$\backslash newcounter$	To define a new counter.
$\backslash newcounteralias$	To define an alias name for a counter.
$\backslash setcounter$	To set the value of counters.
$\backslash addtocounter$	Increase the $\langle counter \rangle$ #1 by the number #2.
$\backslash stepcounter$	Increase the $\langle counter \rangle$ by one.
$\backslash refstepcounter$	Increase the $\langle counter \rangle$ by one, also setting the value used by $\backslash label$.
$\backslash value$	For accessing the value of the counter as a TeX number (as opposed to $\backslash the\langle counter \rangle$ which expands to the <i>printed</i> representation of $\langle counter \rangle$)
$\backslash arabic$	$\backslash arabic\langle counter \rangle$: 1, 2, 3, ...
$\backslash roman$	$\backslash roman\langle counter \rangle$: i, ii, iii, ...
$\backslash Roman$	$\backslash Roman\langle counter \rangle$: I, II, III, ...
$\backslash alph$	$\backslash alph\langle counter \rangle$: a, b, c, ...
$\backslash Alph$	$\backslash Alph\langle counter \rangle$: A, B, C, ...
$\backslash fnsymbol$	$\backslash fnsymbol\langle counter \rangle$: *, †, ‡, ...
$\backslash counterwithin$	$\backslash counterwithin[\langle format \rangle]\langle counter \rangle\langle within-counter \rangle$: Resets $\langle counter \rangle$ whenever $\langle within-counter \rangle$ is stepped. Also redefines $\backslash the\langle counter \rangle$ command to produce $\backslash the\langle within-counter \rangle.\langle format \rangle\langle counter \rangle$ with $\backslash arabic$ as the default for $\langle format \rangle$. Star form omits redefining the print representation. The $*$ alias for the current counter may not be used in either argument.
$\backslash counterwithout$	$\backslash counterwithout[\langle format \rangle]\langle counter \rangle\langle within-counter \rangle$: Removes $\langle counter \rangle$ from the reset list of $\langle within-counter \rangle$. Also redefines $\backslash the\langle counter \rangle$ command to produce $\langle format \rangle\langle counter \rangle$ with $\backslash arabic$ as the default for $\langle format \rangle$. Star form omits redefining the print representation. The $*$ alias for the current counter may not be used in either argument.

1 $\langle *2ekernel \rangle$

1.1 Document command and environment counter macros

An environment `foo` has an associated counter defined by the following control sequences:

<code>\c@foo</code>	Contains the counter's numerical value. It is defined by <code>\newcount\foocounter</code> .
<code>\thefoo</code>	Macro that expands to the printed value of <code>\foocounter</code> . For example, if sections are numbered within chapters, and section headings look like Section II-3. The Nature of Counters then <code>\thesection</code> might be defined by: <code>\def\thesection</code> <code>{\@Roman{\c@chapter}-\@arabic{\c@section}}</code>
<code>\theHfoo</code>	Macro that is used to create destination names. It should give a unique value across the document for every <code>\refstepcounter</code> .
<code>\p@foo</code>	Macro that expands to a printed 'reference prefix' of counter foo. Any <code>\ref</code> to a value created by counter foo will produce the expansion of <code>\p@foo\thefoo</code> when the <code>\label</code> command is executed. See file <code>ltxref.dtx</code> for an extension of this mechanism.
<code>\cl@foo</code>	List of counters to be reset when foo stepped. Has format <code>\@elt{countera}\@elt{counterb}\@elt{counterc}</code> .
<code>\alias@ctr@foo</code>	if foo is an alias counter created with <code>\newcounteralias</code> this command contains the name of the root counter.

For some environments the counter is named slightly differently, e.g., the counters associated with the `enumerate` environment have the names `enumi`, `enumii`, `enumiii`, and `enumiv` depending on the nesting level.

The same internal macros are defined for counters used by commands such as `\chapter`, `\section`, etc.

NOTE:

`\thefoo` and `\p@foo` *must* be defined in such a way that `\edef\bar{\thefoo}` or `\edef\bar{\p@foo}` defines `\bar` so that it will evaluate to the counter value at the time of the `\edef`, even after `\foocounter` and any other counters have been changed. This will happen if you use the standard commands `\@arabic`, `\@Roman`, etc.

The following commands are used to define and modify counters.

`\refstepcounter{<foo>}`

Same as `\stepcounter`, but it also defines `\@currentlabel`, `\@currentHref` and `\@currentcounter` and so that a subsequent `\label{<bar>}` command causes `\ref{<bar>}` to generate the current value of counter `<foo>`.

`\@definecounter{<foo>}`

Initializes counter `<foo>` (with empty reset list), defines `\p@foo` and `\thefoo` to be null and `\theHfoo` to be `\number\value{foo}`. Also adds `<foo>` to `\cl@ckpt` – the reset list of a dummy counter `@ckpt` used for taking checkpoints for the `\include` system.

`\@addtoreset{<foo>}{<bar>}` : Adds counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\@removefromreset{<foo>}{<bar>}` : Removes counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\setcounter` `\setcounter{<foo>}{<val>}` : Globally sets `\foocounter` equal to `<val>`.

```

2 \def\setcounter#1#2{%
3   \ifundefined{c@#1}%
4     {\@nocounterr{#1}}%
5     {\global\csname c@#1\endcsname#2\relax}}
```


(End of definition for `\setcounter`.)

`\addtocounter` `\addtocounter{<foo>}{<val>}` Globally increments `\foocounter` by `<val>`.

```

6 \def\addtocounter#1#2{%
7   \ifundefined{c@#1}%
8     {\@nocounterr{#1}}%
9     {\global\advance\csname c@#1\endcsname #2\relax}}

```

(End of definition for `\addtocounter`.)

`\newcounter` `\newcounter{<newctr>}[<oldctr>]` Defines `<newctr>` to be a counter, which is reset when counter `<oldctr>` is stepped. If `<newctr>` already defined produces ‘`c@newctr already defined`’ error.

```

10 \def\newcounter#1{%
11   \expandafter\ifdefinable \csname c@#1\endcsname
12     {\@definecounter{#1}}%
13   \@ifnextchar[{\@newctr{#1}}{}]}

```

(End of definition for `\newcounter`.)

`\newcounteralias` `\newcounteralias{<aliasctr>}{<rootctr>}` Defines `<aliasctr>` as an alias for the counter `<rootctr>`. The root counter can itself be an alias counter (it is not recommended to chain counter names like this as keeping track of the representations can get tricky, but explicit tests to avoid that seems unnecessary). The various representation commands of the alias counter call the respective representations of the root. This means that if a representation of the root is redefined, e.g. to make `\theH<counter>` unique then the alias follows. It is possible to redefine an alias representation, but this then breaks the connection to the root representation.

Every alias counter FOO has an associated command `\alias@ctr@FOO` which points to the root. This allows to test if a counter is an alias counter and also allows to find the root.

We do not rollback the definition to avoid errors if packages use it that do not rollback.

```

14 \newcommand*{\newcounteralias}[2]{%
15   \ifundefined{c@#2}{%
16     \@nocounterr{#2}%
17   }{%
18     \expandafter\ifdefinable\csname c@#1\endcsname{%
19       \global\expandafter\let
20       \csname c@#1\expandafter\endcsname\csname c@#2\endcsname
21       \expandafter\edef\csname the#1\endcsname{\expandafter\noexpand\csname the#2\endcsname}
22       \expandafter\edef\csname theH#1\endcsname{\expandafter\noexpand\csname theH#2\endcsname}
23       \expandafter\edef\csname p@#1\endcsname{\expandafter\noexpand\csname p@#2\endcsname}%
24       \expandafter\edef\csname cl@#1\endcsname{\expandafter\noexpand\csname cl@#2\endcsname}
25       \expandafter\def\csname alias@ctr@#1\endcsname{#2}%
26     }%
27   }%
28 }

```

(End of definition for `\newcounteralias`.)

`\value` `\value{<ctr>}` produces the value of counter `<ctr>`, for use with a `\setcounter` or `\addtocounter` command.

```

29 \def\value#1{\csname c@#1\endcsname}

```

(End of definition for \value.)

\c@* Make the current counter available as a L^AT_EX counter with name *, so \alph* returns the current counter as a letter, \stepcounter{*} increments the current counter, etc.

```
30 \protected\expandafter
31   \def\csname c@*\endcsname{\value\@currentcounter}
```

(End of definition for \c@*.)

\@newctr

```
32 \def\@newctr#1[#2]{%
33   \ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

(End of definition for \@newctr.)

\stepcounter \stepcounterfoo Globally increments counter \c@F00 and resets all subsidiary counters.

```
34 \def\stepcounter#1{%
35   \addtocounter{#1}\@ne
36   \begingroup
37     \let\@elt\@stpelt
38     \csname cl@#1\endcsname
39   \endgroup}
```

(End of definition for \stepcounter.)

\@stpelt Rather than resetting the “within” counter to zero we set it to −1 and then run \stepcounter that moves it to 0 and also initiates resetting the next level down.

```
40 \</2ekernel>
41 \<latexrelease>\IncludeInRelease{2015/01/01}{\@stpelt}
42 \<latexrelease>                                {Reset nested counters}%
43 \<*2ekernel | latexrelease>
44 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
45 \<latexrelease>\EndIncludeInRelease
46 \</2ekernel | latexrelease>
47 \<latexrelease>\IncludeInRelease{0000/00/00}{\@stpelt}
48 \<latexrelease>                                {Reset nested counters}%%
49 \<latexrelease>\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
50 \<latexrelease>\EndIncludeInRelease
51 \<*2ekernel>
```

(End of definition for \@stpelt.)

\cl@ckpt

```
52 \def\cl@ckpt{\@elt{page}}
```

(End of definition for \cl@ckpt.)

\@definecounter Older definitions of \@definecounter called unconditionally \newcount. This had the effect that connection to an alias counter could break if a user did, e.g.,

```
\newcounter{theorem}
\newcounteralias{lemma}{theorem}
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}{Lemma}
```

The command was therefore changed and now tests if the count already exist. If the counter to define is an already existing alias counter then we do not reset the representations and do not add it to @ckpt.

```

53 </2ekernel>
54 <latexrelease>\IncludeInRelease{2026/06/01}{\@definecounter}
55 <latexrelease>                                     {add aliascounter}%
56 <*2ekernel | latexrelease>
57 \def\@definecounter#1{%
58   \ifundefined{c@#1}{\expandafter\newcount\csname c@#1\endcsname}{}%
59   \setcounter{#1}\z@
60   \ifundefined{alias@ctr@#1}
61   {\global\expandafter\let\csname cl@#1\endcsname\@empty
62    \addtoreset{#1}{@ckpt}%
63    \global\expandafter\let\csname p@#1\endcsname\@empty
64    \expandafter\xdef\csname theH#1\endcsname{%
65     \noexpand\the\noexpand\value{#1}}}%

```

If \the#1 is undefined or \relax we define it with the standard definition for counters, otherwise we warn. This will catch, for example, that somebody defines a counter named “index” conflicting with the theindex environment.

```

66   \expandafter
67   \ifx\csname the#1\endcsname\relax
68     \expandafter
69     \gdef\csname the#1\expandafter\endcsname\expandafter
70     {\expandafter\@arabic\csname c@#1\endcsname}%
71   \else
72     \latex@warning{Command ‘\string\the#1’ already
73                   defined -- not changed}%
74   \fi}
75 <latexrelease>\EndIncludeInRelease
76 </2ekernel | latexrelease>
77 <latexrelease>\IncludeInRelease{2024/11/01}{\@definecounter}
78 <latexrelease>                                     {provide theHfoo commands}%
79 <latexrelease>\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
80 <latexrelease>   \setcounter{#1}\z@
81 <latexrelease>   \global\expandafter\let\csname cl@#1\endcsname\@empty
82 <latexrelease>   \addtoreset{#1}{@ckpt}%
83 <latexrelease>   \global\expandafter\let\csname p@#1\endcsname\@empty
84 <latexrelease>   \expandafter\xdef\csname theH#1\endcsname{%
85 <latexrelease>     \noexpand\the\noexpand\value{#1}}%

```

If \the#1 is undefined or \relax we define it with the standard definition for counters, otherwise we warn. This will catch, for example, that somebody defines a counter named “index” conflicting with the theindex environment.

```

86 <latexrelease>   \expandafter
87 <latexrelease>   \ifx\csname the#1\endcsname\relax
88 <latexrelease>     \expandafter
89 <latexrelease>     \gdef\csname the#1\expandafter\endcsname\expandafter
90 <latexrelease>     {\expandafter\@arabic\csname c@#1\endcsname}%
91 <latexrelease>   \else
92 <latexrelease>     \latex@warning{Command ‘\string\the#1’ already
93 <latexrelease>                   defined -- not changed}%
94 <latexrelease>   \fi}
95 <latexrelease>\EndIncludeInRelease

```

```

96 <latexrelease>\IncludeInRelease{0000/00/00}{\@definecounter}
97 <latexrelease>                                {provide theHfoo commands}%%
98 <latexrelease>\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
99 <latexrelease>    \setcounter{#1}\z@
100 <latexrelease>    \global\expandafter\let\csname cl@#1\endcsname\@empty
101 <latexrelease>    \@addtoreset{#1}{@ckpt}%
102 <latexrelease>    \global\expandafter\let\csname p@#1\endcsname\@empty
103 <latexrelease>    \expandafter
104 <latexrelease>    \ifx\csname the#1\endcsname\relax
105 <latexrelease>    \expandafter
106 <latexrelease>    \gdef\csname the#1\expandafter\endcsname\expandafter
107 <latexrelease>        {\expandafter\@arabic\csname c@#1\endcsname}%
108 <latexrelease>    \else
109 <latexrelease>    \@latex@warning{Command ‘\string\the#1’ already
110 <latexrelease>        defined -- not changed}%
111 <latexrelease>    \fi}
112 <latexrelease>\EndIncludeInRelease
113 <*2ekernel>

```

(End of definition for \@definecounter.)

\@kernel@get@ctr@root A helper command to retrieve the root counter for messages.

```

114 \def\@kernel@get@ctr@root#1{%
115   \@ifundefined{alias@ctr@#1}{#1}
116   {\expandafter\@kernel@get@ctr@root\csname alias@ctr@#1\endcsname}}

```

(End of definition for \@kernel@get@ctr@root.)

\@addtoreset If a counter is reset when a parent counter changes it no longer has an unique value across the document. As `\theH<counter>` should be unique this representation is changed to include also the representation of the parent. This is not 100% guaranteed to work but has been used this way by hyperref for many years without causing problems. Changing `\cl@counter` and `\theHcounter` isn't trivial to do if the counter is an alias counter: e.g. only redefining `\theH<counter>` from the alias counter breaks the connection to the theH-representation of the root counter. Redefining both `\theH`-representations could overwrite changes made by the user. If there is a chain of alias counters it gets even more complicated to keep everything in sync. So unlike the package `aliascnt` we avoid these complications and do not allow alias counters as arguments to `\@addtoreset` and `\@removefromreset`.

```

117 </2ekernel>
118 <latexrelease>\IncludeInRelease{2026/06/01}{\@addtoreset}
119 <latexrelease>                                {support alias counter}%
120 <*2ekernel| latexrelease>
121 \def\@addtoreset#1#2{%
122   \@ifundefined{alias@ctr@#1}{%
123     \@ifundefined{alias@ctr@#2}
124     {\expandafter\@cons\csname cl@#2\endcsname {{#1}}}%
125     \expandafter\xdef\csname theH#1\endcsname{%
126       \expandafter\noexpand\csname theH#2\endcsname.%
127       \noexpand\the\noexpand\value{#1}}}%
128     {\@latex@warning{Alias counters can not be used in a counter reset.\MessageBreak
129       Use ‘\@kernel@get@ctr@root{#2}’ instead of ‘#2’..}}
130     {\@latex@warning{Alias counters can not be used in a counter reset.\MessageBreak
131       Use ‘\@kernel@get@ctr@root{#1}’ instead of ‘#1’..}}

```

```

132 <latexrelease> \EndIncludeInRelease
133 </2ekernel | latexrelease>
134 <latexrelease> \IncludeInRelease{2024/11/01}{\@addtoreset}
135 <latexrelease> {provide theHfoo commands}%
136 <latexrelease> \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}%
137 <latexrelease> \expandafter\xdef\csname theH#1\endcsname{%
138 <latexrelease> \expandafter\noexpand\csname theH#2\endcsname.%
139 <latexrelease> \noexpand\the\noexpand\value{#1}}%
140 <latexrelease>}
141 <latexrelease> \EndIncludeInRelease
142 <latexrelease> \IncludeInRelease{0000/00/00}{\@addtoreset}
143 <latexrelease> {provide theHfoo commands}%%
144 <latexrelease> \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}%
145 <latexrelease> \EndIncludeInRelease
146 <*2ekernel>

(End of definition for \@addtoreset.)

147 </2ekernel>

```

\@removefromreset

```

148 <latexrelease> \IncludeInRelease{2026/06/01}{\@addtoreset}
149 <latexrelease> {support alias counter}%
150 <*2ekernel | latexrelease>
151 \def\@removefromreset#1#2{%

Even through this is internal and the programmer should know what he/she is doing we
test here if counter #2 is defined. If not, the execution would run into a tight loop. As
in \@addtoreset do not allow alias counters as arguments.

152 \ifundefined{c@#2}\relax
153 {\ifundefined{alias@ctr@#1}{%
154 \ifundefined{alias@ctr@#2}{%
155 \begingroup
156 \expandafter\let\csname c@#1\endcsname\@removefromreset
157 \def\@elt##1{%
158 \expandafter\ifx\csname c@##1\endcsname\@removefromreset
159 \else
160 \noexpand\@elt{##1}%
161 \fi}%
162 \expandafter\xdef\csname cl@#2\endcsname
163 {\csname cl@#2\endcsname}%
164 \endgroup}%
165 {\@latex@warning{Alias counters can not be removed from a reset.\MessageBreak
166 Use '\@kernel@get@ctr@root{#2}' instead of '#2'.}}}
167 {\@latex@warning{Alias counters can not be removed from a reset.\MessageBreak
168 Use '\@kernel@get@ctr@root{#1}' instead of '#1'.}}}
169 <latexrelease> \EndIncludeInRelease
170 </2ekernel | latexrelease>

171 <latexrelease> \IncludeInRelease{2018-04-01}
172 <latexrelease> {\@removefromreset}{Add interfaces}%
173 <latexrelease> \def\@removefromreset#1#2{%

```

Even through this is internal and the programmer should know what he/she is doing we test here if counter #2 is defined. If not, the execution would run into a tight loop.

```

174 <latexrelease> \ifundefined{c@#2}\relax

```

```

175 <latexrelease> {\begingroup
176 <latexrelease> \expandafter\let\csname c@#1\endcsname \@removefromreset
177 <latexrelease> \def\@elt##1{%
178 <latexrelease> \expandafter\ifx\csname c@##1\endcsname \@removefromreset
179 <latexrelease> \else
180 <latexrelease> \noexpand\@elt{##1}%
181 <latexrelease> \fi}%
182 <latexrelease> \expandafter\xdef\csname cl@#2\endcsname
183 <latexrelease> {\csname cl@#2\endcsname}%
184 <latexrelease> \endgroup}}

```

(End of definition for \@removefromreset.)

\ifbothcounters Test if arg #1 and #2 are counters and if so execute #3.

```

185 <*2ekernel | latexrelease>
186 \begingroup\catcode'\*=11 \lowercase{\endgroup
187 \def\@ifbothcounters#1#2#3{%
188 \begingroup\let\c@*\@undefined
189 \@ifundefined{c@#1}{\@nocounterr{#1}}%
190 {% else counter is defined
191 \@ifundefined{c@#2}{\@nocounterr{#2}}%
192 {% else both counter and within are defined
193 #3}%
194 \endgroup}}

```

(End of definition for \@ifbothcounters.)

```

195 </2ekernel | latexrelease>
196 <latexrelease>\EndIncludeInRelease
197 <latexrelease>\IncludeInRelease{0000-00-00}
198 <latexrelease> {\@removefromreset}{Add interfaces}%
199 <latexrelease>\let \@removefromreset \undefined
200 <latexrelease>\let \@ifbothcounters \undefined
201 <latexrelease>\EndIncludeInRelease
202 <*2ekernel>

```

\counterwithin New implementation using xparse and supporting an optional format argument.

```

203 </2ekernel>
204 <*2ekernel | latexrelease>
205 <latexrelease>\IncludeInRelease{2025/06/01}%
206 <latexrelease> {\counterwithin}{counter within}%
207 \NewDocumentCommand \counterwithin {s0{\arabic}mm}{%
208 \@ifbothcounters{#3}{#4}{%
209 \addtoreset{#3}{#4}%
210 \IfBooleanF #1%
211 {\expandafter
212 \xdef\csname the#3\expandafter\endcsname
213 {\expandafter\noexpand\csname the#4\endcsname
214 .\unexpanded{#2}{#3}}}%
215 }%
216 }

```

```

217 </2ekernel | latexrelease>
218 <latexrelease>\EndIncludeInRelease
219 <latexrelease>\IncludeInRelease{2021/11/15}%
220 <latexrelease>          {\counterwithin}{counter within}%
221 <latexrelease>
222 <latexrelease>\NewDocumentCommand \counterwithin {s0{\arabic}mm}{%
223 <latexrelease>  \@ifbothcounters{#3}{#4}{%
224 <latexrelease>    \@addtoreset{#3}{#4}%
225 <latexrelease>    \IfBooleanF #1%
226 <latexrelease>      {\expandafter
227 <latexrelease>        \gdef\csname the#3\expandafter\endcsname
228 <latexrelease>        \expandafter
229 <latexrelease>          {\csname the#4\endcsname .#2{#3}}}%
230 <latexrelease>  }%
231 <latexrelease>}
232 <latexrelease>\EndIncludeInRelease

233 <latexrelease>\IncludeInRelease{2018-04-01}
234 <latexrelease>          {\counterwithin}{counter within}%
235 <latexrelease>
236 <latexrelease>\def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
237 <latexrelease>\def\counterwithin@s#1#2{%
238 <latexrelease>  \ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}

239 <latexrelease>\def\counterwithin@x#1#2{%
240 <latexrelease>  \ifbothcounters{#1}{#2}%
241 <latexrelease>    {\@addtoreset{#1}{#2}%
242 <latexrelease>    \expandafter
243 <latexrelease>    \gdef\csname the#1\expandafter\endcsname\expandafter
244 <latexrelease>      {\csname the#2\expandafter\endcsname\expandafter
245 <latexrelease>        .\expandafter
246 <latexrelease>        \@arabic\csname c@#1\endcsname}}}%
247 <latexrelease>
248 <latexrelease>\EndIncludeInRelease
249 <latexrelease>\IncludeInRelease{0000-00-00}
250 <latexrelease>          {\counterwithin}{counter within}%
251 <latexrelease>\let \counterwithin \undefined
252 <latexrelease>\let \counterwithin@s \undefined
253 <latexrelease>\let \counterwithin@x \undefined
254 <latexrelease>\EndIncludeInRelease
255 <*2ekernel>

```

(End of definition for \counterwithin.)

\counterwithout New implementation using xparse and supporting an optional format argument.

```

256 </2ekernel>
257 <*2ekernel | latexrelease>
258 <latexrelease>\IncludeInRelease{2025/06/01}%
259 <latexrelease>          {\counterwithout}{counter without}%
260 <latexrelease>\NewDocumentCommand \counterwithout {s0{\arabic}mm}{%
261 <latexrelease>  \ifbothcounters{#3}{#4}{%
262 <latexrelease>    \@removefromreset{#3}{#4}%
263 <latexrelease>    \IfBooleanF #1%
264 <latexrelease>      {\expandafter
265 <latexrelease>        \xdef\csname the#3\endcsname {\unexpanded{#2}{#3}}}%

```

```

266 }%
267 }

268 </2ekernel | latexrelease>
269 <latexrelease>\EndIncludeInRelease

270 <latexrelease>\IncludeInRelease{2021/11/15}%
271 <latexrelease>          {\counterwithout}{counter without}%
272 <latexrelease>\NewDocumentCommand \counterwithout {s0{\arabic}mm}{-%
273 <latexrelease> \ifbothcounters{#3}{#4}{-%
274 <latexrelease> \removefromreset{#3}{#4}%
275 <latexrelease> \IfBooleanF #1%
276 <latexrelease> \expandafter
277 <latexrelease> \gdef\csname the#3\endcsname {#2{#3}}}%
278 <latexrelease> }%
279 <latexrelease>}
280 <latexrelease>\EndIncludeInRelease

281 <latexrelease>\IncludeInRelease{2018-04-01}
282 <latexrelease>          {\counterwithout}{counter without}%
283 <latexrelease>
284 <latexrelease>\def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
285 <latexrelease>\def\counterwithout@s#1#2{%
286 <latexrelease> \ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}%
287 <latexrelease>\def\counterwithout@x#1#2{%
288 <latexrelease> \ifbothcounters{#1}{#2}%
289 <latexrelease> {\@removefromreset{#1}{#2}%
290 <latexrelease> \expandafter
291 <latexrelease> \gdef\csname the#1\expandafter\endcsname\expandafter
292 <latexrelease> {\expandafter
293 <latexrelease> \@arabic\csname c@#1\endcsname}}%
294 <latexrelease>\EndIncludeInRelease
295 <latexrelease>
296 <latexrelease>\IncludeInRelease{0000-00-00}
297 <latexrelease>          {\counterwithout}{counter without}%
298 <latexrelease>\let \counterwithout \undefined
299 <latexrelease>\let \counterwithout@s \undefined
300 <latexrelease>\let \counterwithout@x \undefined
301 <latexrelease>\EndIncludeInRelease
302 <*2ekernel>

```

(End of definition for \counterwithout.)

Numbering commands for definitions of \theCOUNTER and \list arguments.

All commands can now be used in text and math mode.

\arabic Representation of *<counter>* as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing it COUNTER not positive.

```
303 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
```

(End of definition for \arabic.)

\roman Representation of *<counter>* as lower-case Roman numerals.

```
304 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

(End of definition for \roman.)

`\Roman` Representation of $\langle counter \rangle$ as upper-case Roman numerals.
305 `\def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}`
(End of definition for \Roman.)

`\alph` Representation of $\langle counter \rangle$ as a lower-case letter: 1 = a, 2 = b, etc.
306 `\def\alph#1{\expandafter\@alph\csname c@#1\endcsname}`
(End of definition for \alph.)

`\Alph` Representation of $\langle counter \rangle$ as an upper-case letter: 1 = A, 2 = B, etc.
307 `\def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}`
(End of definition for \Alph.)

`\fnsymbol` Representation of $\langle COUNTER \rangle$ as a footnote symbol: 1 = *, 2 = †, etc.
308 `\def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}`
(End of definition for \fnsymbol.)

`\@arabic` `\@arabic\F00counter` Representation of `\F00counter` as arabic numerals.
309 `\def\@arabic#1{\number #1} %% changed 29 Apr 86`
(End of definition for \@arabic.)

`\@roman` `\@roman\F00counter` Representation of `\F00counter` as lower-case Roman numerals.
310 `\def\@roman#1{\romannumeral #1}`
(End of definition for \@roman.)

`\@Roman` `\@Roman\F00counter` Representation of `\F00counter` as upper-case Roman numerals.
311 `\def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}`
(End of definition for \@Roman.)

`\@slowromancap` Fully expandable macro to change a roman number to uppercase.
312 `\def\@slowromancap#1{\ifx @#1% then terminate`
313 `\else`
314 `\if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if`
315 `c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi`
316 `\expandafter\@slowromancap`
317 `\fi`
318 `}`
(End of definition for \@slowromancap.)

`\@alph` `\@alph\F00counter` Representation of `\F00counter` as a lower-case letter: 1 = a, 2 = b, etc.
319 `\def\@alph#1{%`
320 `\ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or`
321 `k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or`
322 `y\or z\else\@ctrerr\fi}`
(End of definition for \@alph.)

`\@Alph` `\@Alph\F00counter` Representation of `\F00counter` as an upper-case letter: 1 = A, 2 = B, etc.

```

323 \def\@Alph#1{%
324   \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
325     K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or
326     Y\or Z\else\@ctrerr\fi}

```

(End of definition for `\@Alph`.)

`\@fnsymbol` Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

This macro is another example of an ever recurring problem in $\text{T}_{\text{E}}\text{X}$: Determining if something is text-mode or math-mode. It is imperative for the decision between text and math to be delayed until the actual typesetting is done as the code in question may go through an `\edef` or `\write` where an `\ifmmode` test would be executed prematurely. Hence in the implementation below, `\@fnsymbol` is not robust in itself but the parts doing the actual typesetting are.

In the case of `\@fnsymbol` we make use of the robust command `\TextOrMath` which takes two arguments and typesets the first if in text-mode and the second if in math-mode. Note that in order for this command to make the correct decision, it must insert a `\relax` token if run under regular $\text{T}_{\text{E}}\text{X}$, which ruins any kerning between the preceding characters and whatever awaits typesetting. If you use $\text{eT}_{\text{E}}\text{X}$ as engine for $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ (as recommended) this unfortunate side effect is not present.

```

327 </2ekernel>
328 <latexrelease>\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
329 <*2ekernel|latexrelease>
330 \def\@fnsymbol#1{%
331   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
332     \TextOrMath \textdagger \dagger\or
333     \TextOrMath \textdaggerdbl \ddagger \or
334     \TextOrMath \textsection \mathsection\or
335     \TextOrMath \textparagraph \mathparagraph\or
336     \TextOrMath \textbardbl ||\or
337     \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
338     \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
339     \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
340     \@ctrerr \fi
341 }%
342 </2ekernel|latexrelease>
343 <latexrelease>\EndIncludeInRelease
344 <latexrelease>\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TextOrMath}%
345 <latexrelease>\def\@fnsymbol#1{\ensuremath{%
346 <latexrelease>   \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
347 <latexrelease>     \mathparagraph\or ||\or **\or \dagger\dagger
348 <latexrelease>     \or \ddagger\ddagger \else\@ctrerr\fi}}%
349 <latexrelease>\EndIncludeInRelease
350 <*2ekernel>

```

(End of definition for `\@fnsymbol`.)

`\TextOrMath` When using regular $\text{T}_{\text{E}}\text{X}$, we make this command robust so that it always selects the correct branch in an `\ifmmode` switch with the usual disadvantage of ruining kerning. For the application we use it for here that shouldn't matter. The alternative would be to

mimic `\IeC` from `inputenc` but then it will have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell. However, users of `eTeX` will be pleasantly surprised to get the best of both worlds and no bad side effects.

First some code for checking if we are running `eTeX` but making sure not to permanently turn `\protected` into `\relax`.

```

351 </2ekernel>
352 <latexrelease>\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
353 <*2ekernel | latexrelease>
354 \begingroup\expandafter\expandafter\expandafter\endgroup
355 \expandafter\ifx\csname protected\endcsname\relax

```

In case of ordinary `TeX` we define `\TextOrMath` as a robust command but make sure it always grabs its arguments. If we didn't do this it might very well gobble spaces in the input stream.

```

356 \DeclareRobustCommand\TextOrMath{%
357   \ifmmode \expandafter\@secondoftwo
358   \else \expandafter\@firstoftwo \fi}
359 \protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
360 \else

```

For `eTeX` the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```

361 \protected\expandafter\def\csname TextOrMath\space\endcsname{%
362   \ifmmode \expandafter\@secondoftwo
363   \else \expandafter\@firstoftwo \fi}
364 \edef\TextOrMath#1#2{%
365   \expandafter\noexpand\csname TextOrMath\space\endcsname
366   {#1}{#2}}
367 \fi
368 </2ekernel | latexrelease>
369 <latexrelease>\EndIncludeInRelease
370 <latexrelease>\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
371 <latexrelease>\let\TextOrMath\@undefined
372 <latexrelease>\EndIncludeInRelease
373 <*2ekernel>

```

(End of definition for `\TextOrMath`.)

```

374 </2ekernel>

```

File 23

ltnlength.dtx

1 Lengths

```
\newlength Declare #1 to be a new length command.
\setlength Set the length command, #1, to the value #2.
\addtolength Increase the value of the length command, #1, by the value #2.
\settowidth Set the length, #1 to the width of a box containing #2.
\settoheight Set the length, #1 to the height of a box containing #2.
\settodepth Set the length, #1 to the depth of a box containing #2.

1 <*2ekernel>
2 \message{lengths,}

\newlength
3 \def\newlength#1{\@ifdefinable#1{\newskip#1}}
(End of definition for \newlength.)

\setlength
4 </2ekernel>
5 <latexrelease>\IncludeInRelease{2015/01/01}%
6 <latexrelease> \setlength}{Using \setlength with \dimen0}%
7 <*2ekernel | latexrelease>
8 \def\setlength#1#2{#1 #2\relax}
9 </2ekernel | latexrelease>
10 <latexrelease>\EndIncludeInRelease
11 <latexrelease>\IncludeInRelease{0000/00/00}%
12 <latexrelease> \setlength}{Using \setlength with \dimen0}%
13 <latexrelease>\def\setlength#1#2{#1#2\relax}
14 <latexrelease>\EndIncludeInRelease
15 <*2ekernel>
(End of definition for \setlength.)

\addtolength \relax added 24 Mar 86
16 \def\addtolength#1#2{\advance#1 #2\relax}
(End of definition for \addtolength.)

\settoheight The obvious analogs of \settowidth.
\settodepth
\settowidth
17 </2ekernel>
18 <*2ekernel | latexrelease>
19 <latexrelease>\IncludeInRelease{2024/11/01}%
20 <latexrelease> {\@settodim}{suspend tagging}%
21 \def\@settodim#1#2#3{\setbox\@tempboxa\hbox
22 {\SuspendTagging{\@settodim}#3\ResumeTagging{\@settodim}}#2#1\@tempboxa
Clear the memory afterwards (which might be a lot).
23 \setbox\@tempboxa\box\voidb@x}
24 </2ekernel | latexrelease>
25 <latexrelease>\EndIncludeInRelease
```

```

26 <latexrelease>\IncludeInRelease{0000/00/00}%
27 <latexrelease>          {\@settodim}{suspend tagging}%
28 <latexrelease>\def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
29 <latexrelease>          \setbox\@tempboxa\box\voidb@x}
30 <latexrelease>\EndIncludeInRelease
31 <*2ekernel>

32 \DeclareRobustCommand\settoheight{\@settodim\ht}
33 \DeclareRobustCommand\settodepth {\@settodim\dp}
34 \DeclareRobustCommand\settowidth {\@settodim\wd}

```

(End of definition for \settoheight and others.)

\@settopoint This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like 345.4666666pt when calculating a dimension.

```

35 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
36 </2ekernel>

```

(End of definition for \@settopoint.)

File 24

ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX ‘New’ Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Preliminary macros

We define a number of macros that will be used later.

```
1 <*2kernel>
2 \message{NFSS base,}
```

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
3 \def\@nomath#1{\relax\ifmmode
4   \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

(End of definition for `\@nomath`.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T_EX from scanning too far in certain situations.

```
5 \gdef\no@alphabet@error#1{\relax \ifmmode
6   \@latex@error{Math\space alphabet\space identifier\space
7     \noexpand#1is\space undefined\space in\space math\space
8     version\space ‘\math@version’}%
9   {Your\space requested\space math\space alphabet\space
10    is\space undefined\space in\space the\space current\space
11    math\space version.^^JCheck\space the\space spelling\space
12    or\space use\space the\space \noexpand\SetMathAlphabet\space
13    command.}
14   \fi}
```

(End of definition for `\no@alphabet@error`.)

`\new@mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).³²

`\mathgroup`

```
15 %\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
16 \let\mathgroup\fam
17 %\let\newfam\new@mathgroup
18 \@onlypreamble\new@mathgroup
```

(End of definition for `\new@mathgroup` and `\mathgroup`.)

³²For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

2 Macros for setting up the tables

`\DeclareFontShape` The macro `\DeclareFontShape` takes 6 arguments:

```
19 \def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
20 \nfss@catcodes
```

We use `\expandafter \endgroup` to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
21 \expandafter\endgroup
```

```
22 \DeclareFontShape@}
```

(End of definition for `\DeclareFontShape`.)

`\DeclareFontShape@`

```
23 \</2ekernel>
```

```
24 \<*2ekernel | latexrelease>
```

```
25 \<latexrelease>\IncludeInRelease{2020/02/02}%
```

```
26 \<latexrelease>{\DeclareFontShape@}{Maybe drop one m}%
```

```
27 \def\DeclareFontShape@#1#2#3#4#5#6{%
```

```
28 \expandafter\ifx\csname #1+#2\endcsname\relax
```

```
29 \<@latex@error{Font family ‘#1+#2’ unknown}\@eha
```

```
30 \else
```

If the series value is incorrectly specified with an extra “m”, e.g., “mc” instead of just “c”, drop the surplus “m” but keep the “m” if it is by its own. In that case also issue a warning that the declaration needs correction.

For this we compare the given value #3 with one where we may have dropped an “m”. If nothing has changes, fine. Otherwise there was a wrong value which is now corrected in `\reserved@b` so we use that and also issue a warning.

```
31 \edef\reserved@a{#3}%
```

```
32 \series@maybe@drop@one@m\reserved@a\reserved@b
```

```
33 \ifx\reserved@a\reserved@b\else
```

```
34 \<@latex@note{Font shape #1/#2/#3/#4 has incorrect series
```

```
35 value ‘#3’.\MessageBreak It should not contain an ‘m’!
```

```
36 Please correct it.\MessageBreak Found}%
```

```
37 \fi
```

```
38 \expandafter
```

```
39 \xdef\csname#1/#2/\reserved@b/#4\endcsname
```

```
40 {\expandafter\noexpand\csname #5\endcsname}%
```

```
41 %
```

Most of the time #6 is empty so using `\let` to `\@empty` saves on space compared to using `\def`. That’s really one of the old space saving techniques and probably not necessary these days.

```
42 \def\reserved@a{#6}%
```

```
43 \global
```

```
44 \expandafter\let\csname#5\endcsname\expandafter\endcsname
```

```
45 \ifx\reserved@a\@empty
```

```
46 \@empty
```

```
47 \else
```

```
48 \reserved@a
```

```
49 \fi
```

```
50 \fi
```

```
51 }
```

```

52 </2ekernel | latexrelease>
53 <latexrelease>\EndIncludeInRelease
54 <latexrelease>\IncludeInRelease{0000/00/00}%
55 <latexrelease>          {\DeclareFontShape@}{Maybe drop one m}%
56 <latexrelease>
57 <latexrelease>\def\DeclareFontShape@#1#2#3#4#5#6{%
58 <latexrelease>  \expandafter\ifx\csname #1+#2\endcsname\relax
59 <latexrelease>    \@latex@error{Font family ‘#1+#2’ unknown}\@eha
60 <latexrelease>  \else
61 <latexrelease>    \expandafter
62 <latexrelease>      \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
63 <latexrelease>                                     \csname #5\endcsname}%
64 <latexrelease>    \def\reserved@a{#6}%
65 <latexrelease>    \global
66 <latexrelease>    \expandafter\let\csname#5\endcsname\expandafter\endcsname
67 <latexrelease>    \ifx\reserved@a\@empty
68 <latexrelease>      \@empty
69 <latexrelease>    \else
70 <latexrelease>      \reserved@a
71 <latexrelease>    \fi
72 <latexrelease>  \fi
73 <latexrelease> }
74 <latexrelease>\EndIncludeInRelease
75 <*2ekernel>

```

(End of definition for \DeclareFontShape@.)

\DeclareFixedFont Define a direct font switch that avoids all overhead.

```

76 \def\DeclareFixedFont#1#2#3#4#5#6{%
77   \begingroup
78     \math@fontsfalse
79     \every@math@size{}%
80     \fontsize{#6}\z@
81     \usefont{#2}{#3}{#4}{#5}%
82     \global\expandafter\let\expandafter#1\the\font
83   \endgroup
84 }

```

(End of definition for \DeclareFixedFont.)

\do@subst@correction

```

85 \def\do@subst@correction{%
86   \xdef\subst@correction{%
87     \font@name
88     \global\expandafter\font
89     \csname \curr@fontshape/\f@size\endcsname
90     \noexpand\fontname\font
91     \relax}%

```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```

92     \aftergroup\subst@correction
93 }

```

(End of definition for \do@subst@correction.)

`\DeclareFontFamily`

```
94 \def\DeclareFontFamily#1#2#3{%  
If we want fast checking for the encoding scheme we can just check for \T@.. being  
defined.
```

```
95 % \@tempswafalse  
96 % \def\reserved@b{#1}%  
97 % \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%  
98 %       \ifx\reserved@b\reserved@c \@tempswatrue\fi}%  
99 % \cdp@list  
100 % \if@tempswa  
101 % \ifundefined{T@#1}%  
102 %   {%  
103 %     \@latex@error{Encoding scheme ‘#1’ unknown}\@eha  
104 %   }%  
105 %   {%
```

Now we have to define the macro `\langle#1\rangle#2` to contain `#3`. But since most of the time `#3` will be empty we use `\let` in a tricky way rather than a simple `\def` since this will save internal memory. We store the argument `#3` in a temporary macro `\reserved@a`.

```
106 \def\reserved@a{#3}%
```

We compare `\reserved@a` with `\@empty`. If these two are the same we `\let` the ‘extra’ macro equal to `\@empty` which is not the same as doing a `\let` to `\reserved@a` — the latter would blow one extra memory location rather than reusing the one from `\@empty`.

```
107 \global  
108 \expandafter\let\csname #1+#2\expandafter\endcsname  
109 \ifx \reserved@a\@empty  
110 \@empty  
111 \else \reserved@a  
112 \fi  
113 }%  
114 }
```

(End of definition for \DeclareFontFamily.)

`\cdp@list` We initialize the code page list to be empty.

```
115 \let\cdp@list\@empty  
116 \@onlypreamble\cdp@list
```

(End of definition for \cdp@list.)

`\cdp@elt`

```
117 \let\cdp@elt\relax  
118 \@onlypreamble\cdp@elt
```

(End of definition for \cdp@elt.)

`\DeclareFontEncoding`

```
119 \def\DeclareFontEncoding{%
```

First we start with ignoring all blanks and newlines since every surplus space in the second or third argument will come out in a weird place in the document.

```

120 \begingroup
121 \nfss@catcodes
122 \expandafter\endgroup
123 \DeclareFontEncoding@
124 \@onlypreamble\DeclareFontEncoding

125 \def\DeclareFontEncoding@#1#2#3{%
126 \expandafter
127 \ifx\csname T@#1\endcsname\relax
128 \def\cdp@elt{\noexpand\cdp@elt}%
129 \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
130 \{\default@family\}\{\default@series\}%
131 \{\default@shape\}}%

```

To support encoding dependent commands (like accents) we initialize the command `\encoding`-cmd to be `\changed@cmd`. (See `ltoutenc.dtx` for details.)

```

132 \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
133 \else
134 \@font@info{Redefining font encoding #1}%
135 \fi

136 \global\@namedef{T@#1}{#2}%
137 \global\@namedef{M@#1}{\default@M#3}%

```

Keep a record of the last encoding being declared:

```

138 \xdef\LastDeclaredEncoding{#1}%
139 }
140 \@onlypreamble\DeclareFontEncoding@

```

(End of definition for \DeclareFontEncoding.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```

141 \def\LastDeclaredEncoding{}

```

(End of definition for \LastDeclaredEncoding.)

`\DeclareFontSubstitution`

```

142 \def\DeclareFontSubstitution#1#2#3#4{%
143 \expandafter
144 \ifx\csname T@#1\endcsname\relax
145 \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
146 \else
147 \begingroup

```

We loop through the `\cdp@list` and rebuild it anew in `\toks@` thereby replacing the defaults for the encoding in question with the new defaults. It is important to store the encoding to test against expanded in `\reserved@a` since it might just be `\LastDeclaredEncoding` that is passed as `#1`.

```

148 \edef\reserved@a{#1}%
149 \toks@{}%
150 \def\cdp@elt##1##2##3##4{%
151 \def\reserved@b{##1}%
152 \ifx\reserved@a\reserved@b

```

Here we use the new defaults but we use `##1` (i.e., the encoding name already stored previously) since we know that it is expanded.

```

153         \addto@hook\toks@{\cdp@elt{##1}{#2}{#3}{#4}}%
154     \else
If \reserved@a and \reserved@b differ then we simply copy from the old list to the new.
155         \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
156     \fi}%
157     \cdp@list
158     \xdef\cdp@list{\the\toks@}%
159 \endgroup
160 \global
161 \@namedef{D@#1}{%
162     \def\default@family{#2}%
163     \def\default@series{#3}%
164     \def\default@shape{#4}%
165     }%
166 \fi
167 }
168 \@onlypreamble\DeclareFontSubstitution

```

(End of definition for `\DeclareFontSubstitution`.)

`\DeclareFontEncodingDefaults`

```

169 \def\DeclareFontEncodingDefaults#1#2{%
170     \ifx\relax#1\else
171         \ifx\default@T\@empty\else
172             \@font@info{Overwriting encoding scheme text defaults}%
173         \fi
174         \gdef\default@T{#1}%
175     \fi
176     \ifx\relax#2\else
177         \ifx\default@M\@empty\else
178             \@font@info{Overwriting encoding scheme math defaults}%
179         \fi
180         \gdef\default@M{#2}%
181     \fi
182 }
183 \@onlypreamble\DeclareFontEncodingDefaults

```

(End of definition for `\DeclareFontEncodingDefaults`.)

`\default@T`
`\default@M`

```

184 \let\default@T\@empty
185 \let\default@M\@empty

```

(End of definition for `\default@T` and `\default@M`.)

`\DeclareEncodingSubset`

The declaration takes 3 mandatory arguments: an *encoding* for which a subsetting is wanted (currently always `TS1`, and most likely forever), the *font family* for which we declare the subset and finally the *subset* number, with a value between 0 (all of the encoding is supported) and 9 (many glyphs are missing).

For `TS1` the numbers have been chosen in a way that most fonts can be fairly correctly categorized, but the default settings are always conservative, that is they may claim that fewer glyphs are supported than there actually are.

As these days many font families are set up to end in `-LF` (lining figures), `-OsF` (oldstyle figures), etc. the declaration supports a shortcut: if the *font family* name ends in `-*` then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

is the same as writing

```
\DeclareEncodingSubset{TS1}{Alegreya-LF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TOsF}{2}
```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

The coding of the declaration has no error checking as it is mostly for internal use.

```
186 \def\DeclareEncodingSubset#1#2{%
187   \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
188 }
189 \def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%
if #3 is empty then there was no star, otherwise we define all four variants.
190   \expandafter\ifx\expandafter X\detokenize{#3}X%
191     \@DeclareEncodingSubset{#1}{#2}{#4}%
192   \else
193     \@DeclareEncodingSubset{#1}{#2LF}{#4}%
194     \@DeclareEncodingSubset{#1}{#2TLF}{#4}%
195     \@DeclareEncodingSubset{#1}{#2OsF}{#4}%
196     \@DeclareEncodingSubset{#1}{#2ToSF}{#4}%
197   \fi
198 }
```

The subset info is stored in a command with the name `\family:subset` so if that already exists we change otherwise declare a subset.

```
199 \def\@DeclareEncodingSubset#1#2#3{%
200   \@ifundefined{#1:#2}%
201     {\@font@info{Setting #2 sub-encoding to #1/#3}}%
202     {\@font@info{Changing #2 sub-encoding to #1/#3}}%
```

This declaration should be usable in `.fd` files and therefore has to make its definition globally, because such files can get loaded in random places.

```
203   \global\@namedef{#1:#2}{#3}}
```

(End of definition for `\DeclareEncodingSubset`.)

\CheckEncodingSubset The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: `#2` and `#5` of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions in `ltxtextcomp.dtx`.

```

204 </2ekernel>
205 < *2ekernel | latexrelease>
206 < latexrelease> \IncludeInRelease{2025/06/01}%
207 < latexrelease> { \CheckEncodingSubset}{preload .fd file}%
208 \def\CheckEncodingSubset#1#2#3#4#5{%

```

If the `.fd` is not yet loaded we make an attempt to load it now because it may contain a `\DeclareEncodingSubset` declaration for the font.

```

209   \expandafter\ifx\csname #2:\f@family\endcsname\relax
210     \LoadFontDefinitionFile{#2}\f@family
211   \fi
212   \ifnum #4>%
213     \expandafter\ifx\csname #2:\f@family\endcsname\relax
214       0\csname #2:?\endcsname
215     \else
216       \csname #2:\f@family\endcsname
217     \fi
218   \relax
219   \expandafter\@firstoftwo
220 \else
221   \expandafter\@secondoftwo
222 \fi
223   {#1{#2}}{#3}%
224   #5%
225 }
226 </2ekernel | latexrelease>
227 < latexrelease> \EndIncludeInRelease
228 < latexrelease> \IncludeInRelease{0000/00/00}%
229 < latexrelease> { \CheckEncodingSubset}{preload .fd file}%
230 < latexrelease>
231 < latexrelease> \def\CheckEncodingSubset#1#2#3#4#5{%
232 < latexrelease>   \ifnum #4>%
233 < latexrelease>     \expandafter\ifx\csname #2:\f@family\endcsname\relax
234 < latexrelease>       0\csname #2:?\endcsname
235 < latexrelease>     \else
236 < latexrelease>       \csname #2:\f@family\endcsname
237 < latexrelease>     \fi
238 < latexrelease>   \relax
239 < latexrelease>   \expandafter\@firstoftwo
240 < latexrelease> \else
241 < latexrelease>   \expandafter\@secondoftwo
242 < latexrelease> \fi
243 < latexrelease> {#1{#2}}{#3}%
244 < latexrelease> #5%
245 < latexrelease> }
246 < latexrelease> \EndIncludeInRelease
247 < *2ekernel>

```

(End of definition for \CheckEncodingSubset.)

`\DeclarePreloadSizes`

```

248 \def\DeclarePreloadSizes#1#2#3#4#5{%
249   \@ifundefined{T@#1}%
250   {\@latex@error{Encoding scheme ‘#1’ unknown}\@eha}%
251   {%

```

Don't know at the moment what this group here does!

```

252   \begingroup

```

We define a macro `\reserved@f`³³ that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```

253   \def\reserved@f##1,{%

```

The end of the list will be detected when there are no more elements, i.e. when `\reserved@f`'s argument is empty. The trick used here is explained in Appendix D of the *T_EXbook*: if the argument is empty the `\if` will select the first clause and `\let \reserved@f equal to \relax`. (We use the `>` character here since it cannot appear in font file names.)

```

254       \if>##1>%
255       \let\reserved@f\relax
256       \else

```

Otherwise, we define `\font@name` appropriately and call `\pickup@font` to do the work. Note that the requested `\curr@fontshape` combination must have been defined, or you will get an error. The definition of `\font@name` is carried out globally to be consistent with the rest of the code in this file.

```

257       \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
258       \pickup@font

```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```

259       \global\expandafter\let\font@name\relax
260   \fi

```

Finally we call `\reserved@f` again to process the next *size*. If `\reserved@f` was `\let` equal to `\relax` this will end the macro.

```

261       \reserved@f}%

```

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```

262   \reserved@f#5,,%
263   \endgroup
264 }%
265 }
266 \@onlypreamble\DeclarePreloadSizes

```

(End of definition for `\DeclarePreloadSizes`.)

`\ifmath@fonts` We need a switch to decide if we have to switch math fonts. For this purpose we provide `\ifmath@fonts` that can be set to true or false by the `\S@...` macros depending on if math fonts are provided for this size or not. The default is of course to switch all fonts.

```

267 \newif\ifmath@fonts \math@fontstrue

```

³³We cannot use `\@tempa` since it is needed in `\pickup@font`.

(End of definition for \ifmath@fonts.)

\DeclareMathSizes \DeclareMathSizes takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right \S@... macro.

```
268 \def\DeclareMathSizes{%
269   \ifstar{\@DeclareMathSizes\math@fontsfalse}%
270   {\@DeclareMathSizes{}}}%
271 \@onlypreamble\DeclareMathSizes
```

(End of definition for \DeclareMathSizes and \DeclareMathSizes*.)

\@DeclareMathSizes This modification by Michael J. Downes on comp.text.tex on 2002/10/17 allows the user to have settings such as

\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}.

```
272 \</2kernel>
273 \<latexrelease>\IncludeInRelease{2015/01/01}{\@DeclareMathSizes}%
274 \<latexrelease>          {Arbitrary units in \DeclareMathSizes}%
275 \<*2kernel | latexrelease>
276 \def\@DeclareMathSizes #1#2#3#4#5{%
277   \@defaultunits\dimen@ #2pt\relax\@nnil
278   \if $#3$%
279     \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
280   \else
281     \@defaultunits\dimen@ii #3pt\relax\@nnil
282     \@defaultunits\@tempdima #4pt\relax\@nnil
283     \@defaultunits\@tempdimb #5pt\relax\@nnil
284     \toks@{#1}%
285     \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
286       \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
287       \gdef\noexpand\sfontsize{\strip@pt\@tempdima}%
288       \gdef\noexpand\ssfontsize{\strip@pt\@tempdimb}%
289       \the\toks@
290     }%
291     \fi
292   }%
293 \</2kernel | latexrelease>
294 \<latexrelease>\EndIncludeInRelease
295 \<latexrelease>\IncludeInRelease{0000/00/00}{\@DeclareMathSizes}%
296 \<latexrelease>          {Arbitrary units in \DeclareMathSizes}%
297 \<latexrelease>\def\@DeclareMathSizes#1#2#3#4#5{%
298 \<latexrelease>   \@defaultunits\dimen@#2pt\relax\@nnil
299 \<latexrelease>   \if $#3$%
300 \<latexrelease>     \expandafter \let
301 \<latexrelease>       \csname S@\strip@pt\dimen@\endcsname
302 \<latexrelease>       \math@fontsfalse
303 \<latexrelease>   \else
304 \<latexrelease>     \expandafter \gdef
305 \<latexrelease>       \csname S@\strip@pt\dimen@\endcsname
306 \<latexrelease>         {\gdef\tf@size{#3}\gdef\sfontsize{#4}%
307 \<latexrelease>           \gdef\ssfontsize{#5}%
308 \<latexrelease>           #1%
309 \<latexrelease>           }%
310 \<latexrelease>   \fi}%
311 \<latexrelease>\EndIncludeInRelease
```

```

312 <*2ekernel>
313 \onlypreamble\@DeclareMathSizes
(End of definition for \@DeclareMathSizes.)

```

3 Selecting a new font

3.1 Macros for the user

`\fontencoding` As we said in the introduction a font is described by four parameters. We first define `\f@encoding` macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

```

314 \DeclareRobustCommand\fontencoding[1]{%
315   \expandafter\ifx\csname T@#1\endcsname\relax
316     \latex@error{Encoding scheme ‘#1’ unknown}\@eha
317   \else
318     \edef\f@encoding{#1}%
319     \ifx\cf@encoding\f@encoding

```

If the new encoding is the same as the old encoding we have nothing to do. However, in case we had a sequence of several encoding changes without a `\selectfont` in-between we can save processing by making sure that `\enc@update` is `\relax`.

```

320       \let\enc@update\relax
321     \else

```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```

322       \let\enc@update\@enc@update
323     \fi
324   \fi
325 }

```

(End of definition for `\fontencoding` and `\f@encoding`.)

`\@enc@update`

```

326 \def\@enc@update{%

```

When `\@enc@update` is executed `\f@encoding` holds the encoding name for the new encoding and `\cf@encoding` the name of the last active encoding.

We start by setting the init command for encoding dependent macros to `\@changed@cmd`.

```

327   \expandafter
328   \let
329   \csname\cf@encoding -cmd\endcsname
330   \@changed@cmd

```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```

331   \expandafter
332   \let
333   \csname\f@encoding-cmd\endcsname
334   \@current@cmd

```


We execute the default settings `\default@T`, followed by the one for the new encoding.

```
335 \default@T
336 \csname T@f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
337 \csname D@f@encoding\endcsname
338 \let\enc@update\relax
339 \let\cf@encoding\f@encoding
340 }
```

(End of definition for \enc@update.)

`\enc@update` The default action in `\selectfont` is to do nothing.

```
341 \let\enc@update\relax
```

(End of definition for \enc@update.)

`\fontfamily` They are now all defined later (and differently).

```
\f@family 342 %\DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
\fontseries 343 %\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
\f@series 344 %\DeclareRobustCommand\fontshape[1]{\edef\f@shape{#1}}
\fontshape
\f@shape
```

(End of definition for \fontfamily and others.)

`\usefont` Some handy abbreviation if you want to get some particular font in the current size. If also the size should change one has to issue a `\fontsize` command first.

`\fontencoding` needs to do some setup work so we call that, but instead of calling `\fontfamily`, `\fontseries` and `\fontshape` it earlier versions of this code did, we now set `\f@family`, etc. directly. If we would call `\fontseries` or `\fontshape` as it was done in the past, they would now interact with the existing series and shape which is not desired if we intend to use an explicit font shape!

```
345 </2ekernel>
346 <*2ekernel | latexrelease>
347 <latexrelease> \IncludeInRelease{2021/06/01}%
348 <latexrelease> \usefont{Force font face}%
349 \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
350 \edef\f@family{#2}%
351 \set@target@series{#3}%
352 \edef\f@shape{#4}%
```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
353 \let\delayed@f@adjustment\@empty
354 \selectfont
355 \ignorespaces}
356 </2ekernel | latexrelease>
357 <latexrelease> \EndIncludeInRelease
358 <latexrelease> \IncludeInRelease{2020/02/02}%
359 <latexrelease> \usefont{Drop m in usefont}%
360 <latexrelease>
361 <latexrelease> \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
362 <latexrelease> \edef\f@family{#2}%
```

```

363 <latexrelease> \set@target@series{#3}%
364 <latexrelease> \edef\f@shape{#4}\selectfont
365 <latexrelease> \ignorespaces}
366 <latexrelease>
367 <latexrelease>\EndIncludeInRelease
368 <latexrelease>\IncludeInRelease{0000/00/00}%
369 <latexrelease> \usefont{Drop m in usefont}%
370 <latexrelease>
371 <latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
372 <latexrelease> \edef\f@family{#2}%
373 <latexrelease> \edef\f@series{#3}%
374 <latexrelease> \edef\f@shape{#4}\selectfont
375 <latexrelease> \ignorespaces}
376 <latexrelease>
377 <latexrelease>\EndIncludeInRelease
378 <*2kernel>

```

(End of definition for \usefont.)

\linespread The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```

379 \DeclareRobustCommand\linespread[1]
380 { \set@fontsize{#1}\f@size\f@baselineskip}

```

(End of definition for \linespread.)

\fontsize We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. As the first argument to `\set@fontsize` we pass the current value of `\baselinestretch`. This will either match the internal value (in which case nothing changes) or it will be an updated value due to a user change of that macro using `\renewcommand`. If we would pass the internal `\f@linespread` such a change would be effectively overwritten by a size change.

```

381 \DeclareRobustCommand\fontsize[2]
382 { \set@fontsize\baselinestretch{#1}{#2}}

```

(End of definition for \fontsize.)

\f@linespread This macro holds the current internal value for `\baselinestretch`.

```

383 \let\f@family\@empty
384 \let\f@series\@empty
385 \let\f@shape\@empty
386 \let\f@size\@empty
387 \let\f@baselineskip\@empty
388 \let\f@linespread\@empty

```

(End of definition for \f@linespread.)

\cf@encoding

```

389 \let\f@encoding\@empty
390 \let\cf@encoding\@empty

```

(End of definition for \cf@encoding.)

`\@defaultunits` The function `\@defaultunits` when wrapped around a `dimen` or `skip` assignment supplies default units. Usage:

```
\@defaultunits\dimen@=#1pt\relax\@nnil
```

Note: the `\relax` is *important*. Other units can be substituted for the ‘pt’ if desired.

We use `\remove@to@nnil` as an auxiliary macros for `\@defaultunits`. It just has to gobble the supplied default unit ‘pt’ or whatever, if it wasn’t used in the assignment.

```
391 \def\@defaultunits{\afterassignment\remove@to@nnil}
```

(End of definition for `\@defaultunits`.)

`\strip@pt` This macro strips the characters `pt` produced by using `\the` on a `dimen` register.

```
\rem@pt
392 \begingroup
393   \catcode‘P=12
394   \catcode‘T=12
395   \lowercase{
396     \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}
397   \expandafter\endgroup\x
398   \def\strip@pt{\expandafter\rem@pt\the}
```

(End of definition for `\strip@pt` and `\rem@pt`.)

`\mathversion` `\mathversion` takes the *math version* name as argument, defines `\math@version` appropriately and switches to the font selected forcing a call to `\glb@settings` if the *version* is known to the system.

```
399 \DeclareRobustCommand\mathversion[1]
400   {\nomath\mathversion
401     \expandafter\ifx\csname mv@#1\endcsname\relax
402     \latex@error{Math version ‘#1’ is not defined}\@eha\else
```

If there has been a frozen math version reset unconditionally to it if we are at `\@math@level` one, so that in the typical case of one bold symbol within the normal version this doesn’t allocate an additional math alphabet. If the nesting is deeper we do nothing, which means alphabet allocations accumulate until the end of the formula. One could do slightly better but that would mean keeping track of the allocations on all levels severately and this is likely to be overkill in nearly all situations.

```
403     \ifcsname mv@#1\endcsname
404     \ifnum \@math@level = \@ne
405       \unconditionally@reset@math@version {#1}%
406     \fi
407   \fi
408   \edef\math@version{#1}%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting `\glb@currsz` to an invalid value since this will trigger the setup when the formula starts.

```
409   \gdef\glb@currsz{}
```

When the scope of the current `\mathversion` ends we need to restore the old setup. However this time we need to force it directly at least if we are inside math, otherwise we could wait. Another way to enhance this code here is to do the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```
410         \aftergroup\glb@settings
411         \fi}
```

Resetting the math version unconditionally means that we have to copy the frozen version to `\mv@#1` and also reset the counter `\c@mv@#1` to the number of math alphabets allocated in the frozen version.

```
412 \ExplSyntaxOn
413 \cs_new_protected:Npn \unconditionally@reset@math@version #1 {
414   \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
415   \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
416 }
417 \ExplSyntaxOff
```

(End of definition for \mathversion and \math@version.)

If $\TeX$ would support a hook just before the end of a formula (opposite of `\everymath` so to speak) the implementation of the algorithm would be much simpler because in that case we would set up the correct math fonts at this point without having to worry about incorrect settings due to nesting. The same would be true if in $\LaTeX$ the use of `$` (as the primitive $\TeX$ command) would be impossible and instead only a higher-level interface would be available. Note that this does not mean that a `$` couldn't be the short-hand for starting and stopping that higher-level interface, it only means that the direct $\TeX$ function must be hidden.

Anyway, since we don't have this and won't have it in $\LaTeX 2_\epsilon$ we need to implement it in a somewhat slower way.

We test for the current math font setup on entry of a formula, i.e., on the hooks `\everymath` and `\everydisplay`. But since these hooks may contain user data we provide ourselves with an internal version of these hooks which stays frozen.

```
\frozen@everymath New internal names for \everymath and \everydisplay.
\frozen@everydisplay 418 \let\frozen@everymath\everymath
                     419 \let\frozen@everydisplay\everydisplay
```

(End of definition for \frozen@everymath and \frozen@everydisplay.)

```
\everymath Now we provide now user hooks that will be called in the frozen internals.
\everydisplay 420 \newtoks\everymath
               421 \newtoks\everydisplay
```

(End of definition for \everymath and \everydisplay.)

```
\@math@level This counter records the nesting level of math within math.
              422 \newcount\@math@level
```

(End of definition for \@math@level.)

`\frozen@everydisplay` Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.

The check code may push tokens after the math formula with `\aftergroup` and they would prevent a `$$` from dropping following spaces. We therefore use a switch to be set as the first thing after the group so that following code can determine if there was a display or some inline math (in the latter case we better not drop spaces). After setting the switch we also have to place `\ignorespaces` because setting the switch may be the only thing that happens after the display. The issue with handling of spaces was found in 2022, but it is really a bug fix for the code added in 2021/11.

```

423 </2ekernel>
424 <latexrelease>\IncludeInRelease{2021/11/15}
425 <latexrelease> {\frozen@everydisplay}{Handle spaces after math}%
426 <*2ekernel| latexrelease>
427 \frozen@everydisplay = {%
428   \aftergroup\@ignoretrue \aftergroup\ignorespaces

```

Record that we entered another math level.

```

429   \advance\@math@level\@ne
430   \check@mathfonts
431   \the\everydisplay}

```

(End of definition for \frozen@everydisplay.)

`\frozen@everymath` The frozen code for inline math is similar, except that here we do not want to drop following spaces.

```

432 \frozen@everymath = {%
433   \aftergroup\@ignorefalse

```

Record that we entered another math level.

```

434   \advance\@math@level\@ne
435   \check@mathfonts
436   \the\everymath}

```

(End of definition for \frozen@everymath.)

```

437 </2ekernel| latexrelease>
438 <latexrelease>\EndIncludeInRelease
439 <latexrelease>\IncludeInRelease{2020/10/01}
440 <latexrelease> {\frozen@everydisplay}{Handle spaces after math}%
441 <latexrelease>
442 <latexrelease>\frozen@everydisplay = {\check@mathfonts
443                                     \the\everydisplay}
444 <latexrelease>\frozen@everymath = {\check@mathfonts
445                                     \the\everymath}
446 <latexrelease>
447 <latexrelease>\EndIncludeInRelease
448 <*2ekernel>

```

`\curr@math@size` This holds locally the current math size.

```

449 \let\curr@math@size\@empty

```

(End of definition for \curr@math@size.)

3.2 Macros for loading fonts

`\pickup@font` The macro `\pickup@font` which is used in `\selectfont` is very simple: if the font name is undefined (i.e. not known yet) it calls `\define@newfont` to load it.

```
450 \def\pickup@font{%
451     \expandafter \ifx \font@name \relax
452     \define@newfont
453     \fi}
```

(End of definition for `\pickup@font`.)

`\split@name` `\pickup@font` assumes that `\font@name` is set but it is sometimes called when `\f@family`, `\f@series`, `\f@shape`, or `\f@size` may have the wrong settings (see, e.g., the definition of `\getanddefine@fonts`). Therefore we need a macro to extract font *family*, *series*, *shape*, and *size* from the font name. To this end we define `\split@name` which takes the font name as a list of characters of `\catcode 12` (without the backslash at the beginning) delimited by the special control sequence `\@nil`. This is not very complicated: we first ensure that `/` has the right `\catcode`

```
454 {\catcode'\/=12
```

and define `\split@name` so that it will define our private `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, and `\f@size` macros.

```
455 \gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
456                                     \def\f@family{#2}%
457                                     \def\f@series{#3}%
458                                     \def\f@shape{#4}%
459                                     \def\f@size{#5}}}
```

(End of definition for `\split@name`.)

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
460 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

(End of definition for `\curr@fontshape`.)

`\define@newfont` Now we can tackle the problem of defining a new font.

```
461 \def\define@newfont{%
```

We have already mentioned that the token list that `\split@name` will get as argument must not start with a backslash. To reach this goal we will set the `\escapechar` to `-1` so that the `\string` primitive will not generate an escape character. To keep this change local we open a group. We use `\begingroup` for this purpose since `\define@newfont` might be called in math mode, and an empty `\bgroup... \egroup` would add an empty Ord atom to the math list and thus affect the spacing.

Also locally redefine `\typeout` so that ‘No file ...fd’ Warnings become Font Info message just sent to the log file.

```
462 \begingroup
463 \let\typeout\@font@info
464 \escapechar\m@ne
```

Then we extract *encoding scheme*, *family*, *series*, *shape*, and *size* from the font name. Note the four `\expandafter`’s so that `\font@name` is expanded first, then `\string`, and finally `\split@name`.

```
465 \expandafter\expandafter\expandafter
466 \split@name\expandafter\string\font@name\@nil
```

If the `\curr@fontshape` combination is not available, (i.e. undefined) we call the macro `\wrong@fontshape` to take care of this case. Otherwise `\extract@font` will load the external font for us.

```

467 %    \expandafter\ifx
468 %        \csname\curr@fontshape\endcsname \relax
469 %    \try@load@fontshape % try always
470 %    \fi
471 %    \expandafter\ifx
472 %        \csname\curr@fontshape\endcsname \relax
473 %    \wrong@fontshape\else

```

To allow substitution we call the `curr@fontshape` macro which usually will expand to `\relax` but may hold code for substitution (see `\subst@fontshape` definition).

```

474 %    \csname\curr@fontshape\endcsname
475 %    \extract@font\fi

```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```

476 \endgroup}

477 \def\try@load@fontshape{%
478     \expandafter
479     \ifx\csname \f@encoding+\f@family\endcsname\relax
480         \@font@info{Trying to load font information for
481             \f@encoding+\f@family}%

```

We predefine this combination to be `\@empty` which means that next time we don't try again unnecessary in case we don't find a `.fd` file. If the file contains a `\DeclareFontFamily` command than this setting will be overwritten.

```

482     \global\expandafter\let
483     \csname \f@encoding+\f@family\endcsname\@empty

```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```

484     \nfss@catcodes
485     \let\nfss@catcodes\relax

```

For increased portability make the external filename monospace, but look for the (old style) mixed case filename if the first attempt fails.

On any monospace system this means that the file is looked for twice which takes up time and string space, but at least for this release Check for both names to give people time to re-install their private fd files with lowercase names.

```

486     \edef\reserved@a{%
487         \lowercase{%
488             \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}}%
489     \reserved@a\relax
490     {\@input@\f@encoding\f@family.fd}%
491 \fi}

```

(End of definition for \define@newfont.)

\nfss@catcodes This macro should contain the standard `\catcode` assignments to all characters which are used in the commands found in an `.fd` file and which might have special `\catcodes` in the middle of a document. If necessary, this list can be extended in a package file using a suitable number of `\expandafter`, i.e.,

```
\expandafter\def\expandafter\nfss@catcodes
\expandafter{\nfss@catcodes <additional settings>}
```

Note, that this macro might get executed several times since it is also called by `\DeclareFontShape`, thus it probably should not be misused as a general purpose hook.

```
492 \def\nfss@catcodes{%
```

We start by making `@` a letter and ignoring all blanks and newlines.

```
493 \makeatletter
494 \catcode'\ 9%
495 \catcode'\^~I9%
496 \catcode'\^~M9%
```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```
497 \catcode'\\\z@
498 \catcode'\{\@ne
499 \catcode'\}\tw@
500 \catcode'\#6%
501 \catcode'\^7%
502 \catcode'\%14%
```

Then we make sure that the important syntax parts have the right `\catcode`.

```
503 \@makeother\<%
504 \@makeother\>%
505 \@makeother\*%
506 \@makeother\.%
507 \@makeother\-%
508 \@makeother\/%
509 \@makeother\[%
510 \@makeother\]%
511 \@makeother\'%
512 \@makeother\'%
513 \@makeother\"%
514 }
```

(End of definition for \nfss@catcodes.)

`\LoadFontDefinitionFile` Load and `.fd` files for some encoding and family (if it exists).

```
515 </2kernel>
516 <*2kernel | latexrelease>
517 <latexrelease>\IncludeInRelease{2020/02/02}%
518 <latexrelease>\LoadFontDefinitionFile{Loading .fd files}%
519 \def\LoadFontDefinitionFile#1#2{%
520 \begingroup
521 \edef\f@encoding{#1}%
522 \edef\f@family{#2}%
523 \try@load@fontshape
524 \endgroup
525 }
526 </2kernel | latexrelease>
527 <latexrelease>\EndIncludeInRelease
528 <latexrelease>\IncludeInRelease{0000/00/00}%
529 <latexrelease>\LoadFontDefinitionFile{Loading .fd files}%
530 <latexrelease>
```



```

531 \let\LoadFontDefinitionFile\undefined
532 \let\EndIncludeInRelease
533 \*2ekernel

```

(End of definition for \LoadFontDefinitionFile.)

`\DeclareFontFamilySubstitution` The idea for this macro is stolen from the `substitutefont` package by Günter Milde, with some modifications and a new name.

Its purpose is to provide characters in a special encoding that are not available in the current font family to be taken from a different family that is visually compatible (or not if you choose badly). For example, you can match the GFS Didot Greek characters with T_EX Gyre Pagella (Palatino) by specifying

```
\DeclareFontFamilySubstitution{LGR}{qpl}{udidot}
```

This way if you ask for the LGR encoding in for the `qpl` family you get the characters from the `udidot` family substituted.

We need to ensure that the macro is defined with `\nfss@catcodes` in force (not quite sure why at the moment to be honest).

```

534 \end{kernel}
535 \*2ekernel | latexrelease
536 \let\EndIncludeInRelease\IncludeInRelease{2020/02/02}%
537 \let\DeclareFontFamilySubstitution\Provide family substitution}%
538 \begin{group}
539 \nfss@catcodes
540 \gdef\DeclareFontFamilySubstitution#1#2#3{%

```

We only provide a set of silent substitutions. The package also (re)declared the family, but this is incorrect in my eyes and it is better to handle that differently.

Of course the families may still need loading at this point and so we arrange for this. Otherwise we might run into trouble because the necessary `\DeclareFontFamily` has not been seen.

```

541 \LoadFontDefinitionFile{#1}{#2}%
542 \LoadFontDefinitionFile{#1}{#3}%
543 \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
544 \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
545 \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
546 \DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}%

```

These days a few more shapes might be around, so we declare those too. If they don't exist then after the first substitution normal fallbacks will happen.

```

547 \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
548 \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
549 \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%

```

Same game with `b` and `bx`, for other weights you are on your own:

```

550 \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
551 \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
552 \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
553 \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
554 \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
555 \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
556 \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
557 \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%

```

```

558 \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
559 \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
560 \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
561 \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
562 \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
563 \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
564 }
565 \endgroup
566 </2ekernel | latexrelease>
567 <latexrelease>\EndIncludeInRelease
568 <latexrelease>\IncludeInRelease{0000/00/00}%
569 <latexrelease> \DeclareFontFamilySubstitution{Provide family substitution}%
570 <latexrelease>
571 <latexrelease>\let\DeclareFontFamilySubstitution\@undefined
572 <latexrelease>\EndIncludeInRelease
573 <*2ekernel>

```

(End of definition for \DeclareFontFamilySubstitution.)

\DeclareErrorFont Declare the last resort shape! We assume that in this fontshape there is a 10pt font but it doesn't really matter. We only loose one macro name if the assumption is false. But at least the font should be there!

```

574 </2ekernel>
575 <*2ekernel | latexrelease>
576 <latexrelease>\IncludeInRelease{2019/10/01}%
577 <latexrelease> \DeclareErrorFont{No side effects please}%
578 \def\DeclareErrorFont#1#2#3#4#5{%
579 \xdef\error@fontshape{%
580 \noexpand\expandafter\noexpand\split@name\noexpand\string
581 \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
582 \noexpand\@nil}%

```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstrapping process. Later on such values will be updated when an encoding is selected, etc.

We definitely don't want to set \f@encoding; we can set all the others since if they are left "blank" any selection would grab "error default values" as well. However, this probably should go also—and now it did.

```

583 % \gdef\f@encoding{#1}%
584 \gdef\default@family{#2}%
585 \gdef\default@series{#3}%
586 \gdef\default@shape{#4}%
587 }
588 </2ekernel | latexrelease>
589 <latexrelease>\EndIncludeInRelease
590 <latexrelease>\IncludeInRelease{0000/00/00}%
591 <latexrelease> \DeclareErrorFont{No side effects please}%
592 <latexrelease>
593 <latexrelease>\def\DeclareErrorFont#1#2#3#4#5{%
594 <latexrelease> \xdef\error@fontshape{%
595 <latexrelease> \noexpand\expandafter\noexpand\split@name\noexpand\string
596 <latexrelease> \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
597 <latexrelease> \noexpand\@nil}%
598 <latexrelease> \gdef\default@family{#2}%

```

```

599 <latexrelease> \gdef\default@series{#3}%
600 <latexrelease> \gdef\default@shape{#4}%
601 <latexrelease> \global\let\f@family\default@family
602 <latexrelease> \global\let\f@series\default@series
603 <latexrelease> \global\let\f@shape\default@shape
604 <latexrelease> \gdef\f@size{#5}%
605 <latexrelease> \gdef\f@baselineskip{#5pt}%
606 <latexrelease>}
607 <latexrelease>\EndIncludeInRelease
608 <*2ekernel>
609 \@onlypreamble\DeclareErrorFont

```

(End of definition for \DeclareErrorFont.)

\wrong@fontshape Before we come to the macro `\extract@font` we have to take care of unknown `\curr@fontshape` combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails T_EX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```

610 </2ekernel>
611 <latexrelease>\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
612 <latexrelease> \Font substitution in preamble}%
613 <*2ekernel | latexrelease>
614 \def\wrong@fontshape{%
615 \csname D@f@encoding\endcsname % install defaults if in math

```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```

616 \edef\reserved@a{\csname\curr@fontshape\endcsname}%
617 \ifx\last@fontshape\reserved@a
618 \errmessage{Corrupted NFSS tables}%
619 \error@fontshape
620 \else

```

Then we warn the user about the mess and set the shape to its default.

```

621 \let\f@shape\default@shape

```

If the combination is not known, try the default *series*.

```

622 \expandafter\ifx\csname\curr@fontshape\endcsname\relax
623 \let\f@series\default@series

```

If this is still undefined, try the default *family*. Otherwise give up. We never try to change the encoding scheme!

```

624 \expandafter
625 \ifx\csname\curr@fontshape\endcsname\relax
626 \let\f@family\default@family

```

If we change the font family and we are in the preamble then the corresponding `.fd` file may not been loaded yet. Therefore we try this now. Otherwise equating the requested font shape with the finally selected fontshape below will fail and can result in “NFSS tables corrupted”. After begin document that will not happen as all `.fd` files involved in substitution are loaded at `\begin{document}`.

```

627 \begingroup
628 \try@load@fontshape
629 \endgroup
630 \fi \fi
631 \fi

```

At this point a valid `\curr@fontshape` combination must have been found. We inform the user about this fact.

The `\expandafter\string` here stops $\TeX$ adding the space that it usually puts after command names in messages. The similar construction with `\@undefined` just produces ‘undefined’, but saves a few tokens.

`\@wrong@font@char` is locally redefined in `\UseTextSymbol` from its normal (empty) definition, to report the symbol generating the font switch.

```
632 \font@warning{Font shape ‘\expandafter\string\reserved@a’
633 \expandafter\@gobble\string\@undefined\MessageBreak
634 using ‘\curr@fontshape’ instead\@wrong@font@char}%
635 \global\let\last@fontshape\reserved@a
```

We change `\@defaultsubs` to produce a warning at the end of the document. The macro `\@defaultsubs` is initially `\relax` but gets changed here if some default font substitution happens. It is then executed in `\enddocument`.

```
636 \gdef\@defaultsubs{%
637 \font@warning{Some font shapes were not available, defaults
638 substituted.\@gobbletwo}}%
```

If we substitute a `\curr@fontshape` combination by the default one we don’t want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally `\let` the macro corresponding to the wanted combination equal to its substitution. This requires the use of four `\expandafter`’s since `\csname...\endcsname` has to be expanded before `\reserved@a` (i.e. the requested combination), and this must happen before the `\let` is executed.

```
639 \global\expandafter\expandafter\expandafter\let
640 \expandafter\reserved@a
641 \csname\curr@fontshape\endcsname
```

Now we can redefine `\font@name` accordingly. This *must* be done globally since it might occur in the group opened by `\define@newfont`. If we would this definition were local the closing `\endgroup` there would restore the old meaning of `\font@name` and then switch to the wrong font at the end of `\selectfont` although the correct font was loaded.

```
642 \xdef\font@name{%
643 \csname\curr@fontshape/\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
644 \pickup@font}
645 </2ekernel|latexrelease>
646 <latexrelease>\EndIncludeInRelease
647 <latexrelease>\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
648 <latexrelease> {Font substitution in preamble}%
649 <latexrelease>\def\wrong@fontshape{%
650 <latexrelease> \csname D@f@encoding\endcsname
651 <latexrelease> \edef\reserved@a{\csname\curr@fontshape\endcsname}%
652 <latexrelease> \ifx\last@fontshape\reserved@a
653 <latexrelease> \errmessage{Corrupted NFSS tables}%
654 <latexrelease> \error@fontshape
655 <latexrelease> \else
656 <latexrelease> \let\f@shape\default@shape
657 <latexrelease> \expandafter\ifx\csname\curr@fontshape\endcsname\relax
658 <latexrelease> \let\f@series\default@series
659 <latexrelease> \expandafter
```

```

660 <latexrelease>          \ifx\curname\curr@fontshape\endcurname\relax
661 <latexrelease>          \let\f@family\default@family
662 <latexrelease>          \fi \fi
663 <latexrelease>    \fi
664 <latexrelease>    \@font@warning{Font shape
665 <latexrelease>          '\expandafter\string\reserved@a'
666 <latexrelease>          \expandafter\@gobble\string\@undefined
667 <latexrelease>          \MessageBreak
668 <latexrelease>          using '\curr@fontshape' instead\@wrong@font@char}%
669 <latexrelease>    \global\let\last@fontshape\reserved@a
670 <latexrelease>    \gdef\@defaultsubs{%
671 <latexrelease>        \@font@warning{Some font shapes were not available,
672 <latexrelease>                defaults substituted.\@gobbletwo}}%
673 <latexrelease>    \global\expandafter\expandafter\expandafter\let
674 <latexrelease>        \expandafter\reserved@a
675 <latexrelease>        \curname\curr@fontshape\endcurname
676 <latexrelease>    \xdef\font@name{%
677 <latexrelease>        \curname\curr@fontshape/\f@size\endcurname}%
678 <latexrelease>    \pickup@font}
679 <latexrelease>\EndIncludeInRelease
680 <*2ekernel>

```

(End of definition for \wrong@fontshape.)

\@wrong@font@char Normally empty but redefined in \UseTextSymbol so that the Font shape undefined message can refer to the symbol causing the problem.

```
681 \let\@wrong@font@char\@empty
```

(End of definition for \@wrong@font@char.)

\@@defaultsubs See above.

```
\@defaultsubs 682 \let\@defaultsubs\relax
```

(End of definition for \@@defaultsubs and \@defaultsubs.)

\strip@prefix In \extract@font we will need a way to recover the replacement text of a macro. This is done by the primitive \meaning together with the macro \strip@prefix (for the details see appendix D of the T_EXbook, p. 382).

```
683 \def\strip@prefix#1>{\}
```

(End of definition for \strip@prefix.)

4 Assigning math fonts to *versions*

\install@mathalphabet This is just another name for \gdef but we can redefine it if necessary later on.

```
684 \let\install@mathalphabet\gdef
```

(End of definition for \install@mathalphabet.)

\math@fonts

```
685 \let\math@fonts\@empty
```

(End of definition for \math@fonts.)

`\select@group` `\select@group` has four arguments: the new $\langle\text{math alphabet identifier}\rangle$ (a control sequence), the $\langle\text{math group number}\rangle$, the extra macro for math mode and the `\curr@fontshape` definition macro name. We first check if we are in math mode.

```
686 %\def\select@group#1#2#3{\relax\ifmmode
```

We do these things locally using `\begingroup` instead of `\bgroup` to avoid the appearance of an empty Ord atom on the math list.

```
687 % \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
688 % \escapechar\m@ne
```

```
689 % \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
690 % \globaldefs\@ne \math@fonts
```

... and close the group to restore `\globaldefs` and `\escapechar`.

```
691 % \endgroup
```

As long as no *size* or *version* change occurs the $\langle\text{math alphabet identifier}\rangle$ should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new $\langle\text{math alphabet identifier}\rangle$) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro. The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the $\langle\text{math alphabet identifier}\rangle$ and then calling the new definition to switch to the wanted $\langle\text{math group}\rangle$. Now we define the $\langle\text{math alphabet identifier}\rangle$ as a call to the `\use@mathgroup` command.

```
692 % \xdef#1{\noexpand\use@mathgroup\noexpand#2%
```

```
693 % {\number\csname c@mv@\math@version\endcsname}}%
```

But this is not sufficient, as we learned the hard way. The problem here is that the loading of the fonts that comprise the alphabet identifier `#1`, as well as the necessary math font assignments is deferred until it is used. This is OK so far, but if the fonts are switched within the current formula (which may happen if a sub-formula is a box that contains a math version switch) the font assignments for `#1` are not restored unless `#1` is used again. This is disastrous since TeX sees the wrong fonts at the end of the math formula, when it converts the math list into a horizontal list.

This is taken into account as follows: When a math alphabet identifier is used for the first time in a certain version it modifies the corresponding macro `\mv@<version>` so that it calls `\getanddefine@fonts` directly in future as well. We use the macro `\extract@alph@from@version` to do this. It takes the math alphabet identifier `#1` and the math version macro as arguments.

```
694 % \expandafter\extract@alph@from@version
```

```
695 % \csname mv@\math@version\expandafter\endcsname
```

```
696 % \expandafter{\number\csname c@mv@\math@version\endcsname}%
```

```
697 % #1%
```

```
698 % \stepcounter{mv@\math@version}%
```

Finally, it is not possible to simply call the new definition since we have an argument (the third argument of `\use@mathgroup` or more exactly the argument of `\math@egroup` if the `margid` option is in force) which would swallow our closing `\fi`. So we use the `\expandafter` technique to remove the `\fi` before the `\use@mathgroup` is expanded.

```
699 %\expandafter #1\fi}
```

(End of definition for `\select@group`.)

`\extract@alph@from@version`

We proceed to the definition of the macro `\extract@alph@from@version`. As stated above, it takes a math alphabet identifier and a math version macro (e.g. `\mv@normal`) as its arguments.

```
700 \def\extract@alph@from@version#1#2#3{%
```

To extract and replace the definition of math alphabet identifier `#3` in macro `#1` we have to recall how this definition looks like: Somewhere in the replacement text of `#1` there is the sequence

```
\install@mathalphabet<math alphabet identifier> #3{%
  <Definitions for >#3}
```

Hence, the first thing we do is to extract the tokens preceding this definitions, the definition itself, and the tokens following it. To this end we define one auxiliary macro `\reserved@a`.

```
701 \def\reserved@a##1\install@mathalphabet#3##2##3\@nil{%
```

When `\reserved@a` is expanded, it will have the tokens preceding the definition in question in its first argument (`##1`), the following tokens in its third argument (`##3`), and the replacement text for the math alphabet identifier `#3` in its second argument. (`##2`). This is then recorded for later use in a temporary macro `\reserved@b`.

```
702 \def\reserved@b{##2}%
```

Additionally, we define a macro `\reserved@c` to reconstruct the definitions for the math version in question from the tokens that will remain unchanged (`##1` and `##3`) and the yet to build new definitions for the math alphabet identifier `#3`.

```
703 \def\reserved@c####1{\gdef#1{##1####1##3}}%
```

Then we execute our auxiliary macro.

```
704 \expandafter\reserved@a#1\@nil
```

OK, so now we have to build the new definition for `#3`. To do so, we first extract the interesting parts out of the old one. The old definition looks like:

```
\select@group<math alphabet identifier>
  <math group number><math extra part>
<curr@fontshape definition>
```

So we define a new temporary macro `\reserved@a` that extracts these parts.

```
705 \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
706 \reserved@c{%
707 \getanddefine@fonts{#2}##2%
708 \install@mathalphabet#3{%
709 \relax\ifmmode \else \non@alpherr#3\fi
710 \use@mathgroup##1{#2}}%
```

In addition it defines the alphabet the way it should be used from now on.

```
711 \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
712 \use@mathgroup##1{#2}}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
713 \expandafter\reserved@a\reserved@b\@nil
714 }
```

(End of definition for \extract@alph@from@version.)

`\math@bgroup` Here are the default definitions for `\math@bgroup` and `\math@egroup`. We use `\bgroup` instead of `\begingroup` to avoid ‘leaking out’ of style changes. This has the side effect of always producing mathord atoms.

```
715 \let\math@bgroup\bgroup
716 \def\math@egroup#1{#1\egroup}
```

(End of definition for \math@bgroup and \math@egroup.)

`\calculate@math@sizes` Here is the default definition for `\calculate@math@sizes` a more elaborate interface is under testing in `mthscale.sty`.

```
717 \gdef\calculate@math@sizes{%
718 \font@info{Calculating\space math\space sizes\space for\space
719 size\space <\f@size>}%
720 \dimen@ \f@size \p@
721 \tempdimb \defaultscriptratio \dimen@
722 \dimen@ \defaultscriptscriptratio \dimen@
723 \expandafter\xdef\csname S@\f@size\endcsname{%
724 \gdef\noexpand\tf@size{\f@size}%
725 \gdef\noexpand\sf@size{\strip@pt\tempdimb}%
726 \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
727 \noexpand\math@fontstrue}}
```

(End of definition for \calculate@math@sizes.)

`\defaultscriptratio` The default ratio for math sizes is:
`\defaultscriptscriptratio` 1 to `\defaultscriptratio` to `\defaultscriptscriptratio`.
 By default this is 1 to .7 to .5.

```
728 \def\defaultscriptratio{.7}
729 \def\defaultscriptscriptratio{.5}
```

(End of definition for \defaultscriptratio and \defaultscriptscriptratio.)

`\noaccents@` If we don’t have a definition for `\noaccents@` we provide a dummy.

```
730 \ifx\noaccents@\@undefined
731 \let\noaccents@\@empty
732 \fi
```

(End of definition for \noaccents@.)

`\showhyphens` The `\showhyphens` command must be redefined since the version in `plain.tex` uses `\tenrm`. We have also made some further adjustments for its use in L^AT_EX.

```

733 </2ekernel>
734 <latexrelease>\IncludeInRelease{2017/01/01}{\showhyphens}%
735 <latexrelease>                {XeTeX support for \showhyphens}%
736 <*2ekernel|latexrelease>
737 \ifx\XeTeXcharclass\@undefined
Version for engines other than XeTEX.
738 \DeclareRobustCommand\showhyphens[1]{%
739   \setbox0\vbox{%
740     \color@begingroup
741     \tracinglostchars\z@
742     \everypar{}%
743     \parfillskip\z@skip\hsize\maxdimen
744     \normalfont
745     \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
746     \color@endgroup}}
747 \else

```

XeT_EX version. When using system fonts XeT_EX reports consecutive runs of characters as a single item in box logging, which means the standard `\showhyphens` does not work. This version typesets the text into a narrow box to force hyphenation and then reconstructs a horizontal list with explicit hyphens to generate the display. Note that the `lmr` OpenType font is forced, this works even if the characters are not in the font as hyphenation is attempted due to the width of the space and hyphen character. It would generate spurious Missing Character warnings in the log, these are however suppressed from the terminal and log output by ensuring that `\tracinglostchars` is locally zero.

```

748 \DeclareRobustCommand\showhyphens[1]{%
749   \setbox0\vbox{%
750     \usefont{TU}{lmr}{m}{n}%
751     \hsize 1sp %
752     \hbadness\@M
753     \hfuzz\maxdimen
754     \tracingonline\z@
755     \tracinglostchars\z@
756     \everypar={}%
757     \leftskip\z@skip
758     \rightskip\z@skip
759     \parfillskip\z@skip
760     \hyphenpenalty=-\@M
761     \pretolerance\m@ne
762     \interlinepenalty\z@
763     \clubpenalty\z@
764     \widowpenalty\z@
765     \brokenpenalty1127 %
766     \setbox\z@\hbox{}%
767     \noindent
768     \hskip\z@skip
769     #1%
770     \par

```

Note here we stop the loop if made no progress, non-removable items may mean that we can not process the whole list (which would be testable as `\lastnodetype=-1`).

```

771 \loop
772 \@tempswafalse
773 \ifnum\lastnodetype=11\unskip\@tempswatrue\fi
774 \ifnum\lastnodetype=12\unkern\@tempswatrue\fi
775 \ifnum\lastnodetype=13 %
776 \count@\lastpenalty
777 \unpenalty\@tempswatrue
778 \fi
779 \ifnum\lastnodetype=\@ne
780 \setbox\tw@\lastbox\@tempswatrue
781 \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
782 \ifnum\count@=1127 \else\ \fi
783 \unhbox0}%
784 \count@\z@
785 \fi
786 \if@tempswa
787 \repeat
788 \hbadness\z@
789 \hsize\maxdimen
790 \showboxdepth\z@
791 \tolerance\m@ne
792 \hyphenpenalty\z@
793 \noindent\unhbox\z@
794 }}
795 \fi

796 </2ekernel | latexrelease>
797 <latexrelease>\EndIncludeInRelease
798 <latexrelease>\IncludeInRelease{0000/00/00}{\showhyphens}%
799 <latexrelease> \XeTeX support for \showhyphens}%
800 <latexrelease>\gdef\showhyphens#1{%
801 <latexrelease> \setbox0\vbox{%
802 <latexrelease> \color@begingroup
803 <latexrelease> \everypar{%
804 <latexrelease> \parfillskip\z@skip\hsize\maxdimen
805 <latexrelease> \normalfont
806 <latexrelease> \pretolerance\m@ne\tolerance\m@ne
807 <latexrelease> \hbadness\z@\showboxdepth\z@\ #1%
808 <latexrelease> \color@endgroup}}
809 <latexrelease>\EndIncludeInRelease
810 <*2ekernel>

```

(End of definition for \showhyphens.)

\addto@hook We need a macro to add tokens to a hook.

```
811 \long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}
```

(End of definition for \addto@hook.)

\@vpt

```
812 \def\@vpt{5}
```

(End of definition for \@vpt.)

```

\@vipt
813 \def\@vipt{6}
      (End of definition for \@vipt.)

\@viipt
814 \def\@viipt{7}
      (End of definition for \@viipt.)

\@viipt
815 \def\@viipt{8}
      (End of definition for \@viipt.)

\@ixpt
816 \def\@ixpt{9}
      (End of definition for \@ixpt.)

\@xpt
817 \def\@xpt{10}
      (End of definition for \@xpt.)

\@xipt
818 \def\@xipt{10.95}
      (End of definition for \@xipt.)

\@xiipt
819 \def\@xiipt{12}
      (End of definition for \@xiipt.)

\@xivpt
820 \def\@xivpt{14.4}
      (End of definition for \@xivpt.)

\@xvipt
821 \def\@xvipt{17.28}
      (End of definition for \@xvipt.)

\@xxpt
822 \def\@xxpt{20.74}
      (End of definition for \@xxpt.)

\@xxvpt
823 \def\@xxvpt{24.88}
      (End of definition for \@xxvpt.)
824 </2ekernel>

```

File 25

ltfssaxes.dtx

This file contains the implementation for handling extra axes by splitting the series and the shape values into sub-categories. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX Font Selection Scheme.

```
1 <2ekernel>\message{NFSS axes,}
```

Everything in the this file got introduced 2020/02/02, so we use large rollback chunks, only interrupted if necessary.

```
2 <*2ekernel | latexrelease>
```

```
3 <latexrelease>\IncludeInRelease{2025/06/01}%
```

```
4 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
```

1 Changing the font series

In the original NFSS implementation the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. This meant that it was impossible to typeset a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

The new implementation now works differently, by looking at both the current value of `\f@series` and the requested new series; these are then used to select a new series value. Thus, if the current series is `c` and we ask for `b` we now get `bc`.

This is done by consulting a simple lookup table. This table is configurable (though most likely that flexibility will seldom if ever be needed). Adding or changing entries in this table is done with `\DeclareFontSeriesChangeRule`.

1.1 The series lookup table

`\DeclareFontSeriesChangeRule` The macro `\DeclareFontSeriesChangeRule` defines entries in a simple database (implemented as a set of commands) that define mappings from the current series and a requested new series to a result series (and additionally offers an alternative if the desired one is unavailable):

#1 current `\f@series`

#2 requested new series

#3 result (provided this series exists in the given font family)

#4 alternative result (if **#3** does not exist)

If an `.fd` file has its own substitution rules then **#3** exist and thus **#4** is not applied.

If there is no matching database entry, or if neither the result nor the alternative result exists in this font family, then the requested new series is used (which then may trigger substitutions later on).

```
5 \def\DeclareFontSeriesChangeRule#1#2#3#4{%
```

```
6 \@namedef{series@#1@#2}{{#3}{#4}}}
```

(End of definition for `\DeclareFontSeriesChangeRule`.)

1.2 Mapping rules for series changes

The rules set up use explicit series values not `\..default` indirections; my current feeling is that this is in fact better.

With 9 weight values (from `ul` to `ub`) and 9 width values (from `uc` to `ux`), this table is now rather large (far more than 1000 entries), but, on the other hand, the table doesn't change and accessing rules is fast when using a table implemented in this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?` is provided.
- We support “`m<width>`” and “`<weight>m`”, e.g., `mec` to mean “go to medium weight and extra-condensed width”. At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the requested series is used unconditionally. This means that we usually do not need entries where the second argument (the requested series) and the third argument (the result series) are identical (unless we want to use the fourth argument to specify an alternative result series). In particular, this means:
 - Any request for `m` needs no entry, i.e., there are no entries which have `m` as the second argument.
 - Any request to set both weight and width (e.g., `sbx` or `ulc`) needs no entry. For that reason, there are no entries which have a weight+width as second argument (except for cases involving `bx`, see below). In particular, this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced to `b`, or `mc` (medium weight condensed) which becomes `c` as a result.

There is one exception: we do have rules for a request for `bx`; this is because the Computer Modern fonts (or related families like Latin Modern) need `bx` as their bold series default (`\bfdefault`), since they only have a `b` series for a restricted set of font faces. Many other font families have only `b` but no `bx`, so we have explicit rules that fall back from `bx` to `b`.

- For each combination of a non-`m` `<weight>` and a non-`m` `<width>`, there are 19 entries which have “`<weight><width>`” as first argument: 8 of them have the weight values from `ul` to `ub` except `m` as second argument, another 8 have the width values from `uc` to `ux` except `m` as second argument, and (further down in this file) there is one entry which has `bx` as second argument and two further entries which have `m?` and `?m` as second argument. Rules which have `m` or a weight+width (other than `bx`) as second argument aren't needed (see above).
- For each non-`m` `<weight>`, there are at least 11 entries which have “`<weight>`” as first argument: 8 of them have the width values from `uc` to `ux` except `m` as second argument, and (further down in this file) there is one entry which has `bx` as second argument and two further entries which have `m?` and `?m` as second argument. Rules which have a single weight value as second argument aren't needed because the

second and third arguments would then be identical. In some cases, there are extra rules which make use of the fourth argument to specify an alternative result series.

- Similarly, for each non-*m* *<width>*, there are at least 11 entries which have “*<width>*” as first argument: 8 of them have the weight values from *ul* to *ub* except *m* as second argument, and (further down in this file) there is one entry which has *bx* as second argument and two further entries which have *m?* and *?m* as second argument. Rules which have a single width value as second argument aren’t needed because the second and third arguments would then be identical. In some cases, there are extra rules which make use of the fourth argument to specify an alternative result series.
- Only a few entries have “alternative” values.

The idea is that you don’t want the normal substitution to kick in because that would reset the shape first and it may be better to stay with *b* when a change to *c* is requested and *bc* doesn’t exist, than to go to first change the shape to *n* and then find that *bc/n* doesn’t exist either and thus ending up with *m/n*.

We do this, for example, when *sb*, *eb*, or *ub* is requested but can’t be fulfilled. In that case it is better to try to stay with some sort of bold rather than ending up with *m*. There are some other cases where an “alternative” value is specified; these are explained below in the appropriate places.

```

7 \DeclareFontSeriesChangeRule {uluc}{ul} {uluc} {}
8 \DeclareFontSeriesChangeRule {uluc}{el} {eluc} {}
9 \DeclareFontSeriesChangeRule {uluc}{l} {luc} {}
10 \DeclareFontSeriesChangeRule {uluc}{sl} {sluc} {}
11 \DeclareFontSeriesChangeRule {uluc}{sb} {sbuc} {buc}
12 \DeclareFontSeriesChangeRule {uluc}{b} {buc} {}
13 \DeclareFontSeriesChangeRule {uluc}{eb} {ebuc} {buc}
14 \DeclareFontSeriesChangeRule {uluc}{ub} {ubuc} {buc}
15 \DeclareFontSeriesChangeRule {uluc}{uc} {uluc} {}
16 \DeclareFontSeriesChangeRule {uluc}{ec} {ulec} {}
17 \DeclareFontSeriesChangeRule {uluc}{c} {ulc} {}
18 \DeclareFontSeriesChangeRule {uluc}{sc} {ulsc} {}
19 \DeclareFontSeriesChangeRule {uluc}{sx} {ulsx} {}
20 \DeclareFontSeriesChangeRule {uluc}{x} {ulx} {}
21 \DeclareFontSeriesChangeRule {uluc}{ex} {ulex} {}
22 \DeclareFontSeriesChangeRule {uluc}{ux} {ulux} {}

23 \DeclareFontSeriesChangeRule {ulec}{ul} {ulec} {}
24 \DeclareFontSeriesChangeRule {ulec}{el} {elec} {}
25 \DeclareFontSeriesChangeRule {ulec}{l} {lec} {}
26 \DeclareFontSeriesChangeRule {ulec}{sl} {slec} {}
27 \DeclareFontSeriesChangeRule {ulec}{sb} {sbec} {bec}
28 \DeclareFontSeriesChangeRule {ulec}{b} {bec} {}
29 \DeclareFontSeriesChangeRule {ulec}{eb} {ebec} {bec}
30 \DeclareFontSeriesChangeRule {ulec}{ub} {ubec} {bec}
31 \DeclareFontSeriesChangeRule {ulec}{uc} {uluc} {}
32 \DeclareFontSeriesChangeRule {ulec}{ec} {ulec} {}
33 \DeclareFontSeriesChangeRule {ulec}{c} {ulc} {}
34 \DeclareFontSeriesChangeRule {ulec}{sc} {ulsc} {}
35 \DeclareFontSeriesChangeRule {ulec}{sx} {ulsx} {}
36 \DeclareFontSeriesChangeRule {ulec}{x} {ulx} {}
37 \DeclareFontSeriesChangeRule {ulec}{ex} {ulex} {}
38 \DeclareFontSeriesChangeRule {ulec}{ux} {ulux} {}

```

```

39 \DeclareFontSeriesChangeRule {ulc}{ul} {ulc} {}
40 \DeclareFontSeriesChangeRule {ulc}{el} {elc} {}
41 \DeclareFontSeriesChangeRule {ulc}{l} {lc} {}
42 \DeclareFontSeriesChangeRule {ulc}{sl} {slc} {}
43 \DeclareFontSeriesChangeRule {ulc}{sb} {sbc} {bc}
44 \DeclareFontSeriesChangeRule {ulc}{b} {bc} {}
45 \DeclareFontSeriesChangeRule {ulc}{eb} {ebc} {bc}
46 \DeclareFontSeriesChangeRule {ulc}{ub} {ubc} {bc}
47 \DeclareFontSeriesChangeRule {ulc}{uc} {uluc} {}
48 \DeclareFontSeriesChangeRule {ulc}{ec} {ulec} {}
49 \DeclareFontSeriesChangeRule {ulc}{c} {ulc} {}
50 \DeclareFontSeriesChangeRule {ulc}{sc} {ulsc} {}
51 \DeclareFontSeriesChangeRule {ulc}{sx} {ulsx} {}
52 \DeclareFontSeriesChangeRule {ulc}{x} {ulx} {}
53 \DeclareFontSeriesChangeRule {ulc}{ex} {ulex} {}
54 \DeclareFontSeriesChangeRule {ulc}{ux} {ulux} {}

55 \DeclareFontSeriesChangeRule {ulsc}{ul} {ulsc} {}
56 \DeclareFontSeriesChangeRule {ulsc}{el} {elsc} {}
57 \DeclareFontSeriesChangeRule {ulsc}{l} {lsc} {}
58 \DeclareFontSeriesChangeRule {ulsc}{sl} {slsc} {}
59 \DeclareFontSeriesChangeRule {ulsc}{sb} {sbsc} {bsc}
60 \DeclareFontSeriesChangeRule {ulsc}{b} {bsc} {}
61 \DeclareFontSeriesChangeRule {ulsc}{eb} {ebsc} {bsc}
62 \DeclareFontSeriesChangeRule {ulsc}{ub} {ubsc} {bsc}
63 \DeclareFontSeriesChangeRule {ulsc}{uc} {uluc} {}
64 \DeclareFontSeriesChangeRule {ulsc}{ec} {ulec} {}
65 \DeclareFontSeriesChangeRule {ulsc}{c} {ulc} {}
66 \DeclareFontSeriesChangeRule {ulsc}{sc} {ulsc} {}
67 \DeclareFontSeriesChangeRule {ulsc}{sx} {ulsx} {}
68 \DeclareFontSeriesChangeRule {ulsc}{x} {ulx} {}
69 \DeclareFontSeriesChangeRule {ulsc}{ex} {ulex} {}
70 \DeclareFontSeriesChangeRule {ulsc}{ux} {ulux} {}

71 \DeclareFontSeriesChangeRule {ul}{uc} {uluc} {}
72 \DeclareFontSeriesChangeRule {ul}{ec} {ulec} {}
73 \DeclareFontSeriesChangeRule {ul}{c} {ulc} {}
74 \DeclareFontSeriesChangeRule {ul}{sc} {ulsc} {}
75 \DeclareFontSeriesChangeRule {ul}{sx} {ulsx} {}
76 \DeclareFontSeriesChangeRule {ul}{x} {ulx} {}
77 \DeclareFontSeriesChangeRule {ul}{ex} {ulex} {}
78 \DeclareFontSeriesChangeRule {ul}{ux} {ulux} {}
79 \DeclareFontSeriesChangeRule {ul}{sb} {sb} {b}
80 \DeclareFontSeriesChangeRule {ul}{eb} {eb} {b}
81 \DeclareFontSeriesChangeRule {ul}{ub} {ub} {b}

82 \DeclareFontSeriesChangeRule {ulsx}{ul} {ulsx} {}
83 \DeclareFontSeriesChangeRule {ulsx}{el} {elsx} {}
84 \DeclareFontSeriesChangeRule {ulsx}{l} {lsx} {}
85 \DeclareFontSeriesChangeRule {ulsx}{sl} {slsx} {}
86 \DeclareFontSeriesChangeRule {ulsx}{sb} {sbsx} {bsx}
87 \DeclareFontSeriesChangeRule {ulsx}{b} {bsx} {}
88 \DeclareFontSeriesChangeRule {ulsx}{eb} {ebsx} {bsx}
89 \DeclareFontSeriesChangeRule {ulsx}{ub} {ubsx} {bsx}
90 \DeclareFontSeriesChangeRule {ulsx}{uc} {uluc} {}
91 \DeclareFontSeriesChangeRule {ulsx}{ec} {ulec} {}

```

```

92 \DeclareFontSeriesChangeRule {ulsx}{c} {ulc} {}
93 \DeclareFontSeriesChangeRule {ulsx}{sc} {ulsc} {}
94 \DeclareFontSeriesChangeRule {ulsx}{sx} {ulsx} {}
95 \DeclareFontSeriesChangeRule {ulsx}{x} {ulx} {}
96 \DeclareFontSeriesChangeRule {ulsx}{ex} {ulex} {}
97 \DeclareFontSeriesChangeRule {ulsx}{ux} {ulux} {}

98 \DeclareFontSeriesChangeRule {ulx}{ul} {ulx} {}
99 \DeclareFontSeriesChangeRule {ulx}{el} {elx} {}
100 \DeclareFontSeriesChangeRule {ulx}{l} {lx} {}
101 \DeclareFontSeriesChangeRule {ulx}{sl} {slx} {}
102 \DeclareFontSeriesChangeRule {ulx}{sb} {sbx} {bx}
103 \DeclareFontSeriesChangeRule {ulx}{b} {bx} {}
104 \DeclareFontSeriesChangeRule {ulx}{eb} {ebx} {bx}
105 \DeclareFontSeriesChangeRule {ulx}{ub} {ubx} {bx}
106 \DeclareFontSeriesChangeRule {ulx}{uc} {uluc} {}
107 \DeclareFontSeriesChangeRule {ulx}{ec} {ulec} {}
108 \DeclareFontSeriesChangeRule {ulx}{c} {ulc} {}
109 \DeclareFontSeriesChangeRule {ulx}{sc} {ulsc} {}
110 \DeclareFontSeriesChangeRule {ulx}{sx} {ulsx} {}
111 \DeclareFontSeriesChangeRule {ulx}{x} {ulx} {}
112 \DeclareFontSeriesChangeRule {ulx}{ex} {ulex} {}
113 \DeclareFontSeriesChangeRule {ulx}{ux} {ulux} {}

114 \DeclareFontSeriesChangeRule {ulex}{ul} {ulex} {}
115 \DeclareFontSeriesChangeRule {ulex}{el} {ele} {}
116 \DeclareFontSeriesChangeRule {ulex}{l} {lex} {}
117 \DeclareFontSeriesChangeRule {ulex}{sl} {slex} {}
118 \DeclareFontSeriesChangeRule {ulex}{sb} {sbex} {bex}
119 \DeclareFontSeriesChangeRule {ulex}{b} {bex} {}
120 \DeclareFontSeriesChangeRule {ulex}{eb} {ebex} {bex}
121 \DeclareFontSeriesChangeRule {ulex}{ub} {ubex} {bex}
122 \DeclareFontSeriesChangeRule {ulex}{uc} {uluc} {}
123 \DeclareFontSeriesChangeRule {ulex}{ec} {ulec} {}
124 \DeclareFontSeriesChangeRule {ulex}{c} {ulc} {}
125 \DeclareFontSeriesChangeRule {ulex}{sc} {ulsc} {}
126 \DeclareFontSeriesChangeRule {ulex}{sx} {ulsx} {}
127 \DeclareFontSeriesChangeRule {ulex}{x} {ulx} {}
128 \DeclareFontSeriesChangeRule {ulex}{ex} {ulex} {}
129 \DeclareFontSeriesChangeRule {ulex}{ux} {ulux} {}

130 \DeclareFontSeriesChangeRule {ulux}{ul} {ulux} {}
131 \DeclareFontSeriesChangeRule {ulux}{el} {elux} {}
132 \DeclareFontSeriesChangeRule {ulux}{l} {lux} {}
133 \DeclareFontSeriesChangeRule {ulux}{sl} {slux} {}
134 \DeclareFontSeriesChangeRule {ulux}{sb} {sbux} {bux}
135 \DeclareFontSeriesChangeRule {ulux}{b} {bux} {}
136 \DeclareFontSeriesChangeRule {ulux}{eb} {ebux} {bux}
137 \DeclareFontSeriesChangeRule {ulux}{ub} {ubux} {bux}
138 \DeclareFontSeriesChangeRule {ulux}{uc} {uluc} {}
139 \DeclareFontSeriesChangeRule {ulux}{ec} {ulec} {}
140 \DeclareFontSeriesChangeRule {ulux}{c} {ulc} {}
141 \DeclareFontSeriesChangeRule {ulux}{sc} {ulsc} {}
142 \DeclareFontSeriesChangeRule {ulux}{sx} {ulsx} {}
143 \DeclareFontSeriesChangeRule {ulux}{x} {ulx} {}
144 \DeclareFontSeriesChangeRule {ulux}{ex} {ulex} {}

```



```

145 \DeclareFontSeriesChangeRule {ulux}{ux} {ulux} {}
146 \DeclareFontSeriesChangeRule {eluc}{ul} {uluc} {}
147 \DeclareFontSeriesChangeRule {eluc}{el} {eluc} {}
148 \DeclareFontSeriesChangeRule {eluc}{l} {luc} {}
149 \DeclareFontSeriesChangeRule {eluc}{sl} {sluc} {}
150 \DeclareFontSeriesChangeRule {eluc}{sb} {sbuc} {buc}
151 \DeclareFontSeriesChangeRule {eluc}{b} {buc} {}
152 \DeclareFontSeriesChangeRule {eluc}{eb} {ebuc} {buc}
153 \DeclareFontSeriesChangeRule {eluc}{ub} {ubuc} {buc}
154 \DeclareFontSeriesChangeRule {eluc}{uc} {eluc} {}
155 \DeclareFontSeriesChangeRule {eluc}{ec} {elec} {}
156 \DeclareFontSeriesChangeRule {eluc}{c} {elc} {}
157 \DeclareFontSeriesChangeRule {eluc}{sc} {elsc} {}
158 \DeclareFontSeriesChangeRule {eluc}{sx} {elsx} {}
159 \DeclareFontSeriesChangeRule {eluc}{x} {elx} {}
160 \DeclareFontSeriesChangeRule {eluc}{ex} {elex} {}
161 \DeclareFontSeriesChangeRule {eluc}{ux} {elux} {}

162 \DeclareFontSeriesChangeRule {elec}{ul} {ulec} {}
163 \DeclareFontSeriesChangeRule {elec}{el} {elec} {}
164 \DeclareFontSeriesChangeRule {elec}{l} {lec} {}
165 \DeclareFontSeriesChangeRule {elec}{sl} {slec} {}
166 \DeclareFontSeriesChangeRule {elec}{sb} {sbec} {bec}
167 \DeclareFontSeriesChangeRule {elec}{b} {bec} {}
168 \DeclareFontSeriesChangeRule {elec}{eb} {ebec} {bec}
169 \DeclareFontSeriesChangeRule {elec}{ub} {ubec} {bec}
170 \DeclareFontSeriesChangeRule {elec}{uc} {eluc} {}
171 \DeclareFontSeriesChangeRule {elec}{ec} {elec} {}
172 \DeclareFontSeriesChangeRule {elec}{c} {elc} {}
173 \DeclareFontSeriesChangeRule {elec}{sc} {elsc} {}
174 \DeclareFontSeriesChangeRule {elec}{sx} {elsx} {}
175 \DeclareFontSeriesChangeRule {elec}{x} {elx} {}
176 \DeclareFontSeriesChangeRule {elec}{ex} {elex} {}
177 \DeclareFontSeriesChangeRule {elec}{ux} {elux} {}

178 \DeclareFontSeriesChangeRule {elc}{ul} {ulc} {}
179 \DeclareFontSeriesChangeRule {elc}{el} {elc} {}
180 \DeclareFontSeriesChangeRule {elc}{l} {lc} {}
181 \DeclareFontSeriesChangeRule {elc}{sl} {slc} {}
182 \DeclareFontSeriesChangeRule {elc}{sb} {sbc} {bc}
183 \DeclareFontSeriesChangeRule {elc}{b} {bc} {}
184 \DeclareFontSeriesChangeRule {elc}{eb} {ebc} {bc}
185 \DeclareFontSeriesChangeRule {elc}{ub} {ubc} {bc}
186 \DeclareFontSeriesChangeRule {elc}{uc} {eluc} {}
187 \DeclareFontSeriesChangeRule {elc}{ec} {elec} {}
188 \DeclareFontSeriesChangeRule {elc}{c} {elc} {}
189 \DeclareFontSeriesChangeRule {elc}{sc} {elsc} {}
190 \DeclareFontSeriesChangeRule {elc}{sx} {elsx} {}
191 \DeclareFontSeriesChangeRule {elc}{x} {elx} {}
192 \DeclareFontSeriesChangeRule {elc}{ex} {elex} {}
193 \DeclareFontSeriesChangeRule {elc}{ux} {elux} {}

194 \DeclareFontSeriesChangeRule {elsc}{ul} {ulsc} {}
195 \DeclareFontSeriesChangeRule {elsc}{el} {elsc} {}
196 \DeclareFontSeriesChangeRule {elsc}{l} {lsc} {}
197 \DeclareFontSeriesChangeRule {elsc}{sl} {slsc} {}

```

```

198 \DeclareFontSeriesChangeRule {elsc}{sb} {sbsc} {bsc}
199 \DeclareFontSeriesChangeRule {elsc}{b} {bsc} {}
200 \DeclareFontSeriesChangeRule {elsc}{eb} {ebsc} {bsc}
201 \DeclareFontSeriesChangeRule {elsc}{ub} {ubsc} {bsc}
202 \DeclareFontSeriesChangeRule {elsc}{uc} {eluc} {}
203 \DeclareFontSeriesChangeRule {elsc}{ec} {elec} {}
204 \DeclareFontSeriesChangeRule {elsc}{c} {elc} {}
205 \DeclareFontSeriesChangeRule {elsc}{sc} {elsc} {}
206 \DeclareFontSeriesChangeRule {elsc}{sx} {elsx} {}
207 \DeclareFontSeriesChangeRule {elsc}{x} {elx} {}
208 \DeclareFontSeriesChangeRule {elsc}{ex} {elex} {}
209 \DeclareFontSeriesChangeRule {elsc}{ux} {elux} {}

210 \DeclareFontSeriesChangeRule {el}{uc} {eluc} {}
211 \DeclareFontSeriesChangeRule {el}{ec} {elec} {}
212 \DeclareFontSeriesChangeRule {el}{c} {elc} {}
213 \DeclareFontSeriesChangeRule {el}{sc} {elsc} {}
214 \DeclareFontSeriesChangeRule {el}{sx} {elsx} {}
215 \DeclareFontSeriesChangeRule {el}{x} {elx} {}
216 \DeclareFontSeriesChangeRule {el}{ex} {elex} {}
217 \DeclareFontSeriesChangeRule {el}{ux} {elux} {}
218 \DeclareFontSeriesChangeRule {el}{sb} {sb} {b}
219 \DeclareFontSeriesChangeRule {el}{eb} {eb} {b}
220 \DeclareFontSeriesChangeRule {el}{ub} {ub} {b}

221 \DeclareFontSeriesChangeRule {elsx}{ul} {ulsx} {}
222 \DeclareFontSeriesChangeRule {elsx}{el} {elsx} {}
223 \DeclareFontSeriesChangeRule {elsx}{l} {lsx} {}
224 \DeclareFontSeriesChangeRule {elsx}{sl} {slsx} {}
225 \DeclareFontSeriesChangeRule {elsx}{sb} {sbsx} {bsx}
226 \DeclareFontSeriesChangeRule {elsx}{b} {bsx} {}
227 \DeclareFontSeriesChangeRule {elsx}{eb} {ebsx} {bsx}
228 \DeclareFontSeriesChangeRule {elsx}{ub} {ubsx} {bsx}
229 \DeclareFontSeriesChangeRule {elsx}{uc} {eluc} {}
230 \DeclareFontSeriesChangeRule {elsx}{ec} {elec} {}
231 \DeclareFontSeriesChangeRule {elsx}{c} {elc} {}
232 \DeclareFontSeriesChangeRule {elsx}{sc} {elsc} {}
233 \DeclareFontSeriesChangeRule {elsx}{sx} {elsx} {}
234 \DeclareFontSeriesChangeRule {elsx}{x} {elx} {}
235 \DeclareFontSeriesChangeRule {elsx}{ex} {elex} {}
236 \DeclareFontSeriesChangeRule {elsx}{ux} {elux} {}

237 \DeclareFontSeriesChangeRule {elx}{ul} {ulx} {}
238 \DeclareFontSeriesChangeRule {elx}{el} {elx} {}
239 \DeclareFontSeriesChangeRule {elx}{l} {lx} {}
240 \DeclareFontSeriesChangeRule {elx}{sl} {slx} {}
241 \DeclareFontSeriesChangeRule {elx}{sb} {sbx} {bx}
242 \DeclareFontSeriesChangeRule {elx}{b} {bx} {}
243 \DeclareFontSeriesChangeRule {elx}{eb} {ebx} {bx}
244 \DeclareFontSeriesChangeRule {elx}{ub} {ubx} {bx}
245 \DeclareFontSeriesChangeRule {elx}{uc} {eluc} {}
246 \DeclareFontSeriesChangeRule {elx}{ec} {elec} {}
247 \DeclareFontSeriesChangeRule {elx}{c} {elc} {}
248 \DeclareFontSeriesChangeRule {elx}{sc} {elsc} {}
249 \DeclareFontSeriesChangeRule {elx}{sx} {elsx} {}
250 \DeclareFontSeriesChangeRule {elx}{x} {elx} {}

```

```

251 \DeclareFontSeriesChangeRule {elx}{ex} {elex} {}
252 \DeclareFontSeriesChangeRule {elx}{ux} {elux} {}
253 \DeclareFontSeriesChangeRule {elex}{ul} {ulex} {}
254 \DeclareFontSeriesChangeRule {elex}{el} {elex} {}
255 \DeclareFontSeriesChangeRule {elex}{l} {lex} {}
256 \DeclareFontSeriesChangeRule {elex}{sl} {slex} {}
257 \DeclareFontSeriesChangeRule {elex}{sb} {sbex} {bex}
258 \DeclareFontSeriesChangeRule {elex}{b} {bex} {}
259 \DeclareFontSeriesChangeRule {elex}{eb} {ebex} {bex}
260 \DeclareFontSeriesChangeRule {elex}{ub} {ubex} {bex}
261 \DeclareFontSeriesChangeRule {elex}{uc} {eluc} {}
262 \DeclareFontSeriesChangeRule {elex}{ec} {elec} {}
263 \DeclareFontSeriesChangeRule {elex}{c} {elc} {}
264 \DeclareFontSeriesChangeRule {elex}{sc} {elsc} {}
265 \DeclareFontSeriesChangeRule {elex}{sx} {elsx} {}
266 \DeclareFontSeriesChangeRule {elex}{x} {elx} {}
267 \DeclareFontSeriesChangeRule {elex}{ex} {elex} {}
268 \DeclareFontSeriesChangeRule {elex}{ux} {elux} {}
269 \DeclareFontSeriesChangeRule {elux}{ul} {ulux} {}
270 \DeclareFontSeriesChangeRule {elux}{el} {elux} {}
271 \DeclareFontSeriesChangeRule {elux}{l} {lux} {}
272 \DeclareFontSeriesChangeRule {elux}{sl} {slux} {}
273 \DeclareFontSeriesChangeRule {elux}{sb} {sbux} {bux}
274 \DeclareFontSeriesChangeRule {elux}{b} {bux} {}
275 \DeclareFontSeriesChangeRule {elux}{eb} {ebux} {bux}
276 \DeclareFontSeriesChangeRule {elux}{ub} {ubux} {bux}
277 \DeclareFontSeriesChangeRule {elux}{uc} {eluc} {}
278 \DeclareFontSeriesChangeRule {elux}{ec} {elec} {}
279 \DeclareFontSeriesChangeRule {elux}{c} {elc} {}
280 \DeclareFontSeriesChangeRule {elux}{sc} {elsc} {}
281 \DeclareFontSeriesChangeRule {elux}{sx} {elsx} {}
282 \DeclareFontSeriesChangeRule {elux}{x} {elx} {}
283 \DeclareFontSeriesChangeRule {elux}{ex} {elex} {}
284 \DeclareFontSeriesChangeRule {elux}{ux} {elux} {}
285 \DeclareFontSeriesChangeRule {luc}{ul} {uluc} {}
286 \DeclareFontSeriesChangeRule {luc}{el} {eluc} {}
287 \DeclareFontSeriesChangeRule {luc}{l} {luc} {}
288 \DeclareFontSeriesChangeRule {luc}{sl} {sluc} {}
289 \DeclareFontSeriesChangeRule {luc}{sb} {sbuc} {buc}
290 \DeclareFontSeriesChangeRule {luc}{b} {buc} {}
291 \DeclareFontSeriesChangeRule {luc}{eb} {ebuc} {buc}
292 \DeclareFontSeriesChangeRule {luc}{ub} {ubuc} {buc}
293 \DeclareFontSeriesChangeRule {luc}{uc} {luc} {}
294 \DeclareFontSeriesChangeRule {luc}{ec} {lec} {}
295 \DeclareFontSeriesChangeRule {luc}{c} {lc} {}
296 \DeclareFontSeriesChangeRule {luc}{sc} {lsc} {}
297 \DeclareFontSeriesChangeRule {luc}{sx} {lsx} {}
298 \DeclareFontSeriesChangeRule {luc}{x} {lx} {}
299 \DeclareFontSeriesChangeRule {luc}{ex} {lex} {}
300 \DeclareFontSeriesChangeRule {luc}{ux} {lux} {}
301 \DeclareFontSeriesChangeRule {lec}{ul} {ulec} {}
302 \DeclareFontSeriesChangeRule {lec}{el} {elec} {}
303 \DeclareFontSeriesChangeRule {lec}{l} {lec} {}

```

```

304 \DeclareFontSeriesChangeRule {lec}{sl} {slec} {}
305 \DeclareFontSeriesChangeRule {lec}{sb} {sbec} {bec}
306 \DeclareFontSeriesChangeRule {lec}{b} {bec} {}
307 \DeclareFontSeriesChangeRule {lec}{eb} {ebec} {bec}
308 \DeclareFontSeriesChangeRule {lec}{ub} {ubec} {bec}
309 \DeclareFontSeriesChangeRule {lec}{uc} {luc} {}
310 \DeclareFontSeriesChangeRule {lec}{ec} {lec} {}
311 \DeclareFontSeriesChangeRule {lec}{c} {lc} {}
312 \DeclareFontSeriesChangeRule {lec}{sc} {lsc} {}
313 \DeclareFontSeriesChangeRule {lec}{sx} {lsx} {}
314 \DeclareFontSeriesChangeRule {lec}{x} {lx} {}
315 \DeclareFontSeriesChangeRule {lec}{ex} {lex} {}
316 \DeclareFontSeriesChangeRule {lec}{ux} {lux} {}

317 \DeclareFontSeriesChangeRule {lc}{ul} {ulc} {}
318 \DeclareFontSeriesChangeRule {lc}{el} {elc} {}
319 \DeclareFontSeriesChangeRule {lc}{l} {lc} {}
320 \DeclareFontSeriesChangeRule {lc}{sl} {slc} {}
321 \DeclareFontSeriesChangeRule {lc}{sb} {sbc} {bc}
322 \DeclareFontSeriesChangeRule {lc}{b} {bc} {}
323 \DeclareFontSeriesChangeRule {lc}{eb} {ebc} {bc}
324 \DeclareFontSeriesChangeRule {lc}{ub} {ubc} {bc}
325 \DeclareFontSeriesChangeRule {lc}{uc} {luc} {}
326 \DeclareFontSeriesChangeRule {lc}{ec} {lec} {}
327 \DeclareFontSeriesChangeRule {lc}{c} {lc} {}
328 \DeclareFontSeriesChangeRule {lc}{sc} {lsc} {}
329 \DeclareFontSeriesChangeRule {lc}{sx} {lsx} {}
330 \DeclareFontSeriesChangeRule {lc}{x} {lx} {}
331 \DeclareFontSeriesChangeRule {lc}{ex} {lex} {}
332 \DeclareFontSeriesChangeRule {lc}{ux} {lux} {}

333 \DeclareFontSeriesChangeRule {lsc}{ul} {ulsc} {}
334 \DeclareFontSeriesChangeRule {lsc}{el} {elsc} {}
335 \DeclareFontSeriesChangeRule {lsc}{l} {lsc} {}
336 \DeclareFontSeriesChangeRule {lsc}{sl} {slsc} {}
337 \DeclareFontSeriesChangeRule {lsc}{sb} {sbsc} {bsc}
338 \DeclareFontSeriesChangeRule {lsc}{b} {bsc} {}
339 \DeclareFontSeriesChangeRule {lsc}{eb} {ebsc} {bsc}
340 \DeclareFontSeriesChangeRule {lsc}{ub} {ubsc} {bsc}
341 \DeclareFontSeriesChangeRule {lsc}{uc} {luc} {}
342 \DeclareFontSeriesChangeRule {lsc}{ec} {lec} {}
343 \DeclareFontSeriesChangeRule {lsc}{c} {lc} {}
344 \DeclareFontSeriesChangeRule {lsc}{sc} {lsc} {}
345 \DeclareFontSeriesChangeRule {lsc}{sx} {lsx} {}
346 \DeclareFontSeriesChangeRule {lsc}{x} {lx} {}
347 \DeclareFontSeriesChangeRule {lsc}{ex} {lex} {}
348 \DeclareFontSeriesChangeRule {lsc}{ux} {lux} {}

349 \DeclareFontSeriesChangeRule {l}{uc} {luc} {}
350 \DeclareFontSeriesChangeRule {l}{ec} {lec} {}
351 \DeclareFontSeriesChangeRule {l}{c} {lc} {}
352 \DeclareFontSeriesChangeRule {l}{sc} {lsc} {}
353 \DeclareFontSeriesChangeRule {l}{sx} {lsx} {}
354 \DeclareFontSeriesChangeRule {l}{x} {lx} {}
355 \DeclareFontSeriesChangeRule {l}{ex} {lex} {}
356 \DeclareFontSeriesChangeRule {l}{ux} {lux} {}

```

```

357 \DeclareFontSeriesChangeRule {l}{sb}      {sb}    {b}
358 \DeclareFontSeriesChangeRule {l}{eb}      {eb}    {b}
359 \DeclareFontSeriesChangeRule {l}{ub}      {ub}    {b}

360 \DeclareFontSeriesChangeRule {lsx}{ul}     {ulsx}   {}
361 \DeclareFontSeriesChangeRule {lsx}{el}     {elsx}   {}
362 \DeclareFontSeriesChangeRule {lsx}{l}      {lsx}    {}
363 \DeclareFontSeriesChangeRule {lsx}{sl}     {slsx}   {}
364 \DeclareFontSeriesChangeRule {lsx}{sb}     {sbsx}   {bsx}
365 \DeclareFontSeriesChangeRule {lsx}{b}      {bsx}    {}
366 \DeclareFontSeriesChangeRule {lsx}{eb}     {ebsx}   {bsx}
367 \DeclareFontSeriesChangeRule {lsx}{ub}     {ubsx}   {bsx}
368 \DeclareFontSeriesChangeRule {lsx}{uc}     {luc}    {}
369 \DeclareFontSeriesChangeRule {lsx}{ec}     {lec}    {}
370 \DeclareFontSeriesChangeRule {lsx}{c}      {lc}     {}
371 \DeclareFontSeriesChangeRule {lsx}{sc}     {lsc}    {}
372 \DeclareFontSeriesChangeRule {lsx}{sx}     {lsx}    {}
373 \DeclareFontSeriesChangeRule {lsx}{x}      {lx}     {}
374 \DeclareFontSeriesChangeRule {lsx}{ex}     {lex}    {}
375 \DeclareFontSeriesChangeRule {lsx}{ux}     {lux}    {}

376 \DeclareFontSeriesChangeRule {lx}{ul}     {ulx}    {}
377 \DeclareFontSeriesChangeRule {lx}{el}     {elx}    {}
378 \DeclareFontSeriesChangeRule {lx}{l}      {lx}     {}
379 \DeclareFontSeriesChangeRule {lx}{sl}     {slx}    {}
380 \DeclareFontSeriesChangeRule {lx}{sb}     {sbx}    {bx}
381 \DeclareFontSeriesChangeRule {lx}{b}      {bx}     {}
382 \DeclareFontSeriesChangeRule {lx}{eb}     {ebx}    {bx}
383 \DeclareFontSeriesChangeRule {lx}{ub}     {ubx}    {bx}
384 \DeclareFontSeriesChangeRule {lx}{uc}     {luc}    {}
385 \DeclareFontSeriesChangeRule {lx}{ec}     {lec}    {}
386 \DeclareFontSeriesChangeRule {lx}{c}      {lc}     {}
387 \DeclareFontSeriesChangeRule {lx}{sc}     {lsc}    {}
388 \DeclareFontSeriesChangeRule {lx}{sx}     {lsx}    {}
389 \DeclareFontSeriesChangeRule {lx}{x}      {lx}     {}
390 \DeclareFontSeriesChangeRule {lx}{ex}     {lex}    {}
391 \DeclareFontSeriesChangeRule {lx}{ux}     {lux}    {}

392 \DeclareFontSeriesChangeRule {lex}{ul}     {ulex}   {}
393 \DeclareFontSeriesChangeRule {lex}{el}     {ele}    {}
394 \DeclareFontSeriesChangeRule {lex}{l}      {lex}    {}
395 \DeclareFontSeriesChangeRule {lex}{sl}     {slex}   {}
396 \DeclareFontSeriesChangeRule {lex}{sb}     {sbex}   {bex}
397 \DeclareFontSeriesChangeRule {lex}{b}      {bex}    {}
398 \DeclareFontSeriesChangeRule {lex}{eb}     {ebex}   {bex}
399 \DeclareFontSeriesChangeRule {lex}{ub}     {ubex}   {bex}
400 \DeclareFontSeriesChangeRule {lex}{uc}     {luc}    {}
401 \DeclareFontSeriesChangeRule {lex}{ec}     {lec}    {}
402 \DeclareFontSeriesChangeRule {lex}{c}      {lc}     {}
403 \DeclareFontSeriesChangeRule {lex}{sc}     {lsc}    {}
404 \DeclareFontSeriesChangeRule {lex}{sx}     {lsx}    {}
405 \DeclareFontSeriesChangeRule {lex}{x}      {lx}     {}
406 \DeclareFontSeriesChangeRule {lex}{ex}     {lex}    {}
407 \DeclareFontSeriesChangeRule {lex}{ux}     {lux}    {}

408 \DeclareFontSeriesChangeRule {lux}{ul}     {ulux}   {}
409 \DeclareFontSeriesChangeRule {lux}{el}     {elux}   {}

```

```

410 \DeclareFontSeriesChangeRule {lux}{l} {lux} {}
411 \DeclareFontSeriesChangeRule {lux}{sl} {slux} {}
412 \DeclareFontSeriesChangeRule {lux}{sb} {sbux} {bux}
413 \DeclareFontSeriesChangeRule {lux}{b} {bux} {}
414 \DeclareFontSeriesChangeRule {lux}{eb} {ebux} {bux}
415 \DeclareFontSeriesChangeRule {lux}{ub} {ubux} {bux}
416 \DeclareFontSeriesChangeRule {lux}{uc} {luc} {}
417 \DeclareFontSeriesChangeRule {lux}{ec} {lec} {}
418 \DeclareFontSeriesChangeRule {lux}{c} {lc} {}
419 \DeclareFontSeriesChangeRule {lux}{sc} {lsc} {}
420 \DeclareFontSeriesChangeRule {lux}{sx} {lsx} {}
421 \DeclareFontSeriesChangeRule {lux}{x} {lx} {}
422 \DeclareFontSeriesChangeRule {lux}{ex} {lex} {}
423 \DeclareFontSeriesChangeRule {lux}{ux} {lux} {}

424 \DeclareFontSeriesChangeRule {sluc}{ul} {uluc} {}
425 \DeclareFontSeriesChangeRule {sluc}{el} {eluc} {}
426 \DeclareFontSeriesChangeRule {sluc}{l} {luc} {}
427 \DeclareFontSeriesChangeRule {sluc}{sl} {sluc} {}
428 \DeclareFontSeriesChangeRule {sluc}{sb} {sbuc} {buc}
429 \DeclareFontSeriesChangeRule {sluc}{b} {buc} {}
430 \DeclareFontSeriesChangeRule {sluc}{eb} {ebuc} {buc}
431 \DeclareFontSeriesChangeRule {sluc}{ub} {ubuc} {buc}
432 \DeclareFontSeriesChangeRule {sluc}{uc} {sluc} {}
433 \DeclareFontSeriesChangeRule {sluc}{ec} {slec} {}
434 \DeclareFontSeriesChangeRule {sluc}{c} {slc} {}
435 \DeclareFontSeriesChangeRule {sluc}{sc} {slsc} {}
436 \DeclareFontSeriesChangeRule {sluc}{sx} {slsx} {}
437 \DeclareFontSeriesChangeRule {sluc}{x} {slx} {}
438 \DeclareFontSeriesChangeRule {sluc}{ex} {slex} {}
439 \DeclareFontSeriesChangeRule {sluc}{ux} {slux} {}

440 \DeclareFontSeriesChangeRule {slec}{ul} {ulec} {}
441 \DeclareFontSeriesChangeRule {slec}{el} {elec} {}
442 \DeclareFontSeriesChangeRule {slec}{l} {lec} {}
443 \DeclareFontSeriesChangeRule {slec}{sl} {slec} {}
444 \DeclareFontSeriesChangeRule {slec}{sb} {sbec} {bec}
445 \DeclareFontSeriesChangeRule {slec}{b} {bec} {}
446 \DeclareFontSeriesChangeRule {slec}{eb} {ebec} {bec}
447 \DeclareFontSeriesChangeRule {slec}{ub} {ubec} {bec}
448 \DeclareFontSeriesChangeRule {slec}{uc} {sluc} {}
449 \DeclareFontSeriesChangeRule {slec}{ec} {slec} {}
450 \DeclareFontSeriesChangeRule {slec}{c} {slc} {}
451 \DeclareFontSeriesChangeRule {slec}{sc} {slsc} {}
452 \DeclareFontSeriesChangeRule {slec}{sx} {slsx} {}
453 \DeclareFontSeriesChangeRule {slec}{x} {slx} {}
454 \DeclareFontSeriesChangeRule {slec}{ex} {slex} {}
455 \DeclareFontSeriesChangeRule {slec}{ux} {slux} {}

456 \DeclareFontSeriesChangeRule {slc}{ul} {ulc} {}
457 \DeclareFontSeriesChangeRule {slc}{el} {elc} {}
458 \DeclareFontSeriesChangeRule {slc}{l} {lc} {}
459 \DeclareFontSeriesChangeRule {slc}{sl} {slc} {}
460 \DeclareFontSeriesChangeRule {slc}{sb} {sbc} {bc}
461 \DeclareFontSeriesChangeRule {slc}{b} {bc} {}
462 \DeclareFontSeriesChangeRule {slc}{eb} {ebc} {bc}

```

```

463 \DeclareFontSeriesChangeRule {slc}{ub} {ubc} {bc}
464 \DeclareFontSeriesChangeRule {slc}{uc} {sluc} {}
465 \DeclareFontSeriesChangeRule {slc}{ec} {slec} {}
466 \DeclareFontSeriesChangeRule {slc}{c} {slc} {}
467 \DeclareFontSeriesChangeRule {slc}{sc} {slsc} {}
468 \DeclareFontSeriesChangeRule {slc}{sx} {slsx} {}
469 \DeclareFontSeriesChangeRule {slc}{x} {slx} {}
470 \DeclareFontSeriesChangeRule {slc}{ex} {slex} {}
471 \DeclareFontSeriesChangeRule {slc}{ux} {slux} {}

472 \DeclareFontSeriesChangeRule {slsc}{ul} {ulsc} {}
473 \DeclareFontSeriesChangeRule {slsc}{el} {elsc} {}
474 \DeclareFontSeriesChangeRule {slsc}{l} {lsc} {}
475 \DeclareFontSeriesChangeRule {slsc}{sl} {slsc} {}
476 \DeclareFontSeriesChangeRule {slsc}{sb} {sbsc} {bsc}
477 \DeclareFontSeriesChangeRule {slsc}{b} {bsc} {}
478 \DeclareFontSeriesChangeRule {slsc}{eb} {ebsc} {bsc}
479 \DeclareFontSeriesChangeRule {slsc}{ub} {ubsc} {bsc}
480 \DeclareFontSeriesChangeRule {slsc}{uc} {sluc} {}
481 \DeclareFontSeriesChangeRule {slsc}{ec} {slec} {}
482 \DeclareFontSeriesChangeRule {slsc}{c} {slc} {}
483 \DeclareFontSeriesChangeRule {slsc}{sc} {slsc} {}
484 \DeclareFontSeriesChangeRule {slsc}{sx} {slsx} {}
485 \DeclareFontSeriesChangeRule {slsc}{x} {slx} {}
486 \DeclareFontSeriesChangeRule {slsc}{ex} {slex} {}
487 \DeclareFontSeriesChangeRule {slsc}{ux} {slux} {}

488 \DeclareFontSeriesChangeRule {sl}{uc} {sluc} {}
489 \DeclareFontSeriesChangeRule {sl}{ec} {slec} {}
490 \DeclareFontSeriesChangeRule {sl}{c} {slc} {}
491 \DeclareFontSeriesChangeRule {sl}{sc} {slsc} {}
492 \DeclareFontSeriesChangeRule {sl}{sx} {slsx} {}
493 \DeclareFontSeriesChangeRule {sl}{x} {slx} {}
494 \DeclareFontSeriesChangeRule {sl}{ex} {slex} {}
495 \DeclareFontSeriesChangeRule {sl}{ux} {slux} {}
496 \DeclareFontSeriesChangeRule {sl}{sb} {sb} {b}
497 \DeclareFontSeriesChangeRule {sl}{eb} {eb} {b}
498 \DeclareFontSeriesChangeRule {sl}{ub} {ub} {b}

499 \DeclareFontSeriesChangeRule {slsx}{ul} {ulsx} {}
500 \DeclareFontSeriesChangeRule {slsx}{el} {elsx} {}
501 \DeclareFontSeriesChangeRule {slsx}{l} {lsx} {}
502 \DeclareFontSeriesChangeRule {slsx}{sl} {slsx} {}
503 \DeclareFontSeriesChangeRule {slsx}{sb} {sbsx} {bsx}
504 \DeclareFontSeriesChangeRule {slsx}{b} {bsx} {}
505 \DeclareFontSeriesChangeRule {slsx}{eb} {ebsx} {bsx}
506 \DeclareFontSeriesChangeRule {slsx}{ub} {ubsx} {bsx}
507 \DeclareFontSeriesChangeRule {slsx}{uc} {sluc} {}
508 \DeclareFontSeriesChangeRule {slsx}{ec} {slec} {}
509 \DeclareFontSeriesChangeRule {slsx}{c} {slc} {}
510 \DeclareFontSeriesChangeRule {slsx}{sc} {slsc} {}
511 \DeclareFontSeriesChangeRule {slsx}{sx} {slsx} {}
512 \DeclareFontSeriesChangeRule {slsx}{x} {slx} {}
513 \DeclareFontSeriesChangeRule {slsx}{ex} {slex} {}
514 \DeclareFontSeriesChangeRule {slsx}{ux} {slux} {}
515 \DeclareFontSeriesChangeRule {slx}{ul} {ulx} {}

```

```

516 \DeclareFontSeriesChangeRule {slx}{el} {elx} {}
517 \DeclareFontSeriesChangeRule {slx}{l} {lx} {}
518 \DeclareFontSeriesChangeRule {slx}{sl} {slx} {}
519 \DeclareFontSeriesChangeRule {slx}{sb} {sbx} {bx}
520 \DeclareFontSeriesChangeRule {slx}{b} {bx} {}
521 \DeclareFontSeriesChangeRule {slx}{eb} {ebx} {bx}
522 \DeclareFontSeriesChangeRule {slx}{ub} {ubx} {bx}
523 \DeclareFontSeriesChangeRule {slx}{uc} {sluc} {}
524 \DeclareFontSeriesChangeRule {slx}{ec} {slec} {}
525 \DeclareFontSeriesChangeRule {slx}{c} {slc} {}
526 \DeclareFontSeriesChangeRule {slx}{sc} {slsc} {}
527 \DeclareFontSeriesChangeRule {slx}{sx} {slsx} {}
528 \DeclareFontSeriesChangeRule {slx}{x} {slx} {}
529 \DeclareFontSeriesChangeRule {slx}{ex} {slex} {}
530 \DeclareFontSeriesChangeRule {slx}{ux} {slux} {}

531 \DeclareFontSeriesChangeRule {slex}{ul} {ulex} {}
532 \DeclareFontSeriesChangeRule {slex}{el} {ele} {}
533 \DeclareFontSeriesChangeRule {slex}{l} {lex} {}
534 \DeclareFontSeriesChangeRule {slex}{sl} {slex} {}
535 \DeclareFontSeriesChangeRule {slex}{sb} {sbex} {bex}
536 \DeclareFontSeriesChangeRule {slex}{b} {bex} {}
537 \DeclareFontSeriesChangeRule {slex}{eb} {ebex} {bex}
538 \DeclareFontSeriesChangeRule {slex}{ub} {ubex} {bex}
539 \DeclareFontSeriesChangeRule {slex}{uc} {sluc} {}
540 \DeclareFontSeriesChangeRule {slex}{ec} {slec} {}
541 \DeclareFontSeriesChangeRule {slex}{c} {slc} {}
542 \DeclareFontSeriesChangeRule {slex}{sc} {slsc} {}
543 \DeclareFontSeriesChangeRule {slex}{sx} {slsx} {}
544 \DeclareFontSeriesChangeRule {slex}{x} {slx} {}
545 \DeclareFontSeriesChangeRule {slex}{ex} {slex} {}
546 \DeclareFontSeriesChangeRule {slex}{ux} {slux} {}

547 \DeclareFontSeriesChangeRule {slux}{ul} {ulux} {}
548 \DeclareFontSeriesChangeRule {slux}{el} {elux} {}
549 \DeclareFontSeriesChangeRule {slux}{l} {lux} {}
550 \DeclareFontSeriesChangeRule {slux}{sl} {slux} {}
551 \DeclareFontSeriesChangeRule {slux}{sb} {sbux} {bux}
552 \DeclareFontSeriesChangeRule {slux}{b} {bux} {}
553 \DeclareFontSeriesChangeRule {slux}{eb} {ebux} {bux}
554 \DeclareFontSeriesChangeRule {slux}{ub} {ubux} {bux}
555 \DeclareFontSeriesChangeRule {slux}{uc} {sluc} {}
556 \DeclareFontSeriesChangeRule {slux}{ec} {slec} {}
557 \DeclareFontSeriesChangeRule {slux}{c} {slc} {}
558 \DeclareFontSeriesChangeRule {slux}{sc} {slsc} {}
559 \DeclareFontSeriesChangeRule {slux}{sx} {slsx} {}
560 \DeclareFontSeriesChangeRule {slux}{x} {slx} {}
561 \DeclareFontSeriesChangeRule {slux}{ex} {slex} {}
562 \DeclareFontSeriesChangeRule {slux}{ux} {slux} {}

563 \DeclareFontSeriesChangeRule {uc}{ul} {uluc} {}
564 \DeclareFontSeriesChangeRule {uc}{el} {eluc} {}
565 \DeclareFontSeriesChangeRule {uc}{l} {luc} {}
566 \DeclareFontSeriesChangeRule {uc}{sl} {sluc} {}
567 \DeclareFontSeriesChangeRule {uc}{sb} {sbuc} {buc}
568 \DeclareFontSeriesChangeRule {uc}{b} {buc} {}

```



```

569 \DeclareFontSeriesChangeRule {uc}{eb} {ebuc} {buc}
570 \DeclareFontSeriesChangeRule {uc}{ub} {ubuc} {buc}

571 \DeclareFontSeriesChangeRule {ec}{ul} {ulec} {}
572 \DeclareFontSeriesChangeRule {ec}{el} {elec} {}
573 \DeclareFontSeriesChangeRule {ec}{l} {lec} {}
574 \DeclareFontSeriesChangeRule {ec}{sl} {slec} {}
575 \DeclareFontSeriesChangeRule {ec}{sb} {sbec} {bec}
576 \DeclareFontSeriesChangeRule {ec}{b} {bec} {}
577 \DeclareFontSeriesChangeRule {ec}{eb} {ebec} {bec}
578 \DeclareFontSeriesChangeRule {ec}{ub} {ubec} {bec}

```

There are a number of font families that implement condensed series, but often only **c** and **bc**. Therefore, if we see a weight or width change request that can't be fulfilled we try to stay with **c** or **bc**.

```

579 \DeclareFontSeriesChangeRule {c}{ul} {ulc} {c}
580 \DeclareFontSeriesChangeRule {c}{el} {elc} {c}
581 \DeclareFontSeriesChangeRule {c}{l} {lc} {c}
582 \DeclareFontSeriesChangeRule {c}{sl} {slc} {c}
583 \DeclareFontSeriesChangeRule {c}{sb} {sbc} {bc}
584 \DeclareFontSeriesChangeRule {c}{b} {bc} {}
585 \DeclareFontSeriesChangeRule {c}{eb} {ebc} {bc}
586 \DeclareFontSeriesChangeRule {c}{ub} {ubc} {bc}
587 \DeclareFontSeriesChangeRule {c}{uc} {uc} {c}
588 \DeclareFontSeriesChangeRule {c}{ec} {ec} {c}
589 \DeclareFontSeriesChangeRule {c}{sc} {sc} {c}

590 \DeclareFontSeriesChangeRule {sc}{ul} {ulsc} {}
591 \DeclareFontSeriesChangeRule {sc}{el} {elsc} {}
592 \DeclareFontSeriesChangeRule {sc}{l} {lsc} {}
593 \DeclareFontSeriesChangeRule {sc}{sl} {slsc} {}
594 \DeclareFontSeriesChangeRule {sc}{sb} {bsc} {bsc}
595 \DeclareFontSeriesChangeRule {sc}{b} {bsc} {}
596 \DeclareFontSeriesChangeRule {sc}{eb} {ebsc} {bsc}
597 \DeclareFontSeriesChangeRule {sc}{ub} {ubsc} {bsc}

598 \DeclareFontSeriesChangeRule {m}{sb} {sb} {b}
599 \DeclareFontSeriesChangeRule {m}{eb} {eb} {b}
600 \DeclareFontSeriesChangeRule {m}{ub} {ub} {b}

```

This special rule normally does nothing since nearly every font implements the **b** bold series. The exception are Computer Modern and Latin Modern and fonts based on them. They usually only have **bx**, but then they normally provide an `.fd` file declaration mapping from **b** to **bx** and thus pretend that **b** exists. But in case any of them does not, we offer an alternative result and switch to **bx** if **b** can't be found.

```

601 \DeclareFontSeriesChangeRule {m}{b} {b} {bx}

602 \DeclareFontSeriesChangeRule {sx}{ul} {ulsx} {}
603 \DeclareFontSeriesChangeRule {sx}{el} {elsx} {}
604 \DeclareFontSeriesChangeRule {sx}{l} {lsx} {}
605 \DeclareFontSeriesChangeRule {sx}{sl} {slsx} {}
606 \DeclareFontSeriesChangeRule {sx}{sb} {bsx} {bsx}
607 \DeclareFontSeriesChangeRule {sx}{b} {bsx} {}
608 \DeclareFontSeriesChangeRule {sx}{eb} {ebsx} {bsx}
609 \DeclareFontSeriesChangeRule {sx}{ub} {ubsx} {bsx}

```

```

610 \DeclareFontSeriesChangeRule {x}{ul}    {ulx} {}
611 \DeclareFontSeriesChangeRule {x}{el}    {elx} {}
612 \DeclareFontSeriesChangeRule {x}{l}     {lx} {}
613 \DeclareFontSeriesChangeRule {x}{sl}    {slx} {}
614 \DeclareFontSeriesChangeRule {x}{sb}    {sbx} {bx}
615 \DeclareFontSeriesChangeRule {x}{b}     {bx} {}
616 \DeclareFontSeriesChangeRule {x}{eb}    {ebx} {bx}
617 \DeclareFontSeriesChangeRule {x}{ub}    {ubx} {bx}

618 \DeclareFontSeriesChangeRule {ex}{ul}   {ulex} {}
619 \DeclareFontSeriesChangeRule {ex}{el}   {elx} {}
620 \DeclareFontSeriesChangeRule {ex}{l}    {lex} {}
621 \DeclareFontSeriesChangeRule {ex}{sl}   {slex} {}
622 \DeclareFontSeriesChangeRule {ex}{sb}   {sbex} {bex}
623 \DeclareFontSeriesChangeRule {ex}{b}    {bex} {}
624 \DeclareFontSeriesChangeRule {ex}{eb}   {ebex} {bex}
625 \DeclareFontSeriesChangeRule {ex}{ub}   {ubex} {bex}

626 \DeclareFontSeriesChangeRule {ux}{ul}   {ulux} {}
627 \DeclareFontSeriesChangeRule {ux}{el}   {elux} {}
628 \DeclareFontSeriesChangeRule {ux}{l}    {lux} {}
629 \DeclareFontSeriesChangeRule {ux}{sl}   {slux} {}
630 \DeclareFontSeriesChangeRule {ux}{sb}   {sbux} {bux}
631 \DeclareFontSeriesChangeRule {ux}{b}    {bux} {}
632 \DeclareFontSeriesChangeRule {ux}{eb}   {ebux} {bux}
633 \DeclareFontSeriesChangeRule {ux}{ub}   {ubux} {bux}

634 \DeclareFontSeriesChangeRule {sbuc}{ul} {uluc} {}
635 \DeclareFontSeriesChangeRule {sbuc}{el} {eluc} {}
636 \DeclareFontSeriesChangeRule {sbuc}{l}  {luc} {}
637 \DeclareFontSeriesChangeRule {sbuc}{sl} {sluc} {}
638 \DeclareFontSeriesChangeRule {sbuc}{sb} {sbuc} {}
639 \DeclareFontSeriesChangeRule {sbuc}{b}  {buc} {}
640 \DeclareFontSeriesChangeRule {sbuc}{eb} {ebuc} {buc}
641 \DeclareFontSeriesChangeRule {sbuc}{ub} {ubuc} {buc}
642 \DeclareFontSeriesChangeRule {sbuc}{uc} {sbuc} {}
643 \DeclareFontSeriesChangeRule {sbuc}{ec} {sbec} {}
644 \DeclareFontSeriesChangeRule {sbuc}{c}  {sbc} {}
645 \DeclareFontSeriesChangeRule {sbuc}{sc} {sbsc} {}
646 \DeclareFontSeriesChangeRule {sbuc}{sx} {sbsx} {}
647 \DeclareFontSeriesChangeRule {sbuc}{x}  {sbx} {}
648 \DeclareFontSeriesChangeRule {sbuc}{ex} {sbex} {}
649 \DeclareFontSeriesChangeRule {sbuc}{ux} {sbux} {}

650 \DeclareFontSeriesChangeRule {sbec}{ul} {ulec} {}
651 \DeclareFontSeriesChangeRule {sbec}{el} {elec} {}
652 \DeclareFontSeriesChangeRule {sbec}{l}  {lec} {}
653 \DeclareFontSeriesChangeRule {sbec}{sl} {slec} {}
654 \DeclareFontSeriesChangeRule {sbec}{sb} {sbec} {}
655 \DeclareFontSeriesChangeRule {sbec}{b}  {bec} {}
656 \DeclareFontSeriesChangeRule {sbec}{eb} {ebec} {bec}
657 \DeclareFontSeriesChangeRule {sbec}{ub} {ubec} {bec}
658 \DeclareFontSeriesChangeRule {sbec}{uc} {sbuc} {}
659 \DeclareFontSeriesChangeRule {sbec}{ec} {sbec} {}
660 \DeclareFontSeriesChangeRule {sbec}{c}  {sbc} {}
661 \DeclareFontSeriesChangeRule {sbec}{sc} {sbsc} {}
662 \DeclareFontSeriesChangeRule {sbec}{sx} {sbsx} {}

```

```

663 \DeclareFontSeriesChangeRule {sbec}{x} {sbx} {}
664 \DeclareFontSeriesChangeRule {sbec}{ex} {sbex} {}
665 \DeclareFontSeriesChangeRule {sbec}{ux} {sbux} {}

666 \DeclareFontSeriesChangeRule {sbc}{ul} {ulc} {}
667 \DeclareFontSeriesChangeRule {sbc}{el} {elc} {}
668 \DeclareFontSeriesChangeRule {sbc}{l} {lc} {}
669 \DeclareFontSeriesChangeRule {sbc}{sl} {slc} {}
670 \DeclareFontSeriesChangeRule {sbc}{sb} {sbc} {}
671 \DeclareFontSeriesChangeRule {sbc}{b} {bc} {}
672 \DeclareFontSeriesChangeRule {sbc}{eb} {ebc} {bc}
673 \DeclareFontSeriesChangeRule {sbc}{ub} {ubc} {bc}
674 \DeclareFontSeriesChangeRule {sbc}{uc} {sbuc} {}
675 \DeclareFontSeriesChangeRule {sbc}{ec} {sbec} {}
676 \DeclareFontSeriesChangeRule {sbc}{c} {sbc} {}
677 \DeclareFontSeriesChangeRule {sbc}{sc} {sbsc} {}
678 \DeclareFontSeriesChangeRule {sbc}{sx} {sbsx} {}
679 \DeclareFontSeriesChangeRule {sbc}{x} {sbx} {}
680 \DeclareFontSeriesChangeRule {sbc}{ex} {sbex} {}
681 \DeclareFontSeriesChangeRule {sbc}{ux} {sbux} {}

682 \DeclareFontSeriesChangeRule {sbsc}{ul} {ulsc} {}
683 \DeclareFontSeriesChangeRule {sbsc}{el} {elsc} {}
684 \DeclareFontSeriesChangeRule {sbsc}{l} {lsc} {}
685 \DeclareFontSeriesChangeRule {sbsc}{sl} {slsc} {}
686 \DeclareFontSeriesChangeRule {sbsc}{sb} {sbsc} {}
687 \DeclareFontSeriesChangeRule {sbsc}{b} {bsc} {}
688 \DeclareFontSeriesChangeRule {sbsc}{eb} {ebsc} {bsc}
689 \DeclareFontSeriesChangeRule {sbsc}{ub} {ubsc} {bsc}
690 \DeclareFontSeriesChangeRule {sbsc}{uc} {sbuc} {}
691 \DeclareFontSeriesChangeRule {sbsc}{ec} {sbec} {}
692 \DeclareFontSeriesChangeRule {sbsc}{c} {sbc} {}
693 \DeclareFontSeriesChangeRule {sbsc}{sc} {sbsc} {}
694 \DeclareFontSeriesChangeRule {sbsc}{sx} {sbsx} {}
695 \DeclareFontSeriesChangeRule {sbsc}{x} {sbx} {}
696 \DeclareFontSeriesChangeRule {sbsc}{ex} {sbex} {}
697 \DeclareFontSeriesChangeRule {sbsc}{ux} {sbux} {}

698 \DeclareFontSeriesChangeRule {sb}{uc} {sbuc} {}
699 \DeclareFontSeriesChangeRule {sb}{ec} {sbec} {}
700 \DeclareFontSeriesChangeRule {sb}{c} {sbc} {}
701 \DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {}
702 \DeclareFontSeriesChangeRule {sb}{sx} {sbsx} {}
703 \DeclareFontSeriesChangeRule {sb}{x} {sbx} {}
704 \DeclareFontSeriesChangeRule {sb}{ex} {sbex} {}
705 \DeclareFontSeriesChangeRule {sb}{ux} {sbux} {}
706 \DeclareFontSeriesChangeRule {sb}{eb} {eb} {b}
707 \DeclareFontSeriesChangeRule {sb}{ub} {ub} {b}

708 \DeclareFontSeriesChangeRule {sbsx}{ul} {ulsx} {}
709 \DeclareFontSeriesChangeRule {sbsx}{el} {elsx} {}
710 \DeclareFontSeriesChangeRule {sbsx}{l} {lsx} {}
711 \DeclareFontSeriesChangeRule {sbsx}{sl} {slsx} {}
712 \DeclareFontSeriesChangeRule {sbsx}{sb} {sbsx} {}
713 \DeclareFontSeriesChangeRule {sbsx}{b} {bsx} {}
714 \DeclareFontSeriesChangeRule {sbsx}{eb} {ebsx} {bsx}
715 \DeclareFontSeriesChangeRule {sbsx}{ub} {ubsx} {bsx}

```

```

716 \DeclareFontSeriesChangeRule {sbsx}{uc} {sbuc} {}
717 \DeclareFontSeriesChangeRule {sbsx}{ec} {sbec} {}
718 \DeclareFontSeriesChangeRule {sbsx}{c} {sbc} {}
719 \DeclareFontSeriesChangeRule {sbsx}{sc} {sbsc} {}
720 \DeclareFontSeriesChangeRule {sbsx}{sx} {sbsx} {}
721 \DeclareFontSeriesChangeRule {sbsx}{x} {sbx} {}
722 \DeclareFontSeriesChangeRule {sbsx}{ex} {sbex} {}
723 \DeclareFontSeriesChangeRule {sbsx}{ux} {sbux} {}

724 \DeclareFontSeriesChangeRule {sbx}{ul} {ulx} {}
725 \DeclareFontSeriesChangeRule {sbx}{el} {elx} {}
726 \DeclareFontSeriesChangeRule {sbx}{l} {lx} {}
727 \DeclareFontSeriesChangeRule {sbx}{sl} {slx} {}
728 \DeclareFontSeriesChangeRule {sbx}{sb} {sbx} {}
729 \DeclareFontSeriesChangeRule {sbx}{b} {bx} {}
730 \DeclareFontSeriesChangeRule {sbx}{eb} {ebx} {bx}
731 \DeclareFontSeriesChangeRule {sbx}{ub} {ubx} {bx}
732 \DeclareFontSeriesChangeRule {sbx}{uc} {sbuc} {}
733 \DeclareFontSeriesChangeRule {sbx}{ec} {sbec} {}
734 \DeclareFontSeriesChangeRule {sbx}{c} {sbc} {}
735 \DeclareFontSeriesChangeRule {sbx}{sc} {sbsc} {}
736 \DeclareFontSeriesChangeRule {sbx}{sx} {sbsx} {}
737 \DeclareFontSeriesChangeRule {sbx}{x} {sbx} {}
738 \DeclareFontSeriesChangeRule {sbx}{ex} {sbex} {}
739 \DeclareFontSeriesChangeRule {sbx}{ux} {sbux} {}

740 \DeclareFontSeriesChangeRule {sbex}{ul} {ulex} {}
741 \DeclareFontSeriesChangeRule {sbex}{el} {elex} {}
742 \DeclareFontSeriesChangeRule {sbex}{l} {lex} {}
743 \DeclareFontSeriesChangeRule {sbex}{sl} {slex} {}
744 \DeclareFontSeriesChangeRule {sbex}{sb} {sbex} {}
745 \DeclareFontSeriesChangeRule {sbex}{b} {bex} {}
746 \DeclareFontSeriesChangeRule {sbex}{eb} {ebex} {bex}
747 \DeclareFontSeriesChangeRule {sbex}{ub} {ubex} {bex}
748 \DeclareFontSeriesChangeRule {sbex}{uc} {sbuc} {}
749 \DeclareFontSeriesChangeRule {sbex}{ec} {sbec} {}
750 \DeclareFontSeriesChangeRule {sbex}{c} {sbc} {}
751 \DeclareFontSeriesChangeRule {sbex}{sc} {sbsc} {}
752 \DeclareFontSeriesChangeRule {sbex}{sx} {sbsx} {}
753 \DeclareFontSeriesChangeRule {sbex}{x} {sbx} {}
754 \DeclareFontSeriesChangeRule {sbex}{ex} {sbex} {}
755 \DeclareFontSeriesChangeRule {sbex}{ux} {sbux} {}

756 \DeclareFontSeriesChangeRule {sbux}{ul} {ulux} {}
757 \DeclareFontSeriesChangeRule {sbux}{el} {elux} {}
758 \DeclareFontSeriesChangeRule {sbux}{l} {lux} {}
759 \DeclareFontSeriesChangeRule {sbux}{sl} {slux} {}
760 \DeclareFontSeriesChangeRule {sbux}{sb} {sbux} {}
761 \DeclareFontSeriesChangeRule {sbux}{b} {bux} {}
762 \DeclareFontSeriesChangeRule {sbux}{eb} {ebux} {bux}
763 \DeclareFontSeriesChangeRule {sbux}{ub} {ubux} {bux}
764 \DeclareFontSeriesChangeRule {sbux}{uc} {sbuc} {}
765 \DeclareFontSeriesChangeRule {sbux}{ec} {sbec} {}
766 \DeclareFontSeriesChangeRule {sbux}{c} {sbc} {}
767 \DeclareFontSeriesChangeRule {sbux}{sc} {sbsc} {}
768 \DeclareFontSeriesChangeRule {sbux}{sx} {sbsx} {}

```

```

769 \DeclareFontSeriesChangeRule {sbux}{x} {sbx} {}
770 \DeclareFontSeriesChangeRule {sbux}{ex} {sbex} {}
771 \DeclareFontSeriesChangeRule {sbux}{ux} {sbux} {}

772 \DeclareFontSeriesChangeRule {buc}{ul} {uluc} {}
773 \DeclareFontSeriesChangeRule {buc}{el} {eluc} {}
774 \DeclareFontSeriesChangeRule {buc}{l} {luc} {}
775 \DeclareFontSeriesChangeRule {buc}{sl} {sluc} {}
776 \DeclareFontSeriesChangeRule {buc}{sb} {sbuc} {buc}
777 \DeclareFontSeriesChangeRule {buc}{b} {buc} {}
778 \DeclareFontSeriesChangeRule {buc}{eb} {ebuc} {buc}
779 \DeclareFontSeriesChangeRule {buc}{ub} {ubuc} {buc}
780 \DeclareFontSeriesChangeRule {buc}{uc} {buc} {}
781 \DeclareFontSeriesChangeRule {buc}{ec} {bec} {}
782 \DeclareFontSeriesChangeRule {buc}{c} {bc} {}
783 \DeclareFontSeriesChangeRule {buc}{sc} {bsc} {}
784 \DeclareFontSeriesChangeRule {buc}{sx} {bsx} {}
785 \DeclareFontSeriesChangeRule {buc}{x} {bx} {}
786 \DeclareFontSeriesChangeRule {buc}{ex} {bex} {}
787 \DeclareFontSeriesChangeRule {buc}{ux} {bux} {}

788 \DeclareFontSeriesChangeRule {bec}{ul} {ulec} {}
789 \DeclareFontSeriesChangeRule {bec}{el} {elec} {}
790 \DeclareFontSeriesChangeRule {bec}{l} {lec} {}
791 \DeclareFontSeriesChangeRule {bec}{sl} {slec} {}
792 \DeclareFontSeriesChangeRule {bec}{sb} {sbec} {bec}
793 \DeclareFontSeriesChangeRule {bec}{b} {bec} {}
794 \DeclareFontSeriesChangeRule {bec}{eb} {ebec} {bec}
795 \DeclareFontSeriesChangeRule {bec}{ub} {ubec} {bec}
796 \DeclareFontSeriesChangeRule {bec}{uc} {buc} {}
797 \DeclareFontSeriesChangeRule {bec}{ec} {bec} {}
798 \DeclareFontSeriesChangeRule {bec}{c} {bc} {}
799 \DeclareFontSeriesChangeRule {bec}{sc} {bsc} {}
800 \DeclareFontSeriesChangeRule {bec}{sx} {bsx} {}
801 \DeclareFontSeriesChangeRule {bec}{x} {bx} {}
802 \DeclareFontSeriesChangeRule {bec}{ex} {bex} {}
803 \DeclareFontSeriesChangeRule {bec}{ux} {bux} {}

804 \DeclareFontSeriesChangeRule {bc}{ul} {ulc} {}
805 \DeclareFontSeriesChangeRule {bc}{el} {elc} {}
806 \DeclareFontSeriesChangeRule {bc}{l} {lc} {}
807 \DeclareFontSeriesChangeRule {bc}{sl} {slc} {}
808 \DeclareFontSeriesChangeRule {bc}{sb} {sbc} {bc}
809 \DeclareFontSeriesChangeRule {bc}{b} {bc} {}
810 \DeclareFontSeriesChangeRule {bc}{eb} {ebc} {bc}
811 \DeclareFontSeriesChangeRule {bc}{ub} {ubc} {bc}
812 \DeclareFontSeriesChangeRule {bc}{uc} {buc} {}
813 \DeclareFontSeriesChangeRule {bc}{ec} {bec} {}
814 \DeclareFontSeriesChangeRule {bc}{c} {bc} {}
815 \DeclareFontSeriesChangeRule {bc}{sc} {bsc} {}
816 \DeclareFontSeriesChangeRule {bc}{sx} {bsx} {}
817 \DeclareFontSeriesChangeRule {bc}{x} {bx} {}
818 \DeclareFontSeriesChangeRule {bc}{ex} {bex} {}
819 \DeclareFontSeriesChangeRule {bc}{ux} {bux} {}

820 \DeclareFontSeriesChangeRule {bsc}{ul} {ulsc} {}
821 \DeclareFontSeriesChangeRule {bsc}{el} {elsc} {}

```

```

822 \DeclareFontSeriesChangeRule {bsc}{l}   {lsc}   {}
823 \DeclareFontSeriesChangeRule {bsc}{sl}  {slsc}  {}
824 \DeclareFontSeriesChangeRule {bsc}{sb}  {sbsc}  {bsc}
825 \DeclareFontSeriesChangeRule {bsc}{b}   {bsc}   {}
826 \DeclareFontSeriesChangeRule {bsc}{eb}  {ebsc}  {bsc}
827 \DeclareFontSeriesChangeRule {bsc}{ub}  {ubsc}  {bsc}
828 \DeclareFontSeriesChangeRule {bsc}{uc}  {buc}   {}
829 \DeclareFontSeriesChangeRule {bsc}{ec}  {bec}   {}
830 \DeclareFontSeriesChangeRule {bsc}{c}   {bc}    {}
831 \DeclareFontSeriesChangeRule {bsc}{sc}  {bsc}   {}
832 \DeclareFontSeriesChangeRule {bsc}{sx}  {bsx}   {}
833 \DeclareFontSeriesChangeRule {bsc}{x}   {bx}    {}
834 \DeclareFontSeriesChangeRule {bsc}{ex}  {bex}   {}
835 \DeclareFontSeriesChangeRule {bsc}{ux}  {bux}   {}

```

If we are in **b** and a width change is requested that leads to a missing font face we stay in **b** because then the font family probably doesn't implement width changes at all.

```

836 \DeclareFontSeriesChangeRule {b}{uc}    {buc}   {b}
837 \DeclareFontSeriesChangeRule {b}{ec}    {bec}   {b}
838 \DeclareFontSeriesChangeRule {b}{c}     {bc}    {b}
839 \DeclareFontSeriesChangeRule {b}{sc}    {bsc}   {b}
840 \DeclareFontSeriesChangeRule {b}{sx}    {bsx}   {b}
841 \DeclareFontSeriesChangeRule {b}{x}     {bx}    {b}
842 \DeclareFontSeriesChangeRule {b}{ex}    {bex}   {b}
843 \DeclareFontSeriesChangeRule {b}{ux}    {bux}   {b}
844 \DeclareFontSeriesChangeRule {b}{sb}    {sb}    {b}
845 \DeclareFontSeriesChangeRule {b}{eb}    {eb}    {b}
846 \DeclareFontSeriesChangeRule {b}{ub}    {ub}    {b}

847 \DeclareFontSeriesChangeRule {bsx}{ul}  {ulsx}  {}
848 \DeclareFontSeriesChangeRule {bsx}{el}  {elsx}  {}
849 \DeclareFontSeriesChangeRule {bsx}{l}   {lsx}   {}
850 \DeclareFontSeriesChangeRule {bsx}{sl}  {slsx}  {}
851 \DeclareFontSeriesChangeRule {bsx}{sb}  {sbsx}  {bsx}
852 \DeclareFontSeriesChangeRule {bsx}{b}   {bsx}   {}
853 \DeclareFontSeriesChangeRule {bsx}{eb}  {ebsx}  {bsx}
854 \DeclareFontSeriesChangeRule {bsx}{ub}  {ubsx}  {bsx}
855 \DeclareFontSeriesChangeRule {bsx}{uc}  {buc}   {}
856 \DeclareFontSeriesChangeRule {bsx}{ec}  {bec}   {}
857 \DeclareFontSeriesChangeRule {bsx}{c}   {bc}    {}
858 \DeclareFontSeriesChangeRule {bsx}{sc}  {bsc}   {}
859 \DeclareFontSeriesChangeRule {bsx}{sx}  {bsx}   {}
860 \DeclareFontSeriesChangeRule {bsx}{x}   {bx}    {}
861 \DeclareFontSeriesChangeRule {bsx}{ex}  {bex}   {}
862 \DeclareFontSeriesChangeRule {bsx}{ux}  {bux}   {}

863 \DeclareFontSeriesChangeRule {bx}{ul}   {ulx}   {}
864 \DeclareFontSeriesChangeRule {bx}{el}   {elx}   {}
865 \DeclareFontSeriesChangeRule {bx}{l}    {lx}    {}
866 \DeclareFontSeriesChangeRule {bx}{sl}   {slx}   {}
867 \DeclareFontSeriesChangeRule {bx}{sb}   {sbx}   {bx}
868 \DeclareFontSeriesChangeRule {bx}{b}    {bx}    {}
869 \DeclareFontSeriesChangeRule {bx}{eb}   {ebx}   {bx}
870 \DeclareFontSeriesChangeRule {bx}{ub}   {ubx}   {bx}
871 \DeclareFontSeriesChangeRule {bx}{uc}   {buc}   {}

```

```

872 \DeclareFontSeriesChangeRule {bx}{ec} {bec} {}
873 \DeclareFontSeriesChangeRule {bx}{c} {bc} {}
874 \DeclareFontSeriesChangeRule {bx}{sc} {bsc} {}
875 \DeclareFontSeriesChangeRule {bx}{sx} {bsx} {}
876 \DeclareFontSeriesChangeRule {bx}{x} {bx} {}
877 \DeclareFontSeriesChangeRule {bx}{ex} {bex} {}
878 \DeclareFontSeriesChangeRule {bx}{ux} {bux} {}

879 \DeclareFontSeriesChangeRule {bex}{ul} {ulex} {}
880 \DeclareFontSeriesChangeRule {bex}{el} {ellex} {}
881 \DeclareFontSeriesChangeRule {bex}{l} {llex} {}
882 \DeclareFontSeriesChangeRule {bex}{sl} {sllex} {}
883 \DeclareFontSeriesChangeRule {bex}{sb} {sbex} {bex}
884 \DeclareFontSeriesChangeRule {bex}{b} {bex} {}
885 \DeclareFontSeriesChangeRule {bex}{eb} {ebex} {bex}
886 \DeclareFontSeriesChangeRule {bex}{ub} {ubex} {bex}
887 \DeclareFontSeriesChangeRule {bex}{uc} {buc} {}
888 \DeclareFontSeriesChangeRule {bex}{ec} {bec} {}
889 \DeclareFontSeriesChangeRule {bex}{c} {bc} {}
890 \DeclareFontSeriesChangeRule {bex}{sc} {bsc} {}
891 \DeclareFontSeriesChangeRule {bex}{sx} {bsx} {}
892 \DeclareFontSeriesChangeRule {bex}{x} {bx} {}
893 \DeclareFontSeriesChangeRule {bex}{ex} {bex} {}
894 \DeclareFontSeriesChangeRule {bex}{ux} {bux} {}

895 \DeclareFontSeriesChangeRule {bux}{ul} {ulux} {}
896 \DeclareFontSeriesChangeRule {bux}{el} {elux} {}
897 \DeclareFontSeriesChangeRule {bux}{l} {lux} {}
898 \DeclareFontSeriesChangeRule {bux}{sl} {slux} {}
899 \DeclareFontSeriesChangeRule {bux}{sb} {sbux} {bux}
900 \DeclareFontSeriesChangeRule {bux}{b} {bux} {}
901 \DeclareFontSeriesChangeRule {bux}{eb} {ebux} {bux}
902 \DeclareFontSeriesChangeRule {bux}{ub} {ubux} {bux}
903 \DeclareFontSeriesChangeRule {bux}{uc} {buc} {}
904 \DeclareFontSeriesChangeRule {bux}{ec} {bec} {}
905 \DeclareFontSeriesChangeRule {bux}{c} {bc} {}
906 \DeclareFontSeriesChangeRule {bux}{sc} {bsc} {}
907 \DeclareFontSeriesChangeRule {bux}{sx} {bsx} {}
908 \DeclareFontSeriesChangeRule {bux}{x} {bx} {}
909 \DeclareFontSeriesChangeRule {bux}{ex} {bex} {}
910 \DeclareFontSeriesChangeRule {bux}{ux} {bux} {}

911 \DeclareFontSeriesChangeRule {ebuc}{ul} {uluc} {}
912 \DeclareFontSeriesChangeRule {ebuc}{el} {eluc} {}
913 \DeclareFontSeriesChangeRule {ebuc}{l} {luc} {}
914 \DeclareFontSeriesChangeRule {ebuc}{sl} {sluc} {}
915 \DeclareFontSeriesChangeRule {ebuc}{sb} {sbuc} {buc}
916 \DeclareFontSeriesChangeRule {ebuc}{b} {buc} {}
917 \DeclareFontSeriesChangeRule {ebuc}{eb} {ebuc} {}

```

In the following rule, we use **eb** instead of **b** in the fourth argument, since **eb** is a better approximation to **ub** than **b** and **ebuc** is already in the first argument and therefore this font face probably actually exists. A similar consideration also applies to some other rules in the following.

```

918 \DeclareFontSeriesChangeRule {ebuc}{ub} {ubuc} {ebuc}
919 \DeclareFontSeriesChangeRule {ebuc}{uc} {ebuc} {}

```

```

920 \DeclareFontSeriesChangeRule {ebuc}{ec} {ebec} {}
921 \DeclareFontSeriesChangeRule {ebuc}{c} {ebc} {}
922 \DeclareFontSeriesChangeRule {ebuc}{sc} {ebsc} {}
923 \DeclareFontSeriesChangeRule {ebuc}{sx} {ebsx} {}
924 \DeclareFontSeriesChangeRule {ebuc}{x} {ebx} {}
925 \DeclareFontSeriesChangeRule {ebuc}{ex} {ebex} {}
926 \DeclareFontSeriesChangeRule {ebuc}{ux} {ebux} {}

927 \DeclareFontSeriesChangeRule {ebec}{ul} {ulec} {}
928 \DeclareFontSeriesChangeRule {ebec}{el} {elec} {}
929 \DeclareFontSeriesChangeRule {ebec}{l} {lec} {}
930 \DeclareFontSeriesChangeRule {ebec}{sl} {slec} {}
931 \DeclareFontSeriesChangeRule {ebec}{sb} {sbec} {bec}
932 \DeclareFontSeriesChangeRule {ebec}{b} {bec} {}
933 \DeclareFontSeriesChangeRule {ebec}{eb} {ebec} {}
934 \DeclareFontSeriesChangeRule {ebec}{ub} {ubec} {ebec}
935 \DeclareFontSeriesChangeRule {ebec}{uc} {ebuc} {}
936 \DeclareFontSeriesChangeRule {ebec}{ec} {ebec} {}
937 \DeclareFontSeriesChangeRule {ebec}{c} {ebc} {}
938 \DeclareFontSeriesChangeRule {ebec}{sc} {ebsc} {}
939 \DeclareFontSeriesChangeRule {ebec}{sx} {ebsx} {}
940 \DeclareFontSeriesChangeRule {ebec}{x} {ebx} {}
941 \DeclareFontSeriesChangeRule {ebec}{ex} {ebex} {}
942 \DeclareFontSeriesChangeRule {ebec}{ux} {ebux} {}

943 \DeclareFontSeriesChangeRule {ebc}{ul} {ulc} {}
944 \DeclareFontSeriesChangeRule {ebc}{el} {elc} {}
945 \DeclareFontSeriesChangeRule {ebc}{l} {lc} {}
946 \DeclareFontSeriesChangeRule {ebc}{sl} {slc} {}
947 \DeclareFontSeriesChangeRule {ebc}{sb} {sbc} {bc}
948 \DeclareFontSeriesChangeRule {ebc}{b} {bc} {}
949 \DeclareFontSeriesChangeRule {ebc}{eb} {ebc} {}
950 \DeclareFontSeriesChangeRule {ebc}{ub} {ubc} {ebc}
951 \DeclareFontSeriesChangeRule {ebc}{uc} {ebuc} {}
952 \DeclareFontSeriesChangeRule {ebc}{ec} {ebec} {}
953 \DeclareFontSeriesChangeRule {ebc}{c} {ebc} {}
954 \DeclareFontSeriesChangeRule {ebc}{sc} {ebsc} {}
955 \DeclareFontSeriesChangeRule {ebc}{sx} {ebsx} {}
956 \DeclareFontSeriesChangeRule {ebc}{x} {ebx} {}
957 \DeclareFontSeriesChangeRule {ebc}{ex} {ebex} {}
958 \DeclareFontSeriesChangeRule {ebc}{ux} {ebux} {}

959 \DeclareFontSeriesChangeRule {ebsc}{ul} {ulsc} {}
960 \DeclareFontSeriesChangeRule {ebsc}{el} {elsc} {}
961 \DeclareFontSeriesChangeRule {ebsc}{l} {lsc} {}
962 \DeclareFontSeriesChangeRule {ebsc}{sl} {slsc} {}
963 \DeclareFontSeriesChangeRule {ebsc}{sb} {sbsc} {bsc}
964 \DeclareFontSeriesChangeRule {ebsc}{b} {bsc} {}
965 \DeclareFontSeriesChangeRule {ebsc}{eb} {ebsc} {}
966 \DeclareFontSeriesChangeRule {ebsc}{ub} {ubsc} {ebsc}
967 \DeclareFontSeriesChangeRule {ebsc}{uc} {ebuc} {}
968 \DeclareFontSeriesChangeRule {ebsc}{ec} {ebec} {}
969 \DeclareFontSeriesChangeRule {ebsc}{c} {ebc} {}
970 \DeclareFontSeriesChangeRule {ebsc}{sc} {ebsc} {}
971 \DeclareFontSeriesChangeRule {ebsc}{sx} {ebsx} {}
972 \DeclareFontSeriesChangeRule {ebsc}{x} {ebx} {}

```



```

973 \DeclareFontSeriesChangeRule {ebsc}{ex} {ebex} {}
974 \DeclareFontSeriesChangeRule {ebsc}{ux} {ebux} {}

975 \DeclareFontSeriesChangeRule {eb}{uc} {ebuc} {}
976 \DeclareFontSeriesChangeRule {eb}{ec} {ebec} {}
977 \DeclareFontSeriesChangeRule {eb}{c} {ebc} {}
978 \DeclareFontSeriesChangeRule {eb}{sc} {ebsc} {}
979 \DeclareFontSeriesChangeRule {eb}{sx} {ebsx} {}
980 \DeclareFontSeriesChangeRule {eb}{x} {ebx} {}
981 \DeclareFontSeriesChangeRule {eb}{ex} {ebex} {}
982 \DeclareFontSeriesChangeRule {eb}{ux} {ebux} {}
983 \DeclareFontSeriesChangeRule {eb}{sb} {sb} {b}
984 \DeclareFontSeriesChangeRule {eb}{ub} {ub} {eb}

985 \DeclareFontSeriesChangeRule {ebsx}{ul} {ulsx} {}
986 \DeclareFontSeriesChangeRule {ebsx}{el} {elsx} {}
987 \DeclareFontSeriesChangeRule {ebsx}{l} {lsx} {}
988 \DeclareFontSeriesChangeRule {ebsx}{sl} {slsx} {}
989 \DeclareFontSeriesChangeRule {ebsx}{sb} {sbsx} {bsx}
990 \DeclareFontSeriesChangeRule {ebsx}{b} {bsx} {}
991 \DeclareFontSeriesChangeRule {ebsx}{eb} {ebsx} {}
992 \DeclareFontSeriesChangeRule {ebsx}{ub} {ubsx} {ebsx}
993 \DeclareFontSeriesChangeRule {ebsx}{uc} {ebuc} {}
994 \DeclareFontSeriesChangeRule {ebsx}{ec} {ebec} {}
995 \DeclareFontSeriesChangeRule {ebsx}{c} {ebc} {}
996 \DeclareFontSeriesChangeRule {ebsx}{sc} {ebsc} {}
997 \DeclareFontSeriesChangeRule {ebsx}{sx} {ebsx} {}
998 \DeclareFontSeriesChangeRule {ebsx}{x} {ebx} {}
999 \DeclareFontSeriesChangeRule {ebsx}{ex} {ebex} {}
1000 \DeclareFontSeriesChangeRule {ebsx}{ux} {ebux} {}

1001 \DeclareFontSeriesChangeRule {ebx}{ul} {ulx} {}
1002 \DeclareFontSeriesChangeRule {ebx}{el} {elx} {}
1003 \DeclareFontSeriesChangeRule {ebx}{l} {lx} {}
1004 \DeclareFontSeriesChangeRule {ebx}{sl} {slx} {}
1005 \DeclareFontSeriesChangeRule {ebx}{sb} {sbx} {bx}
1006 \DeclareFontSeriesChangeRule {ebx}{b} {bx} {}
1007 \DeclareFontSeriesChangeRule {ebx}{eb} {ebx} {}
1008 \DeclareFontSeriesChangeRule {ebx}{ub} {ubx} {ebx}
1009 \DeclareFontSeriesChangeRule {ebx}{uc} {ebuc} {}
1010 \DeclareFontSeriesChangeRule {ebx}{ec} {ebec} {}
1011 \DeclareFontSeriesChangeRule {ebx}{c} {ebc} {}
1012 \DeclareFontSeriesChangeRule {ebx}{sc} {ebsc} {}
1013 \DeclareFontSeriesChangeRule {ebx}{sx} {ebsx} {}
1014 \DeclareFontSeriesChangeRule {ebx}{x} {ebx} {}
1015 \DeclareFontSeriesChangeRule {ebx}{ex} {ebex} {}
1016 \DeclareFontSeriesChangeRule {ebx}{ux} {ebux} {}

1017 \DeclareFontSeriesChangeRule {ebex}{ul} {ulex} {}
1018 \DeclareFontSeriesChangeRule {ebex}{el} {ele x} {}
1019 \DeclareFontSeriesChangeRule {ebex}{l} {lex} {}
1020 \DeclareFontSeriesChangeRule {ebex}{sl} {slex} {}
1021 \DeclareFontSeriesChangeRule {ebex}{sb} {sbox} {box}
1022 \DeclareFontSeriesChangeRule {ebex}{b} {box} {}
1023 \DeclareFontSeriesChangeRule {ebex}{eb} {ebex} {}
1024 \DeclareFontSeriesChangeRule {ebex}{ub} {ubex} {ebex}
1025 \DeclareFontSeriesChangeRule {ebex}{uc} {ebuc} {}

```

```

1026 \DeclareFontSeriesChangeRule {ebex}{ec} {ebec} {}
1027 \DeclareFontSeriesChangeRule {ebex}{c} {ebc} {}
1028 \DeclareFontSeriesChangeRule {ebex}{sc} {ebsc} {}
1029 \DeclareFontSeriesChangeRule {ebex}{sx} {ebsx} {}
1030 \DeclareFontSeriesChangeRule {ebex}{x} {ebx} {}
1031 \DeclareFontSeriesChangeRule {ebex}{ex} {ebex} {}
1032 \DeclareFontSeriesChangeRule {ebex}{ux} {ebux} {}

1033 \DeclareFontSeriesChangeRule {ebux}{ul} {ulux} {}
1034 \DeclareFontSeriesChangeRule {ebux}{el} {elux} {}
1035 \DeclareFontSeriesChangeRule {ebux}{l} {lux} {}
1036 \DeclareFontSeriesChangeRule {ebux}{sl} {slux} {}
1037 \DeclareFontSeriesChangeRule {ebux}{sb} {sbux} {bux}
1038 \DeclareFontSeriesChangeRule {ebux}{b} {bux} {}
1039 \DeclareFontSeriesChangeRule {ebux}{eb} {ebux} {}
1040 \DeclareFontSeriesChangeRule {ebux}{ub} {ubux} {ebux}
1041 \DeclareFontSeriesChangeRule {ebux}{uc} {ebuc} {}
1042 \DeclareFontSeriesChangeRule {ebux}{ec} {ebec} {}
1043 \DeclareFontSeriesChangeRule {ebux}{c} {ebc} {}
1044 \DeclareFontSeriesChangeRule {ebux}{sc} {ebsc} {}
1045 \DeclareFontSeriesChangeRule {ebux}{sx} {ebsx} {}
1046 \DeclareFontSeriesChangeRule {ebux}{x} {ebx} {}
1047 \DeclareFontSeriesChangeRule {ebux}{ex} {ebex} {}
1048 \DeclareFontSeriesChangeRule {ebux}{ux} {ebux} {}

1049 \DeclareFontSeriesChangeRule {ubuc}{ul} {uluc} {}
1050 \DeclareFontSeriesChangeRule {ubuc}{el} {eluc} {}
1051 \DeclareFontSeriesChangeRule {ubuc}{l} {luc} {}
1052 \DeclareFontSeriesChangeRule {ubuc}{sl} {sluc} {}
1053 \DeclareFontSeriesChangeRule {ubuc}{sb} {sbuc} {buc}
1054 \DeclareFontSeriesChangeRule {ubuc}{b} {buc} {}
1055 \DeclareFontSeriesChangeRule {ubuc}{eb} {ebuc} {ubuc}
1056 \DeclareFontSeriesChangeRule {ubuc}{ub} {ubuc} {}
1057 \DeclareFontSeriesChangeRule {ubuc}{uc} {ubuc} {}
1058 \DeclareFontSeriesChangeRule {ubuc}{ec} {ubec} {}
1059 \DeclareFontSeriesChangeRule {ubuc}{c} {ubc} {}
1060 \DeclareFontSeriesChangeRule {ubuc}{sc} {ubsc} {}
1061 \DeclareFontSeriesChangeRule {ubuc}{sx} {ubsx} {}
1062 \DeclareFontSeriesChangeRule {ubuc}{x} {ubx} {}
1063 \DeclareFontSeriesChangeRule {ubuc}{ex} {ubex} {}
1064 \DeclareFontSeriesChangeRule {ubuc}{ux} {ubux} {}

1065 \DeclareFontSeriesChangeRule {ubec}{ul} {ulec} {}
1066 \DeclareFontSeriesChangeRule {ubec}{el} {elec} {}
1067 \DeclareFontSeriesChangeRule {ubec}{l} {lec} {}
1068 \DeclareFontSeriesChangeRule {ubec}{sl} {slec} {}
1069 \DeclareFontSeriesChangeRule {ubec}{sb} {sbec} {bec}
1070 \DeclareFontSeriesChangeRule {ubec}{b} {bec} {}
1071 \DeclareFontSeriesChangeRule {ubec}{eb} {ebec} {ubec}
1072 \DeclareFontSeriesChangeRule {ubec}{ub} {ubec} {}
1073 \DeclareFontSeriesChangeRule {ubec}{uc} {ubuc} {}
1074 \DeclareFontSeriesChangeRule {ubec}{ec} {ubec} {}
1075 \DeclareFontSeriesChangeRule {ubec}{c} {ubc} {}
1076 \DeclareFontSeriesChangeRule {ubec}{sc} {ubsc} {}
1077 \DeclareFontSeriesChangeRule {ubec}{sx} {ubsx} {}
1078 \DeclareFontSeriesChangeRule {ubec}{x} {ubx} {}

```

```

1079 \DeclareFontSeriesChangeRule {ubec}{ex} {ubex} {}
1080 \DeclareFontSeriesChangeRule {ubec}{ux} {ubux} {}

1081 \DeclareFontSeriesChangeRule {ubc}{ul} {ulc} {}
1082 \DeclareFontSeriesChangeRule {ubc}{el} {elc} {}
1083 \DeclareFontSeriesChangeRule {ubc}{l} {lc} {}
1084 \DeclareFontSeriesChangeRule {ubc}{sl} {slc} {}
1085 \DeclareFontSeriesChangeRule {ubc}{sb} {sbc} {bc}
1086 \DeclareFontSeriesChangeRule {ubc}{b} {bc} {}
1087 \DeclareFontSeriesChangeRule {ubc}{eb} {ebc} {ubc}
1088 \DeclareFontSeriesChangeRule {ubc}{ub} {ubc} {}
1089 \DeclareFontSeriesChangeRule {ubc}{uc} {ubuc} {}
1090 \DeclareFontSeriesChangeRule {ubc}{ec} {ubec} {}
1091 \DeclareFontSeriesChangeRule {ubc}{c} {ubc} {}
1092 \DeclareFontSeriesChangeRule {ubc}{sc} {ubsc} {}
1093 \DeclareFontSeriesChangeRule {ubc}{sx} {ubsx} {}
1094 \DeclareFontSeriesChangeRule {ubc}{x} {ubx} {}
1095 \DeclareFontSeriesChangeRule {ubc}{ex} {ubex} {}
1096 \DeclareFontSeriesChangeRule {ubc}{ux} {ubux} {}

1097 \DeclareFontSeriesChangeRule {ubsc}{ul} {ulsc} {}
1098 \DeclareFontSeriesChangeRule {ubsc}{el} {elsc} {}
1099 \DeclareFontSeriesChangeRule {ubsc}{l} {lsc} {}
1100 \DeclareFontSeriesChangeRule {ubsc}{sl} {slsc} {}
1101 \DeclareFontSeriesChangeRule {ubsc}{sb} {sbsc} {bsc}
1102 \DeclareFontSeriesChangeRule {ubsc}{b} {bsc} {}
1103 \DeclareFontSeriesChangeRule {ubsc}{eb} {ebsc} {ubsc}
1104 \DeclareFontSeriesChangeRule {ubsc}{ub} {ubsc} {}
1105 \DeclareFontSeriesChangeRule {ubsc}{uc} {ubuc} {}
1106 \DeclareFontSeriesChangeRule {ubsc}{ec} {ubec} {}
1107 \DeclareFontSeriesChangeRule {ubsc}{c} {ubc} {}
1108 \DeclareFontSeriesChangeRule {ubsc}{sc} {ubsc} {}
1109 \DeclareFontSeriesChangeRule {ubsc}{sx} {ubsx} {}
1110 \DeclareFontSeriesChangeRule {ubsc}{x} {ubx} {}
1111 \DeclareFontSeriesChangeRule {ubsc}{ex} {ubex} {}
1112 \DeclareFontSeriesChangeRule {ubsc}{ux} {ubux} {}

1113 \DeclareFontSeriesChangeRule {ub}{uc} {ubuc} {}
1114 \DeclareFontSeriesChangeRule {ub}{ec} {ubec} {}
1115 \DeclareFontSeriesChangeRule {ub}{c} {ubc} {}
1116 \DeclareFontSeriesChangeRule {ub}{sc} {ubsc} {}
1117 \DeclareFontSeriesChangeRule {ub}{sx} {ubsx} {}
1118 \DeclareFontSeriesChangeRule {ub}{x} {ubx} {}
1119 \DeclareFontSeriesChangeRule {ub}{ex} {ubex} {}
1120 \DeclareFontSeriesChangeRule {ub}{ux} {ubux} {}
1121 \DeclareFontSeriesChangeRule {ub}{sb} {sb} {b}
1122 \DeclareFontSeriesChangeRule {ub}{eb} {eb} {ub}

1123 \DeclareFontSeriesChangeRule {ubsx}{ul} {ulsx} {}
1124 \DeclareFontSeriesChangeRule {ubsx}{el} {elsx} {}
1125 \DeclareFontSeriesChangeRule {ubsx}{l} {lsx} {}
1126 \DeclareFontSeriesChangeRule {ubsx}{sl} {slsx} {}
1127 \DeclareFontSeriesChangeRule {ubsx}{sb} {sbsx} {bsx}
1128 \DeclareFontSeriesChangeRule {ubsx}{b} {bsx} {}
1129 \DeclareFontSeriesChangeRule {ubsx}{eb} {ebsx} {ubsx}
1130 \DeclareFontSeriesChangeRule {ubsx}{ub} {ubsx} {}
1131 \DeclareFontSeriesChangeRule {ubsx}{uc} {ubuc} {}

```

```

1132 \DeclareFontSeriesChangeRule {ubsx}{ec} {ubec} {}
1133 \DeclareFontSeriesChangeRule {ubsx}{c} {ubc} {}
1134 \DeclareFontSeriesChangeRule {ubsx}{sc} {ubsc} {}
1135 \DeclareFontSeriesChangeRule {ubsx}{sx} {ubsx} {}
1136 \DeclareFontSeriesChangeRule {ubsx}{x} {ubx} {}
1137 \DeclareFontSeriesChangeRule {ubsx}{ex} {ubex} {}
1138 \DeclareFontSeriesChangeRule {ubsx}{ux} {ubux} {}

1139 \DeclareFontSeriesChangeRule {ubx}{ul} {ulx} {}
1140 \DeclareFontSeriesChangeRule {ubx}{el} {elx} {}
1141 \DeclareFontSeriesChangeRule {ubx}{l} {lx} {}
1142 \DeclareFontSeriesChangeRule {ubx}{sl} {slx} {}
1143 \DeclareFontSeriesChangeRule {ubx}{sb} {sbx} {bx}
1144 \DeclareFontSeriesChangeRule {ubx}{b} {bx} {}
1145 \DeclareFontSeriesChangeRule {ubx}{eb} {ebx} {ubx}
1146 \DeclareFontSeriesChangeRule {ubx}{ub} {ubx} {}
1147 \DeclareFontSeriesChangeRule {ubx}{uc} {ubuc} {}
1148 \DeclareFontSeriesChangeRule {ubx}{ec} {ubec} {}
1149 \DeclareFontSeriesChangeRule {ubx}{c} {ubc} {}
1150 \DeclareFontSeriesChangeRule {ubx}{sc} {ubsc} {}
1151 \DeclareFontSeriesChangeRule {ubx}{sx} {ubsx} {}
1152 \DeclareFontSeriesChangeRule {ubx}{x} {ubx} {}
1153 \DeclareFontSeriesChangeRule {ubx}{ex} {ubex} {}
1154 \DeclareFontSeriesChangeRule {ubx}{ux} {ubux} {}

1155 \DeclareFontSeriesChangeRule {ubex}{ul} {ulex} {}
1156 \DeclareFontSeriesChangeRule {ubex}{el} {elx} {}
1157 \DeclareFontSeriesChangeRule {ubex}{l} {lex} {}
1158 \DeclareFontSeriesChangeRule {ubex}{sl} {slex} {}
1159 \DeclareFontSeriesChangeRule {ubex}{sb} {sbex} {bex}
1160 \DeclareFontSeriesChangeRule {ubex}{b} {bex} {}
1161 \DeclareFontSeriesChangeRule {ubex}{eb} {ebex} {ubex}
1162 \DeclareFontSeriesChangeRule {ubex}{ub} {ubex} {}
1163 \DeclareFontSeriesChangeRule {ubex}{uc} {ubuc} {}
1164 \DeclareFontSeriesChangeRule {ubex}{ec} {ubec} {}
1165 \DeclareFontSeriesChangeRule {ubex}{c} {ubc} {}
1166 \DeclareFontSeriesChangeRule {ubex}{sc} {ubsc} {}
1167 \DeclareFontSeriesChangeRule {ubex}{sx} {ubsx} {}
1168 \DeclareFontSeriesChangeRule {ubex}{x} {ubx} {}
1169 \DeclareFontSeriesChangeRule {ubex}{ex} {ubex} {}
1170 \DeclareFontSeriesChangeRule {ubex}{ux} {ubux} {}

1171 \DeclareFontSeriesChangeRule {ubux}{ul} {ulux} {}
1172 \DeclareFontSeriesChangeRule {ubux}{el} {elux} {}
1173 \DeclareFontSeriesChangeRule {ubux}{l} {lux} {}
1174 \DeclareFontSeriesChangeRule {ubux}{sl} {slux} {}
1175 \DeclareFontSeriesChangeRule {ubux}{sb} {sbux} {bux}
1176 \DeclareFontSeriesChangeRule {ubux}{b} {bux} {}
1177 \DeclareFontSeriesChangeRule {ubux}{eb} {ebux} {ubux}
1178 \DeclareFontSeriesChangeRule {ubux}{ub} {ubux} {}
1179 \DeclareFontSeriesChangeRule {ubux}{uc} {ubuc} {}
1180 \DeclareFontSeriesChangeRule {ubux}{ec} {ubec} {}
1181 \DeclareFontSeriesChangeRule {ubux}{c} {ubc} {}
1182 \DeclareFontSeriesChangeRule {ubux}{sc} {ubsc} {}
1183 \DeclareFontSeriesChangeRule {ubux}{sx} {ubsx} {}
1184 \DeclareFontSeriesChangeRule {ubux}{x} {ubx} {}

```

```

1185 \DeclareFontSeriesChangeRule {ubux}{ex} {ubex} {}
1186 \DeclareFontSeriesChangeRule {ubux}{ux} {ubux} {}

```

Special rules for `lm` etc. aren't needed because if `lm` is requested, it will be used if there is no rule, and that is then reduced to `l` automatically. Same for `mc` and friends.

The following entries handle a request for `bx` and fall back to `b` if that can't be fulfilled.

```

1187 \DeclareFontSeriesChangeRule {uluc}{bx} {bx} {b}
1188 \DeclareFontSeriesChangeRule {ulec}{bx} {bx} {b}
1189 \DeclareFontSeriesChangeRule {ulc}{bx} {bx} {b}
1190 \DeclareFontSeriesChangeRule {ulsc}{bx} {bx} {b}
1191 \DeclareFontSeriesChangeRule {ul}{bx} {bx} {b}
1192 \DeclareFontSeriesChangeRule {ulsx}{bx} {bx} {b}
1193 \DeclareFontSeriesChangeRule {ulx}{bx} {bx} {b}
1194 \DeclareFontSeriesChangeRule {ulex}{bx} {bx} {b}
1195 \DeclareFontSeriesChangeRule {ulux}{bx} {bx} {b}

1196 \DeclareFontSeriesChangeRule {eluc}{bx} {bx} {b}
1197 \DeclareFontSeriesChangeRule {elec}{bx} {bx} {b}
1198 \DeclareFontSeriesChangeRule {elc}{bx} {bx} {b}
1199 \DeclareFontSeriesChangeRule {elsc}{bx} {bx} {b}
1200 \DeclareFontSeriesChangeRule {el}{bx} {bx} {b}
1201 \DeclareFontSeriesChangeRule {elsx}{bx} {bx} {b}
1202 \DeclareFontSeriesChangeRule {elx}{bx} {bx} {b}
1203 \DeclareFontSeriesChangeRule {ellex}{bx} {bx} {b}
1204 \DeclareFontSeriesChangeRule {elux}{bx} {bx} {b}

1205 \DeclareFontSeriesChangeRule {luc}{bx} {bx} {b}
1206 \DeclareFontSeriesChangeRule {lec}{bx} {bx} {b}
1207 \DeclareFontSeriesChangeRule {lc}{bx} {bx} {b}
1208 \DeclareFontSeriesChangeRule {lsc}{bx} {bx} {b}
1209 \DeclareFontSeriesChangeRule {l}{bx} {bx} {b}
1210 \DeclareFontSeriesChangeRule {lsx}{bx} {bx} {b}
1211 \DeclareFontSeriesChangeRule {lx}{bx} {bx} {b}
1212 \DeclareFontSeriesChangeRule {lex}{bx} {bx} {b}
1213 \DeclareFontSeriesChangeRule {lux}{bx} {bx} {b}

1214 \DeclareFontSeriesChangeRule {sluc}{bx} {bx} {b}
1215 \DeclareFontSeriesChangeRule {slec}{bx} {bx} {b}
1216 \DeclareFontSeriesChangeRule {slc}{bx} {bx} {b}
1217 \DeclareFontSeriesChangeRule {slsc}{bx} {bx} {b}
1218 \DeclareFontSeriesChangeRule {sl}{bx} {bx} {b}
1219 \DeclareFontSeriesChangeRule {slsx}{bx} {bx} {b}
1220 \DeclareFontSeriesChangeRule {slx}{bx} {bx} {b}
1221 \DeclareFontSeriesChangeRule {slex}{bx} {bx} {b}
1222 \DeclareFontSeriesChangeRule {slux}{bx} {bx} {b}

1223 \DeclareFontSeriesChangeRule {uc}{bx} {bx} {b}
1224 \DeclareFontSeriesChangeRule {ec}{bx} {bx} {b}
1225 \DeclareFontSeriesChangeRule {c}{bx} {bx} {b}
1226 \DeclareFontSeriesChangeRule {sc}{bx} {bx} {b}
1227 \DeclareFontSeriesChangeRule {m}{bx} {bx} {b}
1228 \DeclareFontSeriesChangeRule {sx}{bx} {bx} {b}
1229 \DeclareFontSeriesChangeRule {x}{bx} {bx} {b}
1230 \DeclareFontSeriesChangeRule {ex}{bx} {bx} {b}
1231 \DeclareFontSeriesChangeRule {ux}{bx} {bx} {b}

```

```

1232 \DeclareFontSeriesChangeRule {sbuc}{bx} {bx} {b}
1233 \DeclareFontSeriesChangeRule {sbec}{bx} {bx} {b}
1234 \DeclareFontSeriesChangeRule {sbc}{bx} {bx} {b}
1235 \DeclareFontSeriesChangeRule {sbsc}{bx} {bx} {b}
1236 \DeclareFontSeriesChangeRule {sb}{bx} {bx} {b}
1237 \DeclareFontSeriesChangeRule {sbsx}{bx} {bx} {b}
1238 \DeclareFontSeriesChangeRule {sbx}{bx} {bx} {b}
1239 \DeclareFontSeriesChangeRule {sbex}{bx} {bx} {b}
1240 \DeclareFontSeriesChangeRule {sbux}{bx} {bx} {b}

1241 \DeclareFontSeriesChangeRule {buc}{bx} {bx} {b}
1242 \DeclareFontSeriesChangeRule {bec}{bx} {bx} {b}
1243 \DeclareFontSeriesChangeRule {bc}{bx} {bx} {b}
1244 \DeclareFontSeriesChangeRule {bsc}{bx} {bx} {b}
1245 \DeclareFontSeriesChangeRule {b}{bx} {bx} {b}
1246 \DeclareFontSeriesChangeRule {bsx}{bx} {bx} {b}
1247 \DeclareFontSeriesChangeRule {bx}{bx} {bx} {b}
1248 \DeclareFontSeriesChangeRule {bex}{bx} {bx} {b}
1249 \DeclareFontSeriesChangeRule {bux}{bx} {bx} {b}

1250 \DeclareFontSeriesChangeRule {ebuc}{bx} {bx} {b}
1251 \DeclareFontSeriesChangeRule {ebec}{bx} {bx} {b}
1252 \DeclareFontSeriesChangeRule {ebc}{bx} {bx} {b}
1253 \DeclareFontSeriesChangeRule {ebsc}{bx} {bx} {b}
1254 \DeclareFontSeriesChangeRule {eb}{bx} {bx} {b}
1255 \DeclareFontSeriesChangeRule {ebsx}{bx} {bx} {b}
1256 \DeclareFontSeriesChangeRule {ebx}{bx} {bx} {b}
1257 \DeclareFontSeriesChangeRule {ebex}{bx} {bx} {b}
1258 \DeclareFontSeriesChangeRule {ebux}{bx} {bx} {b}

1259 \DeclareFontSeriesChangeRule {ubuc}{bx} {bx} {b}
1260 \DeclareFontSeriesChangeRule {ubec}{bx} {bx} {b}
1261 \DeclareFontSeriesChangeRule {ubc}{bx} {bx} {b}
1262 \DeclareFontSeriesChangeRule {ubsc}{bx} {bx} {b}
1263 \DeclareFontSeriesChangeRule {ub}{bx} {bx} {b}
1264 \DeclareFontSeriesChangeRule {ubsx}{bx} {bx} {b}
1265 \DeclareFontSeriesChangeRule {ubx}{bx} {bx} {b}
1266 \DeclareFontSeriesChangeRule {ubex}{bx} {bx} {b}
1267 \DeclareFontSeriesChangeRule {ubux}{bx} {bx} {b}

```

Here are the special rules for $m?$. We offer m as alternative result series for all entries where this is not already the result series, because otherwise, if the result series does not exist, $m?$ would be tried as the series value, which cannot work. (Of course, this does not help if the m series does not exist either, but this should only affect very few fonts.)

```

1268 \DeclareFontSeriesChangeRule {uluc}{m?} {uc} {m}
1269 \DeclareFontSeriesChangeRule {ulec}{m?} {ec} {m}
1270 \DeclareFontSeriesChangeRule {ulc}{m?} {c} {m}
1271 \DeclareFontSeriesChangeRule {ulsc}{m?} {sc} {m}
1272 \DeclareFontSeriesChangeRule {ul}{m?} {m} {}
1273 \DeclareFontSeriesChangeRule {ulsx}{m?} {sx} {m}
1274 \DeclareFontSeriesChangeRule {ulx}{m?} {x} {m}
1275 \DeclareFontSeriesChangeRule {ulex}{m?} {ex} {m}
1276 \DeclareFontSeriesChangeRule {ulux}{m?} {ux} {m}

1277 \DeclareFontSeriesChangeRule {eluc}{m?} {uc} {m}
1278 \DeclareFontSeriesChangeRule {elec}{m?} {ec} {m}

```

```

1279 \DeclareFontSeriesChangeRule {elc}{m?} {c} {m}
1280 \DeclareFontSeriesChangeRule {elsc}{m?} {sc} {m}
1281 \DeclareFontSeriesChangeRule {el}{m?} {m} {}
1282 \DeclareFontSeriesChangeRule {elsx}{m?} {sx} {m}
1283 \DeclareFontSeriesChangeRule {elx}{m?} {x} {m}
1284 \DeclareFontSeriesChangeRule {elex}{m?} {ex} {m}
1285 \DeclareFontSeriesChangeRule {elux}{m?} {ux} {m}

1286 \DeclareFontSeriesChangeRule {luc}{m?} {uc} {m}
1287 \DeclareFontSeriesChangeRule {lec}{m?} {ec} {m}
1288 \DeclareFontSeriesChangeRule {lc}{m?} {c} {m}
1289 \DeclareFontSeriesChangeRule {lsc}{m?} {sc} {m}
1290 \DeclareFontSeriesChangeRule {l}{m?} {m} {}
1291 \DeclareFontSeriesChangeRule {lsx}{m?} {sx} {m}
1292 \DeclareFontSeriesChangeRule {lx}{m?} {x} {m}
1293 \DeclareFontSeriesChangeRule {lex}{m?} {ex} {m}
1294 \DeclareFontSeriesChangeRule {lux}{m?} {ux} {m}

1295 \DeclareFontSeriesChangeRule {sluc}{m?} {uc} {m}
1296 \DeclareFontSeriesChangeRule {slec}{m?} {ec} {m}
1297 \DeclareFontSeriesChangeRule {slc}{m?} {c} {m}
1298 \DeclareFontSeriesChangeRule {slsc}{m?} {sc} {m}
1299 \DeclareFontSeriesChangeRule {sl}{m?} {m} {}
1300 \DeclareFontSeriesChangeRule {slsx}{m?} {sx} {m}
1301 \DeclareFontSeriesChangeRule {slx}{m?} {x} {m}
1302 \DeclareFontSeriesChangeRule {slex}{m?} {ex} {m}
1303 \DeclareFontSeriesChangeRule {slux}{m?} {ux} {m}

1304 \DeclareFontSeriesChangeRule {uc}{m?} {uc} {m}
1305 \DeclareFontSeriesChangeRule {ec}{m?} {ec} {m}
1306 \DeclareFontSeriesChangeRule {c}{m?} {c} {m}
1307 \DeclareFontSeriesChangeRule {sc}{m?} {sc} {m}
1308 \DeclareFontSeriesChangeRule {m}{m?} {m} {}
1309 \DeclareFontSeriesChangeRule {sx}{m?} {sx} {m}
1310 \DeclareFontSeriesChangeRule {x}{m?} {x} {m}
1311 \DeclareFontSeriesChangeRule {ex}{m?} {ex} {m}
1312 \DeclareFontSeriesChangeRule {ux}{m?} {ux} {m}

1313 \DeclareFontSeriesChangeRule {sbuc}{m?} {uc} {m}
1314 \DeclareFontSeriesChangeRule {sbec}{m?} {ec} {m}
1315 \DeclareFontSeriesChangeRule {sbc}{m?} {c} {m}
1316 \DeclareFontSeriesChangeRule {sbsc}{m?} {sc} {m}
1317 \DeclareFontSeriesChangeRule {sb}{m?} {m} {}
1318 \DeclareFontSeriesChangeRule {sbsx}{m?} {sx} {m}
1319 \DeclareFontSeriesChangeRule {sbx}{m?} {x} {m}
1320 \DeclareFontSeriesChangeRule {sbex}{m?} {ex} {m}
1321 \DeclareFontSeriesChangeRule {sbux}{m?} {ux} {m}

1322 \DeclareFontSeriesChangeRule {buc}{m?} {uc} {m}
1323 \DeclareFontSeriesChangeRule {bec}{m?} {ec} {m}
1324 \DeclareFontSeriesChangeRule {bc}{m?} {c} {m}
1325 \DeclareFontSeriesChangeRule {bsc}{m?} {sc} {m}
1326 \DeclareFontSeriesChangeRule {b}{m?} {m} {}
1327 \DeclareFontSeriesChangeRule {bsx}{m?} {sx} {m}
1328 \DeclareFontSeriesChangeRule {bx}{m?} {x} {m}
1329 \DeclareFontSeriesChangeRule {bex}{m?} {ex} {m}
1330 \DeclareFontSeriesChangeRule {bux}{m?} {ux} {m}

```

```

1331 \DeclareFontSeriesChangeRule {ebuc}{m?} {uc} {m}
1332 \DeclareFontSeriesChangeRule {ebec}{m?} {ec} {m}
1333 \DeclareFontSeriesChangeRule {ebc}{m?} {c} {m}
1334 \DeclareFontSeriesChangeRule {ebsc}{m?} {sc} {m}
1335 \DeclareFontSeriesChangeRule {eb}{m?} {m} {}
1336 \DeclareFontSeriesChangeRule {ebsx}{m?} {sx} {m}
1337 \DeclareFontSeriesChangeRule {ebx}{m?} {x} {m}
1338 \DeclareFontSeriesChangeRule {ebex}{m?} {ex} {m}
1339 \DeclareFontSeriesChangeRule {ebux}{m?} {ux} {m}

1340 \DeclareFontSeriesChangeRule {ubuc}{m?} {uc} {m}
1341 \DeclareFontSeriesChangeRule {ubec}{m?} {ec} {m}
1342 \DeclareFontSeriesChangeRule {ubc}{m?} {c} {m}
1343 \DeclareFontSeriesChangeRule {ubsc}{m?} {sc} {m}
1344 \DeclareFontSeriesChangeRule {ub}{m?} {m} {}
1345 \DeclareFontSeriesChangeRule {ubsx}{m?} {sx} {m}
1346 \DeclareFontSeriesChangeRule {ubx}{m?} {x} {m}
1347 \DeclareFontSeriesChangeRule {ubex}{m?} {ex} {m}
1348 \DeclareFontSeriesChangeRule {ubux}{m?} {ux} {m}

```

And here are the special rules for ?m. Again, we offer m as alternative result series for all entries where this is not already the result series.

```

1349 \DeclareFontSeriesChangeRule {uluc}{?m} {ul} {m}
1350 \DeclareFontSeriesChangeRule {ulec}{?m} {ul} {m}
1351 \DeclareFontSeriesChangeRule {ulc}{?m} {ul} {m}
1352 \DeclareFontSeriesChangeRule {ulsc}{?m} {ul} {m}
1353 \DeclareFontSeriesChangeRule {ul}{?m} {ul} {m}
1354 \DeclareFontSeriesChangeRule {ulsx}{?m} {ul} {m}
1355 \DeclareFontSeriesChangeRule {ulx}{?m} {ul} {m}
1356 \DeclareFontSeriesChangeRule {ulex}{?m} {ul} {m}
1357 \DeclareFontSeriesChangeRule {ulux}{?m} {ul} {m}

1358 \DeclareFontSeriesChangeRule {eluc}{?m} {el} {m}
1359 \DeclareFontSeriesChangeRule {elec}{?m} {el} {m}
1360 \DeclareFontSeriesChangeRule {elc}{?m} {el} {m}
1361 \DeclareFontSeriesChangeRule {elsc}{?m} {el} {m}
1362 \DeclareFontSeriesChangeRule {el}{?m} {el} {m}
1363 \DeclareFontSeriesChangeRule {elsx}{?m} {el} {m}
1364 \DeclareFontSeriesChangeRule {elx}{?m} {el} {m}
1365 \DeclareFontSeriesChangeRule {elcx}{?m} {el} {m}
1366 \DeclareFontSeriesChangeRule {elux}{?m} {el} {m}

1367 \DeclareFontSeriesChangeRule {luc}{?m} {l} {m}
1368 \DeclareFontSeriesChangeRule {lec}{?m} {l} {m}
1369 \DeclareFontSeriesChangeRule {lc}{?m} {l} {m}
1370 \DeclareFontSeriesChangeRule {lsc}{?m} {l} {m}
1371 \DeclareFontSeriesChangeRule {l}{?m} {l} {m}
1372 \DeclareFontSeriesChangeRule {lsx}{?m} {l} {m}
1373 \DeclareFontSeriesChangeRule {lx}{?m} {l} {m}
1374 \DeclareFontSeriesChangeRule {lex}{?m} {l} {m}
1375 \DeclareFontSeriesChangeRule {lux}{?m} {l} {m}

1376 \DeclareFontSeriesChangeRule {sluc}{?m} {sl} {m}
1377 \DeclareFontSeriesChangeRule {slec}{?m} {sl} {m}
1378 \DeclareFontSeriesChangeRule {slc}{?m} {sl} {m}
1379 \DeclareFontSeriesChangeRule {slsc}{?m} {sl} {m}
1380 \DeclareFontSeriesChangeRule {sl}{?m} {sl} {m}

```



```

1381 \DeclareFontSeriesChangeRule {slsx}{?m} {sl} {m}
1382 \DeclareFontSeriesChangeRule {slx}{?m} {sl} {m}
1383 \DeclareFontSeriesChangeRule {slex}{?m} {sl} {m}
1384 \DeclareFontSeriesChangeRule {slux}{?m} {sl} {m}

1385 \DeclareFontSeriesChangeRule {uc}{?m} {m} {}
1386 \DeclareFontSeriesChangeRule {ec}{?m} {m} {}
1387 \DeclareFontSeriesChangeRule {c}{?m} {m} {}
1388 \DeclareFontSeriesChangeRule {sc}{?m} {m} {}
1389 \DeclareFontSeriesChangeRule {m}{?m} {m} {}
1390 \DeclareFontSeriesChangeRule {sx}{?m} {m} {}
1391 \DeclareFontSeriesChangeRule {x}{?m} {m} {}
1392 \DeclareFontSeriesChangeRule {ex}{?m} {m} {}
1393 \DeclareFontSeriesChangeRule {ux}{?m} {m} {}

1394 \DeclareFontSeriesChangeRule {sbuc}{?m} {sb} {m}
1395 \DeclareFontSeriesChangeRule {sbec}{?m} {sb} {m}
1396 \DeclareFontSeriesChangeRule {sbc}{?m} {sb} {m}
1397 \DeclareFontSeriesChangeRule {sbsc}{?m} {sb} {m}
1398 \DeclareFontSeriesChangeRule {sb}{?m} {sb} {m}
1399 \DeclareFontSeriesChangeRule {sbsx}{?m} {sb} {m}
1400 \DeclareFontSeriesChangeRule {sbx}{?m} {sb} {m}
1401 \DeclareFontSeriesChangeRule {sbex}{?m} {sb} {m}
1402 \DeclareFontSeriesChangeRule {sbux}{?m} {sb} {m}

1403 \DeclareFontSeriesChangeRule {buc}{?m} {b} {m}
1404 \DeclareFontSeriesChangeRule {bec}{?m} {b} {m}
1405 \DeclareFontSeriesChangeRule {bc}{?m} {b} {m}
1406 \DeclareFontSeriesChangeRule {bsc}{?m} {b} {m}
1407 \DeclareFontSeriesChangeRule {b}{?m} {b} {m}
1408 \DeclareFontSeriesChangeRule {bsx}{?m} {b} {m}
1409 \DeclareFontSeriesChangeRule {bx}{?m} {b} {m}
1410 \DeclareFontSeriesChangeRule {bex}{?m} {b} {m}
1411 \DeclareFontSeriesChangeRule {bux}{?m} {b} {m}

1412 \DeclareFontSeriesChangeRule {ebuc}{?m} {eb} {m}
1413 \DeclareFontSeriesChangeRule {ebec}{?m} {eb} {m}
1414 \DeclareFontSeriesChangeRule {ebc}{?m} {eb} {m}
1415 \DeclareFontSeriesChangeRule {ebsc}{?m} {eb} {m}
1416 \DeclareFontSeriesChangeRule {eb}{?m} {eb} {m}
1417 \DeclareFontSeriesChangeRule {ebsx}{?m} {eb} {m}
1418 \DeclareFontSeriesChangeRule {ebx}{?m} {eb} {m}
1419 \DeclareFontSeriesChangeRule {ebex}{?m} {eb} {m}
1420 \DeclareFontSeriesChangeRule {ebux}{?m} {eb} {m}

1421 \DeclareFontSeriesChangeRule {ubuc}{?m} {ub} {m}
1422 \DeclareFontSeriesChangeRule {ubec}{?m} {ub} {m}
1423 \DeclareFontSeriesChangeRule {ubc}{?m} {ub} {m}
1424 \DeclareFontSeriesChangeRule {ubsc}{?m} {ub} {m}
1425 \DeclareFontSeriesChangeRule {ub}{?m} {ub} {m}
1426 \DeclareFontSeriesChangeRule {ubsx}{?m} {ub} {m}
1427 \DeclareFontSeriesChangeRule {ubx}{?m} {ub} {m}
1428 \DeclareFontSeriesChangeRule {ubex}{?m} {ub} {m}
1429 \DeclareFontSeriesChangeRule {ubux}{?m} {ub} {m}

1430 </2ekernel | latexrelease>
1431 <latexrelease>\EndIncludeInRelease

```

Supporting rollback ...

```
1432 <latexrelease>\IncludeInRelease{2020/02/02}%
1433 <latexrelease> \DeclareFontSeriesChangeRule{Series change rules}%
```

The next definition is only needed if somebody rolls forward from a release older than 2020-02-02 but not to the latest version but one before 2025-06-01. Pretty unlikely, but ...

```
1434 <latexrelease>
1435 <latexrelease>\def\DeclareFontSeriesChangeRule#1#2#3#4{%
1436 <latexrelease> \@namedef{series@#1@#2}{#{#3}{#4}}
1437 <latexrelease>
```

The huge set of declarations below are those from 2020-02-02 plus all from above that were newly added (but now with empty result and alternative result arguments). To compile this list I sorted both together and then dropped entries appearing twice. This is why the sorting is now different from the one above.

```
1438 <latexrelease>\DeclareFontSeriesChangeRule {bc}{bx} {} {}
1439 <latexrelease>\DeclareFontSeriesChangeRule {bc}{b}{bc}{}
1440 <latexrelease>\DeclareFontSeriesChangeRule {bc}{c}{bc}{}
1441 <latexrelease>\DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
1442 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
1443 <latexrelease>\DeclareFontSeriesChangeRule {bc}{el}{elc}{}
1444 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ex} {} {}
1445 <latexrelease>\DeclareFontSeriesChangeRule {bc}{l}{lc}{}
1446 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
1447 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
1448 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sl}{slc}{}
1449 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sx} {} {}
1450 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
1451 <latexrelease>\DeclareFontSeriesChangeRule {bc}{uc} {} {}
1452 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
1453 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ux} {} {}
1454 <latexrelease>\DeclareFontSeriesChangeRule {bc}{x}{bx}{}
1455 <latexrelease>\DeclareFontSeriesChangeRule {bec}{bx} {} {}
1456 <latexrelease>\DeclareFontSeriesChangeRule {bec}{b} {} {}
1457 <latexrelease>\DeclareFontSeriesChangeRule {bec}{c} {} {}
1458 <latexrelease>\DeclareFontSeriesChangeRule {bec}{eb} {} {}
1459 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ec} {} {}
1460 <latexrelease>\DeclareFontSeriesChangeRule {bec}{el} {} {}
1461 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ex} {} {}
1462 <latexrelease>\DeclareFontSeriesChangeRule {bec}{l} {} {}
1463 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sb} {} {}
1464 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sc} {} {}
1465 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sl} {} {}
1466 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sx} {} {}
1467 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ub} {} {}
1468 <latexrelease>\DeclareFontSeriesChangeRule {bec}{uc} {} {}
1469 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ul} {} {}
1470 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ux} {} {}
1471 <latexrelease>\DeclareFontSeriesChangeRule {bec}{x} {} {}
1472 <latexrelease>\DeclareFontSeriesChangeRule {bex}{?m} {} {}
1473 <latexrelease>\DeclareFontSeriesChangeRule {bex}{bx} {} {}
1474 <latexrelease>\DeclareFontSeriesChangeRule {bex}{b} {} {}
1475 <latexrelease>\DeclareFontSeriesChangeRule {bex}{c} {} {}
```

```

1476 \latexrelease\DeclareFontSeriesChangeRule {bex}{eb} {} {}
1477 \latexrelease\DeclareFontSeriesChangeRule {bex}{ec} {} {}
1478 \latexrelease\DeclareFontSeriesChangeRule {bex}{el} {} {}
1479 \latexrelease\DeclareFontSeriesChangeRule {bex}{ex} {} {}
1480 \latexrelease\DeclareFontSeriesChangeRule {bex}{l} {} {}
1481 \latexrelease\DeclareFontSeriesChangeRule {bex}{m?} {} {}
1482 \latexrelease\DeclareFontSeriesChangeRule {bex}{sb} {} {}
1483 \latexrelease\DeclareFontSeriesChangeRule {bex}{sc} {} {}
1484 \latexrelease\DeclareFontSeriesChangeRule {bex}{sl} {} {}
1485 \latexrelease\DeclareFontSeriesChangeRule {bex}{sx} {} {}
1486 \latexrelease\DeclareFontSeriesChangeRule {bex}{ub} {} {}
1487 \latexrelease\DeclareFontSeriesChangeRule {bex}{uc} {} {}
1488 \latexrelease\DeclareFontSeriesChangeRule {bex}{ul} {} {}
1489 \latexrelease\DeclareFontSeriesChangeRule {bex}{ux} {} {}
1490 \latexrelease\DeclareFontSeriesChangeRule {bex}{x} {} {}
1491 \latexrelease\DeclareFontSeriesChangeRule {bsc}{bx} {} {}
1492 \latexrelease\DeclareFontSeriesChangeRule {bsc}{b} {} {}
1493 \latexrelease\DeclareFontSeriesChangeRule {bsc}{c} {} {}
1494 \latexrelease\DeclareFontSeriesChangeRule {bsc}{eb} {} {}
1495 \latexrelease\DeclareFontSeriesChangeRule {bsc}{ec} {} {}
1496 \latexrelease\DeclareFontSeriesChangeRule {bsc}{el} {} {}
1497 \latexrelease\DeclareFontSeriesChangeRule {bsc}{ex} {} {}
1498 \latexrelease\DeclareFontSeriesChangeRule {bsc}{l} {} {}
1499 \latexrelease\DeclareFontSeriesChangeRule {bsc}{sb} {} {}
1500 \latexrelease\DeclareFontSeriesChangeRule {bsc}{sc} {} {}
1501 \latexrelease\DeclareFontSeriesChangeRule {bsc}{sl} {} {}
1502 \latexrelease\DeclareFontSeriesChangeRule {bsc}{sx} {} {}
1503 \latexrelease\DeclareFontSeriesChangeRule {bsc}{ub} {} {}
1504 \latexrelease\DeclareFontSeriesChangeRule {bsc}{uc} {} {}
1505 \latexrelease\DeclareFontSeriesChangeRule {bsc}{ul} {} {}
1506 \latexrelease\DeclareFontSeriesChangeRule {bsc}{ux} {} {}
1507 \latexrelease\DeclareFontSeriesChangeRule {bsc}{x} {} {}
1508 \latexrelease\DeclareFontSeriesChangeRule {bsx}{?m} {} {}
1509 \latexrelease\DeclareFontSeriesChangeRule {bsx}{bx} {} {}
1510 \latexrelease\DeclareFontSeriesChangeRule {bsx}{b} {} {}
1511 \latexrelease\DeclareFontSeriesChangeRule {bsx}{c} {} {}
1512 \latexrelease\DeclareFontSeriesChangeRule {bsx}{eb} {} {}
1513 \latexrelease\DeclareFontSeriesChangeRule {bsx}{ec} {} {}
1514 \latexrelease\DeclareFontSeriesChangeRule {bsx}{el} {} {}
1515 \latexrelease\DeclareFontSeriesChangeRule {bsx}{ex} {} {}
1516 \latexrelease\DeclareFontSeriesChangeRule {bsx}{l} {} {}
1517 \latexrelease\DeclareFontSeriesChangeRule {bsx}{m?} {} {}
1518 \latexrelease\DeclareFontSeriesChangeRule {bsx}{sb} {} {}
1519 \latexrelease\DeclareFontSeriesChangeRule {bsx}{sc} {} {}
1520 \latexrelease\DeclareFontSeriesChangeRule {bsx}{sl} {} {}
1521 \latexrelease\DeclareFontSeriesChangeRule {bsx}{sx} {} {}
1522 \latexrelease\DeclareFontSeriesChangeRule {bsx}{ub} {} {}
1523 \latexrelease\DeclareFontSeriesChangeRule {bsx}{uc} {} {}
1524 \latexrelease\DeclareFontSeriesChangeRule {bsx}{ul} {} {}
1525 \latexrelease\DeclareFontSeriesChangeRule {bsx}{ux} {} {}
1526 \latexrelease\DeclareFontSeriesChangeRule {bsx}{x} {} {}
1527 \latexrelease\DeclareFontSeriesChangeRule {buc}{?m} {} {}
1528 \latexrelease\DeclareFontSeriesChangeRule {buc}{bx} {} {}
1529 \latexrelease\DeclareFontSeriesChangeRule {buc}{b} {} {}

```

```

1530 \latexrelease\DeclareFontSeriesChangeRule {buc}{c} {} {}
1531 \latexrelease\DeclareFontSeriesChangeRule {buc}{eb} {} {}
1532 \latexrelease\DeclareFontSeriesChangeRule {buc}{ec} {} {}
1533 \latexrelease\DeclareFontSeriesChangeRule {buc}{el} {} {}
1534 \latexrelease\DeclareFontSeriesChangeRule {buc}{ex} {} {}
1535 \latexrelease\DeclareFontSeriesChangeRule {buc}{l} {} {}
1536 \latexrelease\DeclareFontSeriesChangeRule {buc}{m?} {} {}
1537 \latexrelease\DeclareFontSeriesChangeRule {buc}{sb} {} {}
1538 \latexrelease\DeclareFontSeriesChangeRule {buc}{sc} {} {}
1539 \latexrelease\DeclareFontSeriesChangeRule {buc}{sl} {} {}
1540 \latexrelease\DeclareFontSeriesChangeRule {buc}{sx} {} {}
1541 \latexrelease\DeclareFontSeriesChangeRule {buc}{ub} {} {}
1542 \latexrelease\DeclareFontSeriesChangeRule {buc}{uc} {} {}
1543 \latexrelease\DeclareFontSeriesChangeRule {buc}{ul} {} {}
1544 \latexrelease\DeclareFontSeriesChangeRule {buc}{ux} {} {}
1545 \latexrelease\DeclareFontSeriesChangeRule {buc}{x} {} {}
1546 \latexrelease\DeclareFontSeriesChangeRule {bux}{?m} {} {}
1547 \latexrelease\DeclareFontSeriesChangeRule {bux}{bx} {} {}
1548 \latexrelease\DeclareFontSeriesChangeRule {bux}{b} {} {}
1549 \latexrelease\DeclareFontSeriesChangeRule {bux}{c} {} {}
1550 \latexrelease\DeclareFontSeriesChangeRule {bux}{eb} {} {}
1551 \latexrelease\DeclareFontSeriesChangeRule {bux}{ec} {} {}
1552 \latexrelease\DeclareFontSeriesChangeRule {bux}{el} {} {}
1553 \latexrelease\DeclareFontSeriesChangeRule {bux}{ex} {} {}
1554 \latexrelease\DeclareFontSeriesChangeRule {bux}{l} {} {}
1555 \latexrelease\DeclareFontSeriesChangeRule {bux}{m?} {} {}
1556 \latexrelease\DeclareFontSeriesChangeRule {bux}{sb} {} {}
1557 \latexrelease\DeclareFontSeriesChangeRule {bux}{sc} {} {}
1558 \latexrelease\DeclareFontSeriesChangeRule {bux}{sl} {} {}
1559 \latexrelease\DeclareFontSeriesChangeRule {bux}{sx} {} {}
1560 \latexrelease\DeclareFontSeriesChangeRule {bux}{ub} {} {}
1561 \latexrelease\DeclareFontSeriesChangeRule {bux}{uc} {} {}
1562 \latexrelease\DeclareFontSeriesChangeRule {bux}{ul} {} {}
1563 \latexrelease\DeclareFontSeriesChangeRule {bux}{ux} {} {}
1564 \latexrelease\DeclareFontSeriesChangeRule {bux}{x} {} {}
1565 \latexrelease\DeclareFontSeriesChangeRule {bx}{bx} {} {}
1566 \latexrelease\DeclareFontSeriesChangeRule {bx}{b}{bx} {} {}
1567 \latexrelease\DeclareFontSeriesChangeRule {bx}{c} {bc} {bx}
1568 \latexrelease\DeclareFontSeriesChangeRule {bx}{eb}{ebx} {} {}
1569 \latexrelease\DeclareFontSeriesChangeRule {bx}{ec} {bec} {bx}
1570 \latexrelease\DeclareFontSeriesChangeRule {bx}{el}{elx} {} {}
1571 \latexrelease\DeclareFontSeriesChangeRule {bx}{ex} {} {}
1572 \latexrelease\DeclareFontSeriesChangeRule {bx}{l}{lx} {} {}
1573 \latexrelease\DeclareFontSeriesChangeRule {bx}{sb} {sbx} {} {}
1574 \latexrelease\DeclareFontSeriesChangeRule {bx}{sc} {bsc} {bx}
1575 \latexrelease\DeclareFontSeriesChangeRule {bx}{sl}{slx} {} {}
1576 \latexrelease\DeclareFontSeriesChangeRule {bx}{sx} {} {}
1577 \latexrelease\DeclareFontSeriesChangeRule {bx}{ub}{ubx} {} {}
1578 \latexrelease\DeclareFontSeriesChangeRule {bx}{uc} {} {}
1579 \latexrelease\DeclareFontSeriesChangeRule {bx}{ul}{ulx} {} {}
1580 \latexrelease\DeclareFontSeriesChangeRule {bx}{ux} {} {}
1581 \latexrelease\DeclareFontSeriesChangeRule {bx}{x}{bx} {} {}
1582 \latexrelease\DeclareFontSeriesChangeRule {b}{bx} {bx} {b}
1583 \latexrelease\DeclareFontSeriesChangeRule {b}{c} {bc} {b}

```

```

1584 <latexrelease>\DeclareFontSeriesChangeRule {b}{eb} {} {}
1585 <latexrelease>\DeclareFontSeriesChangeRule {b}{ec} {bec} {b}
1586 <latexrelease>\DeclareFontSeriesChangeRule {b}{ex} {} {}
1587 <latexrelease>\DeclareFontSeriesChangeRule {b}{sb} {sb} {b}
1588 <latexrelease>\DeclareFontSeriesChangeRule {b}{sc} {bsc} {b}
1589 <latexrelease>\DeclareFontSeriesChangeRule {b}{sx} {} {}
1590 <latexrelease>\DeclareFontSeriesChangeRule {b}{ub} {} {}
1591 <latexrelease>\DeclareFontSeriesChangeRule {b}{uc} {} {}
1592 <latexrelease>\DeclareFontSeriesChangeRule {b}{ux} {} {}
1593 <latexrelease>\DeclareFontSeriesChangeRule {b}{x} {bx} {b}
1594 <latexrelease>\DeclareFontSeriesChangeRule {c}{bx} {bx} {b}
1595 <latexrelease>\DeclareFontSeriesChangeRule {c}{b}{bc}{b}
1596 <latexrelease>\DeclareFontSeriesChangeRule {c}{eb}{ebc}{b}
1597 <latexrelease>\DeclareFontSeriesChangeRule {c}{ec} {} {}
1598 <latexrelease>\DeclareFontSeriesChangeRule {c}{el}{elc}{b}
1599 <latexrelease>\DeclareFontSeriesChangeRule {c}{l}{lc}{b}
1600 <latexrelease>\DeclareFontSeriesChangeRule {c}{sb}{sbc}{b}
1601 <latexrelease>\DeclareFontSeriesChangeRule {c}{sc} {} {}
1602 <latexrelease>\DeclareFontSeriesChangeRule {c}{sl}{slc}{b}
1603 <latexrelease>\DeclareFontSeriesChangeRule {c}{ub}{ubc}{b}
1604 <latexrelease>\DeclareFontSeriesChangeRule {c}{uc} {} {}
1605 <latexrelease>\DeclareFontSeriesChangeRule {c}{x}{x}{m}
1606 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{bx} {} {}
1607 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{b}{bc}{b}
1608 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{c}{ebc}{b}
1609 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{b}
1610 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}
1611 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{el}{elc}{b}
1612 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ex} {} {}
1613 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{l}{lc}{b}
1614 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{b}
1615 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}
1616 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sl}{slc}{b}
1617 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sx} {} {}
1618 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{b}
1619 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{uc} {} {}
1620 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{b}
1621 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ux} {} {}
1622 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{x}{ebx}{b}
1623 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{bx} {} {}
1624 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{b} {} {}
1625 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{c} {} {}
1626 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{eb} {} {}
1627 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ec} {} {}
1628 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{el} {} {}
1629 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ex} {} {}
1630 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{l} {} {}
1631 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sb} {} {}
1632 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sc} {} {}
1633 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sl} {} {}
1634 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sx} {} {}
1635 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ub} {} {}
1636 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{uc} {} {}
1637 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ul} {} {}

```

```

1638 \latexrelease\DeclareFontSeriesChangeRule {ebec}{ux} {} {}
1639 \latexrelease\DeclareFontSeriesChangeRule {ebec}{x} {} {}
1640 \latexrelease\DeclareFontSeriesChangeRule {ebex}{?m} {} {}
1641 \latexrelease\DeclareFontSeriesChangeRule {ebex}{bx} {} {}
1642 \latexrelease\DeclareFontSeriesChangeRule {ebex}{b} {} {}
1643 \latexrelease\DeclareFontSeriesChangeRule {ebex}{c} {} {}
1644 \latexrelease\DeclareFontSeriesChangeRule {ebex}{eb} {} {}
1645 \latexrelease\DeclareFontSeriesChangeRule {ebex}{ec} {} {}
1646 \latexrelease\DeclareFontSeriesChangeRule {ebex}{el} {} {}
1647 \latexrelease\DeclareFontSeriesChangeRule {ebex}{ex} {} {}
1648 \latexrelease\DeclareFontSeriesChangeRule {ebex}{l} {} {}
1649 \latexrelease\DeclareFontSeriesChangeRule {ebex}{m?} {} {}
1650 \latexrelease\DeclareFontSeriesChangeRule {ebex}{sb} {} {}
1651 \latexrelease\DeclareFontSeriesChangeRule {ebex}{sc} {} {}
1652 \latexrelease\DeclareFontSeriesChangeRule {ebex}{sl} {} {}
1653 \latexrelease\DeclareFontSeriesChangeRule {ebex}{sx} {} {}
1654 \latexrelease\DeclareFontSeriesChangeRule {ebex}{ub} {} {}
1655 \latexrelease\DeclareFontSeriesChangeRule {ebex}{uc} {} {}
1656 \latexrelease\DeclareFontSeriesChangeRule {ebex}{ul} {} {}
1657 \latexrelease\DeclareFontSeriesChangeRule {ebex}{ux} {} {}
1658 \latexrelease\DeclareFontSeriesChangeRule {ebex}{x} {} {}
1659 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{bx} {} {}
1660 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{b} {} {}
1661 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{c} {} {}
1662 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{eb} {} {}
1663 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{ec} {} {}
1664 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{el} {} {}
1665 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{ex} {} {}
1666 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{l} {} {}
1667 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{sb} {} {}
1668 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{sc} {} {}
1669 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{sl} {} {}
1670 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{sx} {} {}
1671 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{ub} {} {}
1672 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{uc} {} {}
1673 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{ul} {} {}
1674 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{ux} {} {}
1675 \latexrelease\DeclareFontSeriesChangeRule {ebsc}{x} {} {}
1676 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{?m} {} {}
1677 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{bx} {} {}
1678 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{b} {} {}
1679 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{c} {} {}
1680 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{eb} {} {}
1681 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{ec} {} {}
1682 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{el} {} {}
1683 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{ex} {} {}
1684 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{l} {} {}
1685 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{m?} {} {}
1686 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{sb} {} {}
1687 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{sc} {} {}
1688 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{sl} {} {}
1689 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{sx} {} {}
1690 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{ub} {} {}
1691 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{uc} {} {}

```

```

1692 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{ul} {} {}
1693 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{ux} {} {}
1694 \latexrelease\DeclareFontSeriesChangeRule {ebsx}{x} {} {}
1695 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{?m} {} {}
1696 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{bx} {} {}
1697 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{b} {} {}
1698 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{c} {} {}
1699 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{eb} {} {}
1700 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{ec} {} {}
1701 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{el} {} {}
1702 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{ex} {} {}
1703 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{l} {} {}
1704 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{m?} {} {}
1705 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{sb} {} {}
1706 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{sc} {} {}
1707 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{sl} {} {}
1708 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{sx} {} {}
1709 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{ub} {} {}
1710 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{uc} {} {}
1711 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{ul} {} {}
1712 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{ux} {} {}
1713 \latexrelease\DeclareFontSeriesChangeRule {ebuc}{x} {} {}
1714 \latexrelease\DeclareFontSeriesChangeRule {ebux}{?m} {} {}
1715 \latexrelease\DeclareFontSeriesChangeRule {ebux}{bx} {} {}
1716 \latexrelease\DeclareFontSeriesChangeRule {ebux}{b} {} {}
1717 \latexrelease\DeclareFontSeriesChangeRule {ebux}{c} {} {}
1718 \latexrelease\DeclareFontSeriesChangeRule {ebux}{eb} {} {}
1719 \latexrelease\DeclareFontSeriesChangeRule {ebux}{ec} {} {}
1720 \latexrelease\DeclareFontSeriesChangeRule {ebux}{el} {} {}
1721 \latexrelease\DeclareFontSeriesChangeRule {ebux}{ex} {} {}
1722 \latexrelease\DeclareFontSeriesChangeRule {ebux}{l} {} {}
1723 \latexrelease\DeclareFontSeriesChangeRule {ebux}{m?} {} {}
1724 \latexrelease\DeclareFontSeriesChangeRule {ebux}{sb} {} {}
1725 \latexrelease\DeclareFontSeriesChangeRule {ebux}{sc} {} {}
1726 \latexrelease\DeclareFontSeriesChangeRule {ebux}{sl} {} {}
1727 \latexrelease\DeclareFontSeriesChangeRule {ebux}{sx} {} {}
1728 \latexrelease\DeclareFontSeriesChangeRule {ebux}{ub} {} {}
1729 \latexrelease\DeclareFontSeriesChangeRule {ebux}{uc} {} {}
1730 \latexrelease\DeclareFontSeriesChangeRule {ebux}{ul} {} {}
1731 \latexrelease\DeclareFontSeriesChangeRule {ebux}{ux} {} {}
1732 \latexrelease\DeclareFontSeriesChangeRule {ebux}{x} {} {}
1733 \latexrelease\DeclareFontSeriesChangeRule {ebx}{bx} {} {}
1734 \latexrelease\DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
1735 \latexrelease\DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
1736 \latexrelease\DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
1737 \latexrelease\DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
1738 \latexrelease\DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
1739 \latexrelease\DeclareFontSeriesChangeRule {ebx}{ex} {} {}
1740 \latexrelease\DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
1741 \latexrelease\DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
1742 \latexrelease\DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
1743 \latexrelease\DeclareFontSeriesChangeRule {ebx}{sl}{slx}{}
1744 \latexrelease\DeclareFontSeriesChangeRule {ebx}{sx} {} {}
1745 \latexrelease\DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}

```

```

1746 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{uc} {} {}
1747 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
1748 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ux} {} {}
1749 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}
1750 <latexrelease>\DeclareFontSeriesChangeRule {eb}{bx} {} {}
1751 <latexrelease>\DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
1752 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
1753 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ex} {} {}
1754 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sb} {} {}
1755 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
1756 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sx} {} {}
1757 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ub} {} {}
1758 <latexrelease>\DeclareFontSeriesChangeRule {eb}{uc} {} {}
1759 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ux} {} {}
1760 <latexrelease>\DeclareFontSeriesChangeRule {eb}{x}{ebx}{}
1761 <latexrelease>\DeclareFontSeriesChangeRule {ec}{bx} {bx} {b}
1762 <latexrelease>\DeclareFontSeriesChangeRule {ec}{b}{bec}{}
1763 <latexrelease>\DeclareFontSeriesChangeRule {ec}{eb}{ebec}{}
1764 <latexrelease>\DeclareFontSeriesChangeRule {ec}{el}{elec}{}
1765 <latexrelease>\DeclareFontSeriesChangeRule {ec}{l}{lec}{}
1766 <latexrelease>\DeclareFontSeriesChangeRule {ec}{sb}{sbec}{}
1767 <latexrelease>\DeclareFontSeriesChangeRule {ec}{sl}{slec}{}
1768 <latexrelease>\DeclareFontSeriesChangeRule {ec}{ub}{ubec}{}
1769 <latexrelease>\DeclareFontSeriesChangeRule {ec}{x}{x}{m}
1770 <latexrelease>\DeclareFontSeriesChangeRule {elc}{bx} {} {}
1771 <latexrelease>\DeclareFontSeriesChangeRule {elc}{b}{bc}{}
1772 <latexrelease>\DeclareFontSeriesChangeRule {elc}{c}{elc}{}
1773 <latexrelease>\DeclareFontSeriesChangeRule {elc}{eb}{ebc}{}
1774 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ec}{elec}{}
1775 <latexrelease>\DeclareFontSeriesChangeRule {elc}{el}{elc}{}
1776 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ex} {} {}
1777 <latexrelease>\DeclareFontSeriesChangeRule {elc}{l}{lc}{}
1778 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sb}{sbc}{}
1779 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sc}{elsc}{}
1780 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sl}{slc}{}
1781 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sx} {} {}
1782 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ub}{ubc}{}
1783 <latexrelease>\DeclareFontSeriesChangeRule {elc}{uc} {} {}
1784 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ul}{ulc}{}
1785 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ux} {} {}
1786 <latexrelease>\DeclareFontSeriesChangeRule {elc}{x}{elx}{}
1787 <latexrelease>\DeclareFontSeriesChangeRule {elec}{bx} {} {}
1788 <latexrelease>\DeclareFontSeriesChangeRule {elec}{b} {} {}
1789 <latexrelease>\DeclareFontSeriesChangeRule {elec}{c} {} {}
1790 <latexrelease>\DeclareFontSeriesChangeRule {elec}{eb} {} {}
1791 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ec} {} {}
1792 <latexrelease>\DeclareFontSeriesChangeRule {elec}{el} {} {}
1793 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ex} {} {}
1794 <latexrelease>\DeclareFontSeriesChangeRule {elec}{l} {} {}
1795 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sb} {} {}
1796 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sc} {} {}
1797 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sl} {} {}
1798 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sx} {} {}
1799 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ub} {} {}

```



```

1800 <latexrelease>\DeclareFontSeriesChangeRule {elec}{uc} {} {}
1801 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ul} {} {}
1802 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ux} {} {}
1803 <latexrelease>\DeclareFontSeriesChangeRule {elec}{x} {} {}
1804 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{?m} {} {}
1805 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{bx} {} {}
1806 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{b} {} {}
1807 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{c} {} {}
1808 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{eb} {} {}
1809 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{ec} {} {}
1810 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{el} {} {}
1811 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{ex} {} {}
1812 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{l} {} {}
1813 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{m?} {} {}
1814 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{sb} {} {}
1815 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{sc} {} {}
1816 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{sl} {} {}
1817 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{sx} {} {}
1818 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{ub} {} {}
1819 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{uc} {} {}
1820 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{ul} {} {}
1821 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{ux} {} {}
1822 <latexrelease>\DeclareFontSeriesChangeRule {ellex}{x} {} {}
1823 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{bx} {} {}
1824 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{b} {} {}
1825 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{c} {} {}
1826 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{eb} {} {}
1827 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ec} {} {}
1828 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{el} {} {}
1829 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ex} {} {}
1830 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{l} {} {}
1831 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sb} {} {}
1832 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sc} {} {}
1833 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sl} {} {}
1834 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sx} {} {}
1835 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ub} {} {}
1836 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{uc} {} {}
1837 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ul} {} {}
1838 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ux} {} {}
1839 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{x} {} {}
1840 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{?m} {} {}
1841 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{bx} {} {}
1842 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{b} {} {}
1843 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{c} {} {}
1844 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{eb} {} {}
1845 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ec} {} {}
1846 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{el} {} {}
1847 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ex} {} {}
1848 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{l} {} {}
1849 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{m?} {} {}
1850 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sb} {} {}
1851 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sc} {} {}
1852 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sl} {} {}
1853 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sx} {} {}

```

```

1854 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ub} {} {}
1855 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{uc} {} {}
1856 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ul} {} {}
1857 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ux} {} {}
1858 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{x} {} {}
1859 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{?m} {} {}
1860 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{bx} {} {}
1861 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{b} {} {}
1862 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{c} {} {}
1863 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{eb} {} {}
1864 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ec} {} {}
1865 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{el} {} {}
1866 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ex} {} {}
1867 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{l} {} {}
1868 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{m?} {} {}
1869 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sb} {} {}
1870 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sc} {} {}
1871 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sl} {} {}
1872 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sx} {} {}
1873 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ub} {} {}
1874 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{uc} {} {}
1875 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ul} {} {}
1876 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ux} {} {}
1877 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{x} {} {}
1878 <latexrelease>\DeclareFontSeriesChangeRule {elux}{?m} {} {}
1879 <latexrelease>\DeclareFontSeriesChangeRule {elux}{bx} {} {}
1880 <latexrelease>\DeclareFontSeriesChangeRule {elux}{b} {} {}
1881 <latexrelease>\DeclareFontSeriesChangeRule {elux}{c} {} {}
1882 <latexrelease>\DeclareFontSeriesChangeRule {elux}{eb} {} {}
1883 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ec} {} {}
1884 <latexrelease>\DeclareFontSeriesChangeRule {elux}{el} {} {}
1885 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ex} {} {}
1886 <latexrelease>\DeclareFontSeriesChangeRule {elux}{l} {} {}
1887 <latexrelease>\DeclareFontSeriesChangeRule {elux}{m?} {} {}
1888 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sb} {} {}
1889 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sc} {} {}
1890 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sl} {} {}
1891 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sx} {} {}
1892 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ub} {} {}
1893 <latexrelease>\DeclareFontSeriesChangeRule {elux}{uc} {} {}
1894 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ul} {} {}
1895 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ux} {} {}
1896 <latexrelease>\DeclareFontSeriesChangeRule {elux}{x} {} {}
1897 <latexrelease>\DeclareFontSeriesChangeRule {elx}{bx} {} {}
1898 <latexrelease>\DeclareFontSeriesChangeRule {elx}{b}{bx}{}
1899 <latexrelease>\DeclareFontSeriesChangeRule {elx}{c}{elc}{}
1900 <latexrelease>\DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
1901 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
1902 <latexrelease>\DeclareFontSeriesChangeRule {elx}{el}{elx}{}
1903 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ex} {} {}
1904 <latexrelease>\DeclareFontSeriesChangeRule {elx}{l}{lx}{}
1905 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
1906 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
1907 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sl}{slx}{}

```

```

1908 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sx} {} {}
1909 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
1910 <latexrelease>\DeclareFontSeriesChangeRule {elx}{uc} {} {}
1911 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
1912 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ux} {} {}
1913 <latexrelease>\DeclareFontSeriesChangeRule {elx}{x}{elx}{}
1914 <latexrelease>\DeclareFontSeriesChangeRule {el}{bx} {} {}
1915 <latexrelease>\DeclareFontSeriesChangeRule {el}{c}{elc}{}
1916 <latexrelease>\DeclareFontSeriesChangeRule {el}{eb} {} {}
1917 <latexrelease>\DeclareFontSeriesChangeRule {el}{ec}{elec}{}
1918 <latexrelease>\DeclareFontSeriesChangeRule {el}{ex} {} {}
1919 <latexrelease>\DeclareFontSeriesChangeRule {el}{sb} {} {}
1920 <latexrelease>\DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
1921 <latexrelease>\DeclareFontSeriesChangeRule {el}{sx} {} {}
1922 <latexrelease>\DeclareFontSeriesChangeRule {el}{ub} {} {}
1923 <latexrelease>\DeclareFontSeriesChangeRule {el}{uc} {} {}
1924 <latexrelease>\DeclareFontSeriesChangeRule {el}{ux} {} {}
1925 <latexrelease>\DeclareFontSeriesChangeRule {el}{x}{elx}{}
1926 <latexrelease>\DeclareFontSeriesChangeRule {ex}{?m} {} {}
1927 <latexrelease>\DeclareFontSeriesChangeRule {ex}{bx} {} {}
1928 <latexrelease>\DeclareFontSeriesChangeRule {ex}{b} {} {}
1929 <latexrelease>\DeclareFontSeriesChangeRule {ex}{eb} {} {}
1930 <latexrelease>\DeclareFontSeriesChangeRule {ex}{el} {} {}
1931 <latexrelease>\DeclareFontSeriesChangeRule {ex}{l} {} {}
1932 <latexrelease>\DeclareFontSeriesChangeRule {ex}{m?} {} {}
1933 <latexrelease>\DeclareFontSeriesChangeRule {ex}{sb} {} {}
1934 <latexrelease>\DeclareFontSeriesChangeRule {ex}{sl} {} {}
1935 <latexrelease>\DeclareFontSeriesChangeRule {ex}{ub} {} {}
1936 <latexrelease>\DeclareFontSeriesChangeRule {ex}{ul} {} {}
1937 <latexrelease>\DeclareFontSeriesChangeRule {lc}{bx} {} {}
1938 <latexrelease>\DeclareFontSeriesChangeRule {lc}{b}{bc}{}
1939 <latexrelease>\DeclareFontSeriesChangeRule {lc}{c}{lc}{}
1940 <latexrelease>\DeclareFontSeriesChangeRule {lc}{eb}{ebc}{}
1941 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ec}{lec}{}
1942 <latexrelease>\DeclareFontSeriesChangeRule {lc}{el}{elc}{}
1943 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ex} {} {}
1944 <latexrelease>\DeclareFontSeriesChangeRule {lc}{l}{lc}{}
1945 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sb}{sbc}{}
1946 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sc}{lsc}{}
1947 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sl}{slc}{}
1948 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sx} {} {}
1949 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ub}{ubc}{}
1950 <latexrelease>\DeclareFontSeriesChangeRule {lc}{uc} {} {}
1951 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ul}{ulc}{}
1952 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ux} {} {}
1953 <latexrelease>\DeclareFontSeriesChangeRule {lc}{x}{lx}{}
1954 <latexrelease>\DeclareFontSeriesChangeRule {lec}{bx} {} {}
1955 <latexrelease>\DeclareFontSeriesChangeRule {lec}{b} {} {}
1956 <latexrelease>\DeclareFontSeriesChangeRule {lec}{c} {} {}
1957 <latexrelease>\DeclareFontSeriesChangeRule {lec}{eb} {} {}
1958 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ec} {} {}
1959 <latexrelease>\DeclareFontSeriesChangeRule {lec}{el} {} {}
1960 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ex} {} {}
1961 <latexrelease>\DeclareFontSeriesChangeRule {lec}{l} {} {}

```

```

1962 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sb} {} {}
1963 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sc} {} {}
1964 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sl} {} {}
1965 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sx} {} {}
1966 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ub} {} {}
1967 <latexrelease>\DeclareFontSeriesChangeRule {lec}{uc} {} {}
1968 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ul} {} {}
1969 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ux} {} {}
1970 <latexrelease>\DeclareFontSeriesChangeRule {lec}{x} {} {}
1971 <latexrelease>\DeclareFontSeriesChangeRule {lex}{?m} {} {}
1972 <latexrelease>\DeclareFontSeriesChangeRule {lex}{bx} {} {}
1973 <latexrelease>\DeclareFontSeriesChangeRule {lex}{b} {} {}
1974 <latexrelease>\DeclareFontSeriesChangeRule {lex}{c} {} {}
1975 <latexrelease>\DeclareFontSeriesChangeRule {lex}{eb} {} {}
1976 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ec} {} {}
1977 <latexrelease>\DeclareFontSeriesChangeRule {lex}{el} {} {}
1978 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ex} {} {}
1979 <latexrelease>\DeclareFontSeriesChangeRule {lex}{l} {} {}
1980 <latexrelease>\DeclareFontSeriesChangeRule {lex}{m?} {} {}
1981 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sb} {} {}
1982 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sc} {} {}
1983 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sl} {} {}
1984 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sx} {} {}
1985 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ub} {} {}
1986 <latexrelease>\DeclareFontSeriesChangeRule {lex}{uc} {} {}
1987 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ul} {} {}
1988 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ux} {} {}
1989 <latexrelease>\DeclareFontSeriesChangeRule {lex}{x} {} {}
1990 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{bx} {} {}
1991 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{b} {} {}
1992 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{c} {} {}
1993 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{eb} {} {}
1994 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ec} {} {}
1995 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{el} {} {}
1996 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ex} {} {}
1997 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{l} {} {}
1998 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sb} {} {}
1999 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sc} {} {}
2000 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sl} {} {}
2001 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sx} {} {}
2002 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ub} {} {}
2003 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{uc} {} {}
2004 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ul} {} {}
2005 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ux} {} {}
2006 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{x} {} {}
2007 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{?m} {} {}
2008 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{bx} {} {}
2009 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{b} {} {}
2010 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{c} {} {}
2011 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{eb} {} {}
2012 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ec} {} {}
2013 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{el} {} {}
2014 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ex} {} {}
2015 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{l} {} {}

```

```

2016 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{m?} {} {}
2017 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sb} {} {}
2018 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sc} {} {}
2019 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sl} {} {}
2020 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sx} {} {}
2021 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ub} {} {}
2022 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{uc} {} {}
2023 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ul} {} {}
2024 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ux} {} {}
2025 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{x} {} {}
2026 <latexrelease>\DeclareFontSeriesChangeRule {luc}{?m} {} {}
2027 <latexrelease>\DeclareFontSeriesChangeRule {luc}{bx} {} {}
2028 <latexrelease>\DeclareFontSeriesChangeRule {luc}{b} {} {}
2029 <latexrelease>\DeclareFontSeriesChangeRule {luc}{c} {} {}
2030 <latexrelease>\DeclareFontSeriesChangeRule {luc}{eb} {} {}
2031 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ec} {} {}
2032 <latexrelease>\DeclareFontSeriesChangeRule {luc}{el} {} {}
2033 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ex} {} {}
2034 <latexrelease>\DeclareFontSeriesChangeRule {luc}{l} {} {}
2035 <latexrelease>\DeclareFontSeriesChangeRule {luc}{m?} {} {}
2036 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sb} {} {}
2037 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sc} {} {}
2038 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sl} {} {}
2039 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sx} {} {}
2040 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ub} {} {}
2041 <latexrelease>\DeclareFontSeriesChangeRule {luc}{uc} {} {}
2042 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ul} {} {}
2043 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ux} {} {}
2044 <latexrelease>\DeclareFontSeriesChangeRule {luc}{x} {} {}
2045 <latexrelease>\DeclareFontSeriesChangeRule {lux}{?m} {} {}
2046 <latexrelease>\DeclareFontSeriesChangeRule {lux}{bx} {} {}
2047 <latexrelease>\DeclareFontSeriesChangeRule {lux}{b} {} {}
2048 <latexrelease>\DeclareFontSeriesChangeRule {lux}{c} {} {}
2049 <latexrelease>\DeclareFontSeriesChangeRule {lux}{eb} {} {}
2050 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ec} {} {}
2051 <latexrelease>\DeclareFontSeriesChangeRule {lux}{el} {} {}
2052 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ex} {} {}
2053 <latexrelease>\DeclareFontSeriesChangeRule {lux}{l} {} {}
2054 <latexrelease>\DeclareFontSeriesChangeRule {lux}{m?} {} {}
2055 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sb} {} {}
2056 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sc} {} {}
2057 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sl} {} {}
2058 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sx} {} {}
2059 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ub} {} {}
2060 <latexrelease>\DeclareFontSeriesChangeRule {lux}{uc} {} {}
2061 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ul} {} {}
2062 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ux} {} {}
2063 <latexrelease>\DeclareFontSeriesChangeRule {lux}{x} {} {}
2064 <latexrelease>\DeclareFontSeriesChangeRule {lx}{bx} {} {}
2065 <latexrelease>\DeclareFontSeriesChangeRule {lx}{b}{bx}{}
2066 <latexrelease>\DeclareFontSeriesChangeRule {lx}{c}{lc}{}
2067 <latexrelease>\DeclareFontSeriesChangeRule {lx}{eb}{ebx}{}
2068 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ec}{lec}{}
2069 <latexrelease>\DeclareFontSeriesChangeRule {lx}{el}{elx}{}

```

```

2070 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ex} {} {}
2071 <latexrelease>\DeclareFontSeriesChangeRule {lx}{l}{lx}{}
2072 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sb}{sbx}{}
2073 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sc}{lsc}{}
2074 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sl}{slx}{}
2075 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sx} {} {}
2076 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ub}{ubx}{}
2077 <latexrelease>\DeclareFontSeriesChangeRule {lx}{uc} {} {}
2078 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ul}{ulx}{}
2079 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ux} {} {}
2080 <latexrelease>\DeclareFontSeriesChangeRule {lx}{x}{lx}{}
2081 <latexrelease>\DeclareFontSeriesChangeRule {l}{b} {b} {bx}
2082 <latexrelease>\DeclareFontSeriesChangeRule {l}{c} {l} {cl}
2083 <latexrelease>\DeclareFontSeriesChangeRule {l}{eb} {} {}
2084 <latexrelease>\DeclareFontSeriesChangeRule {l}{ec} {l} {lec} {l}
2085 <latexrelease>\DeclareFontSeriesChangeRule {l}{ex} {} {}
2086 <latexrelease>\DeclareFontSeriesChangeRule {l}{sb} {sb} {b}
2087 <latexrelease>\DeclareFontSeriesChangeRule {l}{sc} {l} {lsc} {l}
2088 <latexrelease>\DeclareFontSeriesChangeRule {l}{sx} {} {}
2089 <latexrelease>\DeclareFontSeriesChangeRule {l}{ub} {} {}
2090 <latexrelease>\DeclareFontSeriesChangeRule {l}{uc} {} {}
2091 <latexrelease>\DeclareFontSeriesChangeRule {l}{ux} {} {}
2092 <latexrelease>\DeclareFontSeriesChangeRule {l}{x} {l} {lx} {l}
2093 <latexrelease>\DeclareFontSeriesChangeRule {m}{bx} {bx} {b}
2094 <latexrelease>\DeclareFontSeriesChangeRule {m}{c} {c} {m}
2095 <latexrelease>\DeclareFontSeriesChangeRule {m}{l} {l} {m}
2096 <latexrelease>\DeclareFontSeriesChangeRule {m}{sc} {sc} {m}
2097 <latexrelease>\DeclareFontSeriesChangeRule {m}{x} {x} {m}
2098 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{bx} {} {}
2099 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{b}{bc}{}
2100 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{c}{sbc}{}
2101 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{eb}{ebc}{}
2102 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
2103 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{el}{elc}{}
2104 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ex} {} {}
2105 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{l}{lc}{}
2106 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sb}{sbc}{}
2107 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
2108 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sl}{slc}{}
2109 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sx} {} {}
2110 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ub}{ubc}{}
2111 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{uc} {} {}
2112 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ul}{ulc}{}
2113 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ux} {} {}
2114 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{x}{sbx}{}
2115 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{bx} {} {}
2116 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{b} {} {}
2117 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{c} {} {}
2118 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{eb} {} {}
2119 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ec} {} {}
2120 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{el} {} {}
2121 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ex} {} {}
2122 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{l} {} {}
2123 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{sb} {} {}

```

```

2124 \latexrelease\DeclareFontSeriesChangeRule {sbec}{sc} {} {}
2125 \latexrelease\DeclareFontSeriesChangeRule {sbec}{sl} {} {}
2126 \latexrelease\DeclareFontSeriesChangeRule {sbec}{sx} {} {}
2127 \latexrelease\DeclareFontSeriesChangeRule {sbec}{ub} {} {}
2128 \latexrelease\DeclareFontSeriesChangeRule {sbec}{uc} {} {}
2129 \latexrelease\DeclareFontSeriesChangeRule {sbec}{ul} {} {}
2130 \latexrelease\DeclareFontSeriesChangeRule {sbec}{ux} {} {}
2131 \latexrelease\DeclareFontSeriesChangeRule {sbec}{x} {} {}
2132 \latexrelease\DeclareFontSeriesChangeRule {sbex}{?m} {} {}
2133 \latexrelease\DeclareFontSeriesChangeRule {sbex}{bx} {} {}
2134 \latexrelease\DeclareFontSeriesChangeRule {sbex}{b} {} {}
2135 \latexrelease\DeclareFontSeriesChangeRule {sbex}{c} {} {}
2136 \latexrelease\DeclareFontSeriesChangeRule {sbex}{eb} {} {}
2137 \latexrelease\DeclareFontSeriesChangeRule {sbex}{ec} {} {}
2138 \latexrelease\DeclareFontSeriesChangeRule {sbex}{el} {} {}
2139 \latexrelease\DeclareFontSeriesChangeRule {sbex}{ex} {} {}
2140 \latexrelease\DeclareFontSeriesChangeRule {sbex}{l} {} {}
2141 \latexrelease\DeclareFontSeriesChangeRule {sbex}{m?} {} {}
2142 \latexrelease\DeclareFontSeriesChangeRule {sbex}{sb} {} {}
2143 \latexrelease\DeclareFontSeriesChangeRule {sbex}{sc} {} {}
2144 \latexrelease\DeclareFontSeriesChangeRule {sbex}{sl} {} {}
2145 \latexrelease\DeclareFontSeriesChangeRule {sbex}{sx} {} {}
2146 \latexrelease\DeclareFontSeriesChangeRule {sbex}{ub} {} {}
2147 \latexrelease\DeclareFontSeriesChangeRule {sbex}{uc} {} {}
2148 \latexrelease\DeclareFontSeriesChangeRule {sbex}{ul} {} {}
2149 \latexrelease\DeclareFontSeriesChangeRule {sbex}{ux} {} {}
2150 \latexrelease\DeclareFontSeriesChangeRule {sbex}{x} {} {}
2151 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{bx} {} {}
2152 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{b} {} {}
2153 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{c} {} {}
2154 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{eb} {} {}
2155 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{ec} {} {}
2156 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{el} {} {}
2157 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{ex} {} {}
2158 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{l} {} {}
2159 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{sb} {} {}
2160 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{sc} {} {}
2161 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{sl} {} {}
2162 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{sx} {} {}
2163 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{ub} {} {}
2164 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{uc} {} {}
2165 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{ul} {} {}
2166 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{ux} {} {}
2167 \latexrelease\DeclareFontSeriesChangeRule {sbsc}{x} {} {}
2168 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{?m} {} {}
2169 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{bx} {} {}
2170 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{b} {} {}
2171 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{c} {} {}
2172 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{eb} {} {}
2173 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{ec} {} {}
2174 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{el} {} {}
2175 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{ex} {} {}
2176 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{l} {} {}
2177 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{m?} {} {}

```

```

2178 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{sb} {} {}
2179 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{sc} {} {}
2180 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{sl} {} {}
2181 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{sx} {} {}
2182 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{ub} {} {}
2183 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{uc} {} {}
2184 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{ul} {} {}
2185 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{ux} {} {}
2186 \latexrelease\DeclareFontSeriesChangeRule {sbsx}{x} {} {}
2187 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{?m} {} {}
2188 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{bx} {} {}
2189 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{b} {} {}
2190 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{c} {} {}
2191 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{eb} {} {}
2192 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{ec} {} {}
2193 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{el} {} {}
2194 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{ex} {} {}
2195 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{l} {} {}
2196 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{m?} {} {}
2197 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{sb} {} {}
2198 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{sc} {} {}
2199 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{sl} {} {}
2200 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{sx} {} {}
2201 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{ub} {} {}
2202 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{uc} {} {}
2203 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{ul} {} {}
2204 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{ux} {} {}
2205 \latexrelease\DeclareFontSeriesChangeRule {sbuc}{x} {} {}
2206 \latexrelease\DeclareFontSeriesChangeRule {sbux}{?m} {} {}
2207 \latexrelease\DeclareFontSeriesChangeRule {sbux}{bx} {} {}
2208 \latexrelease\DeclareFontSeriesChangeRule {sbux}{b} {} {}
2209 \latexrelease\DeclareFontSeriesChangeRule {sbux}{c} {} {}
2210 \latexrelease\DeclareFontSeriesChangeRule {sbux}{eb} {} {}
2211 \latexrelease\DeclareFontSeriesChangeRule {sbux}{ec} {} {}
2212 \latexrelease\DeclareFontSeriesChangeRule {sbux}{el} {} {}
2213 \latexrelease\DeclareFontSeriesChangeRule {sbux}{ex} {} {}
2214 \latexrelease\DeclareFontSeriesChangeRule {sbux}{l} {} {}
2215 \latexrelease\DeclareFontSeriesChangeRule {sbux}{m?} {} {}
2216 \latexrelease\DeclareFontSeriesChangeRule {sbux}{sb} {} {}
2217 \latexrelease\DeclareFontSeriesChangeRule {sbux}{sc} {} {}
2218 \latexrelease\DeclareFontSeriesChangeRule {sbux}{sl} {} {}
2219 \latexrelease\DeclareFontSeriesChangeRule {sbux}{sx} {} {}
2220 \latexrelease\DeclareFontSeriesChangeRule {sbux}{ub} {} {}
2221 \latexrelease\DeclareFontSeriesChangeRule {sbux}{uc} {} {}
2222 \latexrelease\DeclareFontSeriesChangeRule {sbux}{ul} {} {}
2223 \latexrelease\DeclareFontSeriesChangeRule {sbux}{ux} {} {}
2224 \latexrelease\DeclareFontSeriesChangeRule {sbux}{x} {} {}
2225 \latexrelease\DeclareFontSeriesChangeRule {sbx}{bx} {} {}
2226 \latexrelease\DeclareFontSeriesChangeRule {sbx}{b}{bx}{}
2227 \latexrelease\DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}
2228 \latexrelease\DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}
2229 \latexrelease\DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}
2230 \latexrelease\DeclareFontSeriesChangeRule {sbx}{el}{elx}{}
2231 \latexrelease\DeclareFontSeriesChangeRule {sbx}{ex} {} {}

```



```

2232 \latexrelease\DeclareFontSeriesChangeRule {sbx}{l}{lx}{ }
2233 \latexrelease\DeclareFontSeriesChangeRule {sbx}{sb}{sbs}{ }
2234 \latexrelease\DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{ }
2235 \latexrelease\DeclareFontSeriesChangeRule {sbx}{sl}{slx}{ }
2236 \latexrelease\DeclareFontSeriesChangeRule {sbx}{sx}{ } { }
2237 \latexrelease\DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{ }
2238 \latexrelease\DeclareFontSeriesChangeRule {sbx}{uc}{ } { }
2239 \latexrelease\DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{ }
2240 \latexrelease\DeclareFontSeriesChangeRule {sbx}{ux}{ } { }
2241 \latexrelease\DeclareFontSeriesChangeRule {sbx}{x}{sbsx}{ }
2242 \latexrelease\DeclareFontSeriesChangeRule {sb}{bx}{ } { }
2243 \latexrelease\DeclareFontSeriesChangeRule {sb}{c}{sbc}{bc}
2244 \latexrelease\DeclareFontSeriesChangeRule {sb}{eb}{ } { }
2245 \latexrelease\DeclareFontSeriesChangeRule {sb}{ec}{sbec}{sbc}
2246 \latexrelease\DeclareFontSeriesChangeRule {sb}{ex}{ } { }
2247 \latexrelease\DeclareFontSeriesChangeRule {sb}{sc}{sbsc}{sbc}
2248 \latexrelease\DeclareFontSeriesChangeRule {sb}{sx}{ } { }
2249 \latexrelease\DeclareFontSeriesChangeRule {sb}{ub}{ } { }
2250 \latexrelease\DeclareFontSeriesChangeRule {sb}{uc}{ } { }
2251 \latexrelease\DeclareFontSeriesChangeRule {sb}{ux}{ } { }
2252 \latexrelease\DeclareFontSeriesChangeRule {sb}{x}{sbsx}{bx}
2253 \latexrelease\DeclareFontSeriesChangeRule {sc}{bx}{bx}{b}
2254 \latexrelease\DeclareFontSeriesChangeRule {sc}{b}{bsc}{ }
2255 \latexrelease\DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{ }
2256 \latexrelease\DeclareFontSeriesChangeRule {sc}{el}{elsc}{ }
2257 \latexrelease\DeclareFontSeriesChangeRule {sc}{l}{lsc}{ }
2258 \latexrelease\DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{ }
2259 \latexrelease\DeclareFontSeriesChangeRule {sc}{sl}{slsc}{ }
2260 \latexrelease\DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{ }
2261 \latexrelease\DeclareFontSeriesChangeRule {sc}{x}{x}{m}
2262 \latexrelease\DeclareFontSeriesChangeRule {slc}{bx}{ } { }
2263 \latexrelease\DeclareFontSeriesChangeRule {slc}{b}{bc}{ }
2264 \latexrelease\DeclareFontSeriesChangeRule {slc}{c}{slc}{ }
2265 \latexrelease\DeclareFontSeriesChangeRule {slc}{eb}{ebc}{ }
2266 \latexrelease\DeclareFontSeriesChangeRule {slc}{ec}{slec}{ }
2267 \latexrelease\DeclareFontSeriesChangeRule {slc}{el}{elc}{ }
2268 \latexrelease\DeclareFontSeriesChangeRule {slc}{ex}{ } { }
2269 \latexrelease\DeclareFontSeriesChangeRule {slc}{l}{lc}{ }
2270 \latexrelease\DeclareFontSeriesChangeRule {slc}{sb}{sbc}{ }
2271 \latexrelease\DeclareFontSeriesChangeRule {slc}{sc}{slsc}{ }
2272 \latexrelease\DeclareFontSeriesChangeRule {slc}{sl}{slc}{ }
2273 \latexrelease\DeclareFontSeriesChangeRule {slc}{sx}{ } { }
2274 \latexrelease\DeclareFontSeriesChangeRule {slc}{ub}{ubc}{ }
2275 \latexrelease\DeclareFontSeriesChangeRule {slc}{uc}{ } { }
2276 \latexrelease\DeclareFontSeriesChangeRule {slc}{ul}{ulc}{ }
2277 \latexrelease\DeclareFontSeriesChangeRule {slc}{ux}{ } { }
2278 \latexrelease\DeclareFontSeriesChangeRule {slc}{x}{slx}{ }
2279 \latexrelease\DeclareFontSeriesChangeRule {slec}{bx}{ } { }
2280 \latexrelease\DeclareFontSeriesChangeRule {slec}{b}{ } { }
2281 \latexrelease\DeclareFontSeriesChangeRule {slec}{c}{ } { }
2282 \latexrelease\DeclareFontSeriesChangeRule {slec}{eb}{ } { }
2283 \latexrelease\DeclareFontSeriesChangeRule {slec}{ec}{ } { }
2284 \latexrelease\DeclareFontSeriesChangeRule {slec}{el}{ } { }
2285 \latexrelease\DeclareFontSeriesChangeRule {slec}{ex}{ } { }

```

```

2286 \latexrelease\DeclareFontSeriesChangeRule {slec}{l} {} {}
2287 \latexrelease\DeclareFontSeriesChangeRule {slec}{sb} {} {}
2288 \latexrelease\DeclareFontSeriesChangeRule {slec}{sc} {} {}
2289 \latexrelease\DeclareFontSeriesChangeRule {slec}{sl} {} {}
2290 \latexrelease\DeclareFontSeriesChangeRule {slec}{sx} {} {}
2291 \latexrelease\DeclareFontSeriesChangeRule {slec}{ub} {} {}
2292 \latexrelease\DeclareFontSeriesChangeRule {slec}{uc} {} {}
2293 \latexrelease\DeclareFontSeriesChangeRule {slec}{ul} {} {}
2294 \latexrelease\DeclareFontSeriesChangeRule {slec}{ux} {} {}
2295 \latexrelease\DeclareFontSeriesChangeRule {slec}{x} {} {}
2296 \latexrelease\DeclareFontSeriesChangeRule {slex}{?m} {} {}
2297 \latexrelease\DeclareFontSeriesChangeRule {slex}{bx} {} {}
2298 \latexrelease\DeclareFontSeriesChangeRule {slex}{b} {} {}
2299 \latexrelease\DeclareFontSeriesChangeRule {slex}{c} {} {}
2300 \latexrelease\DeclareFontSeriesChangeRule {slex}{ec} {} {}
2301 \latexrelease\DeclareFontSeriesChangeRule {slex}{el} {} {}
2302 \latexrelease\DeclareFontSeriesChangeRule {slex}{ex} {} {}
2303 \latexrelease\DeclareFontSeriesChangeRule {slex}{l} {} {}
2304 \latexrelease\DeclareFontSeriesChangeRule {slex}{m?} {} {}
2305 \latexrelease\DeclareFontSeriesChangeRule {slex}{sb} {} {}
2306 \latexrelease\DeclareFontSeriesChangeRule {slex}{sc} {} {}
2307 \latexrelease\DeclareFontSeriesChangeRule {slex}{sl} {} {}
2308 \latexrelease\DeclareFontSeriesChangeRule {slex}{sx} {} {}
2309 \latexrelease\DeclareFontSeriesChangeRule {slex}{ub} {} {}
2310 \latexrelease\DeclareFontSeriesChangeRule {slex}{uc} {} {}
2311 \latexrelease\DeclareFontSeriesChangeRule {slex}{ul} {} {}
2312 \latexrelease\DeclareFontSeriesChangeRule {slex}{ux} {} {}
2313 \latexrelease\DeclareFontSeriesChangeRule {slex}{x} {} {}
2314 \latexrelease\DeclareFontSeriesChangeRule {slsc}{bx} {} {}
2315 \latexrelease\DeclareFontSeriesChangeRule {slsc}{b} {} {}
2316 \latexrelease\DeclareFontSeriesChangeRule {slsc}{c} {} {}
2317 \latexrelease\DeclareFontSeriesChangeRule {slsc}{eb} {} {}
2318 \latexrelease\DeclareFontSeriesChangeRule {slsc}{ec} {} {}
2319 \latexrelease\DeclareFontSeriesChangeRule {slsc}{el} {} {}
2320 \latexrelease\DeclareFontSeriesChangeRule {slsc}{ex} {} {}
2321 \latexrelease\DeclareFontSeriesChangeRule {slsc}{l} {} {}
2322 \latexrelease\DeclareFontSeriesChangeRule {slsc}{sb} {} {}
2323 \latexrelease\DeclareFontSeriesChangeRule {slsc}{sc} {} {}
2324 \latexrelease\DeclareFontSeriesChangeRule {slsc}{sl} {} {}
2325 \latexrelease\DeclareFontSeriesChangeRule {slsc}{sx} {} {}
2326 \latexrelease\DeclareFontSeriesChangeRule {slsc}{ub} {} {}
2327 \latexrelease\DeclareFontSeriesChangeRule {slsc}{uc} {} {}
2328 \latexrelease\DeclareFontSeriesChangeRule {slsc}{ul} {} {}
2329 \latexrelease\DeclareFontSeriesChangeRule {slsc}{ux} {} {}
2330 \latexrelease\DeclareFontSeriesChangeRule {slsc}{x} {} {}
2331 \latexrelease\DeclareFontSeriesChangeRule {slsx}{?m} {} {}
2332 \latexrelease\DeclareFontSeriesChangeRule {slsx}{bx} {} {}
2333 \latexrelease\DeclareFontSeriesChangeRule {slsx}{b} {} {}
2334 \latexrelease\DeclareFontSeriesChangeRule {slsx}{c} {} {}
2335 \latexrelease\DeclareFontSeriesChangeRule {slsx}{eb} {} {}
2336 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ec} {} {}
2337 \latexrelease\DeclareFontSeriesChangeRule {slsx}{el} {} {}
2338 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ex} {} {}
2339 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ex} {} {}

```

```

2340 \latexrelease\DeclareFontSeriesChangeRule {slsx}{l} {} {}
2341 \latexrelease\DeclareFontSeriesChangeRule {slsx}{m?} {} {}
2342 \latexrelease\DeclareFontSeriesChangeRule {slsx}{sb} {} {}
2343 \latexrelease\DeclareFontSeriesChangeRule {slsx}{sc} {} {}
2344 \latexrelease\DeclareFontSeriesChangeRule {slsx}{sl} {} {}
2345 \latexrelease\DeclareFontSeriesChangeRule {slsx}{sx} {} {}
2346 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ub} {} {}
2347 \latexrelease\DeclareFontSeriesChangeRule {slsx}{uc} {} {}
2348 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ul} {} {}
2349 \latexrelease\DeclareFontSeriesChangeRule {slsx}{ux} {} {}
2350 \latexrelease\DeclareFontSeriesChangeRule {slsx}{x} {} {}
2351 \latexrelease\DeclareFontSeriesChangeRule {sluc}{?m} {} {}
2352 \latexrelease\DeclareFontSeriesChangeRule {sluc}{bx} {} {}
2353 \latexrelease\DeclareFontSeriesChangeRule {sluc}{b} {} {}
2354 \latexrelease\DeclareFontSeriesChangeRule {sluc}{c} {} {}
2355 \latexrelease\DeclareFontSeriesChangeRule {sluc}{eb} {} {}
2356 \latexrelease\DeclareFontSeriesChangeRule {sluc}{ec} {} {}
2357 \latexrelease\DeclareFontSeriesChangeRule {sluc}{el} {} {}
2358 \latexrelease\DeclareFontSeriesChangeRule {sluc}{ex} {} {}
2359 \latexrelease\DeclareFontSeriesChangeRule {sluc}{l} {} {}
2360 \latexrelease\DeclareFontSeriesChangeRule {sluc}{m?} {} {}
2361 \latexrelease\DeclareFontSeriesChangeRule {sluc}{sb} {} {}
2362 \latexrelease\DeclareFontSeriesChangeRule {sluc}{sc} {} {}
2363 \latexrelease\DeclareFontSeriesChangeRule {sluc}{sl} {} {}
2364 \latexrelease\DeclareFontSeriesChangeRule {sluc}{sx} {} {}
2365 \latexrelease\DeclareFontSeriesChangeRule {sluc}{ub} {} {}
2366 \latexrelease\DeclareFontSeriesChangeRule {sluc}{uc} {} {}
2367 \latexrelease\DeclareFontSeriesChangeRule {sluc}{ul} {} {}
2368 \latexrelease\DeclareFontSeriesChangeRule {sluc}{ux} {} {}
2369 \latexrelease\DeclareFontSeriesChangeRule {sluc}{x} {} {}
2370 \latexrelease\DeclareFontSeriesChangeRule {slux}{?m} {} {}
2371 \latexrelease\DeclareFontSeriesChangeRule {slux}{bx} {} {}
2372 \latexrelease\DeclareFontSeriesChangeRule {slux}{b} {} {}
2373 \latexrelease\DeclareFontSeriesChangeRule {slux}{c} {} {}
2374 \latexrelease\DeclareFontSeriesChangeRule {slux}{eb} {} {}
2375 \latexrelease\DeclareFontSeriesChangeRule {slux}{ec} {} {}
2376 \latexrelease\DeclareFontSeriesChangeRule {slux}{el} {} {}
2377 \latexrelease\DeclareFontSeriesChangeRule {slux}{ex} {} {}
2378 \latexrelease\DeclareFontSeriesChangeRule {slux}{l} {} {}
2379 \latexrelease\DeclareFontSeriesChangeRule {slux}{m?} {} {}
2380 \latexrelease\DeclareFontSeriesChangeRule {slux}{sb} {} {}
2381 \latexrelease\DeclareFontSeriesChangeRule {slux}{sc} {} {}
2382 \latexrelease\DeclareFontSeriesChangeRule {slux}{sl} {} {}
2383 \latexrelease\DeclareFontSeriesChangeRule {slux}{sx} {} {}
2384 \latexrelease\DeclareFontSeriesChangeRule {slux}{ub} {} {}
2385 \latexrelease\DeclareFontSeriesChangeRule {slux}{uc} {} {}
2386 \latexrelease\DeclareFontSeriesChangeRule {slux}{ul} {} {}
2387 \latexrelease\DeclareFontSeriesChangeRule {slux}{ux} {} {}
2388 \latexrelease\DeclareFontSeriesChangeRule {slux}{x} {} {}
2389 \latexrelease\DeclareFontSeriesChangeRule {slx}{bx} {} {}
2390 \latexrelease\DeclareFontSeriesChangeRule {slx}{b}{bx}{}
2391 \latexrelease\DeclareFontSeriesChangeRule {slx}{c}{slc}{}
2392 \latexrelease\DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}
2393 \latexrelease\DeclareFontSeriesChangeRule {slx}{ec}{slec}{}

```

```

2394 \latexrelease\DeclareFontSeriesChangeRule {slx}{el}{elx}{}
2395 \latexrelease\DeclareFontSeriesChangeRule {slx}{ex} {} {}
2396 \latexrelease\DeclareFontSeriesChangeRule {slx}{l}{lx}{}
2397 \latexrelease\DeclareFontSeriesChangeRule {slx}{sb}{sbx}{}
2398 \latexrelease\DeclareFontSeriesChangeRule {slx}{sc}{slsc}{}
2399 \latexrelease\DeclareFontSeriesChangeRule {slx}{sl}{slx}{}
2400 \latexrelease\DeclareFontSeriesChangeRule {slx}{sx} {} {}
2401 \latexrelease\DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}
2402 \latexrelease\DeclareFontSeriesChangeRule {slx}{uc} {} {}
2403 \latexrelease\DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}
2404 \latexrelease\DeclareFontSeriesChangeRule {slx}{ux} {} {}
2405 \latexrelease\DeclareFontSeriesChangeRule {slx}{x}{slx}{}
2406 \latexrelease\DeclareFontSeriesChangeRule {sl}{bx} {} {}
2407 \latexrelease\DeclareFontSeriesChangeRule {sl}{c}{slc}{}
2408 \latexrelease\DeclareFontSeriesChangeRule {sl}{eb} {} {}
2409 \latexrelease\DeclareFontSeriesChangeRule {sl}{ec}{slec}{}
2410 \latexrelease\DeclareFontSeriesChangeRule {sl}{ex} {} {}
2411 \latexrelease\DeclareFontSeriesChangeRule {sl}{sb} {} {}
2412 \latexrelease\DeclareFontSeriesChangeRule {sl}{sc}{slsc}{}
2413 \latexrelease\DeclareFontSeriesChangeRule {sl}{sx} {} {}
2414 \latexrelease\DeclareFontSeriesChangeRule {sl}{ub} {} {}
2415 \latexrelease\DeclareFontSeriesChangeRule {sl}{uc} {} {}
2416 \latexrelease\DeclareFontSeriesChangeRule {sl}{ux} {} {}
2417 \latexrelease\DeclareFontSeriesChangeRule {sl}{x}{slx}{}
2418 \latexrelease\DeclareFontSeriesChangeRule {sx}{?m} {} {}
2419 \latexrelease\DeclareFontSeriesChangeRule {sx}{bx} {} {}
2420 \latexrelease\DeclareFontSeriesChangeRule {sx}{b} {} {}
2421 \latexrelease\DeclareFontSeriesChangeRule {sx}{eb} {} {}
2422 \latexrelease\DeclareFontSeriesChangeRule {sx}{el} {} {}
2423 \latexrelease\DeclareFontSeriesChangeRule {sx}{l} {} {}
2424 \latexrelease\DeclareFontSeriesChangeRule {sx}{m?} {} {}
2425 \latexrelease\DeclareFontSeriesChangeRule {sx}{sb} {} {}
2426 \latexrelease\DeclareFontSeriesChangeRule {sx}{sl} {} {}
2427 \latexrelease\DeclareFontSeriesChangeRule {sx}{ub} {} {}
2428 \latexrelease\DeclareFontSeriesChangeRule {sx}{ul} {} {}
2429 \latexrelease\DeclareFontSeriesChangeRule {ubc}{bx} {} {}
2430 \latexrelease\DeclareFontSeriesChangeRule {ubc}{b}{bc}{}
2431 \latexrelease\DeclareFontSeriesChangeRule {ubc}{c}{ubc}{}
2432 \latexrelease\DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}
2433 \latexrelease\DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}
2434 \latexrelease\DeclareFontSeriesChangeRule {ubc}{el}{elc}{}
2435 \latexrelease\DeclareFontSeriesChangeRule {ubc}{ex} {} {}
2436 \latexrelease\DeclareFontSeriesChangeRule {ubc}{l}{lc}{}
2437 \latexrelease\DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}
2438 \latexrelease\DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}
2439 \latexrelease\DeclareFontSeriesChangeRule {ubc}{sl}{slc}{}
2440 \latexrelease\DeclareFontSeriesChangeRule {ubc}{sx} {} {}
2441 \latexrelease\DeclareFontSeriesChangeRule {ubc}{ub}{ubc}{}
2442 \latexrelease\DeclareFontSeriesChangeRule {ubc}{uc} {} {}
2443 \latexrelease\DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
2444 \latexrelease\DeclareFontSeriesChangeRule {ubc}{ux} {} {}
2445 \latexrelease\DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}
2446 \latexrelease\DeclareFontSeriesChangeRule {ubec}{bx} {} {}
2447 \latexrelease\DeclareFontSeriesChangeRule {ubec}{b} {} {}

```

```

2448 \latexrelease\DeclareFontSeriesChangeRule {ubec}{c} {} {}
2449 \latexrelease\DeclareFontSeriesChangeRule {ubec}{eb} {} {}
2450 \latexrelease\DeclareFontSeriesChangeRule {ubec}{ec} {} {}
2451 \latexrelease\DeclareFontSeriesChangeRule {ubec}{el} {} {}
2452 \latexrelease\DeclareFontSeriesChangeRule {ubec}{ex} {} {}
2453 \latexrelease\DeclareFontSeriesChangeRule {ubec}{l} {} {}
2454 \latexrelease\DeclareFontSeriesChangeRule {ubec}{sb} {} {}
2455 \latexrelease\DeclareFontSeriesChangeRule {ubec}{sc} {} {}
2456 \latexrelease\DeclareFontSeriesChangeRule {ubec}{sl} {} {}
2457 \latexrelease\DeclareFontSeriesChangeRule {ubec}{sx} {} {}
2458 \latexrelease\DeclareFontSeriesChangeRule {ubec}{ub} {} {}
2459 \latexrelease\DeclareFontSeriesChangeRule {ubec}{uc} {} {}
2460 \latexrelease\DeclareFontSeriesChangeRule {ubec}{ul} {} {}
2461 \latexrelease\DeclareFontSeriesChangeRule {ubec}{ux} {} {}
2462 \latexrelease\DeclareFontSeriesChangeRule {ubec}{x} {} {}
2463 \latexrelease\DeclareFontSeriesChangeRule {ubex}{?m} {} {}
2464 \latexrelease\DeclareFontSeriesChangeRule {ubex}{bx} {} {}
2465 \latexrelease\DeclareFontSeriesChangeRule {ubex}{b} {} {}
2466 \latexrelease\DeclareFontSeriesChangeRule {ubex}{c} {} {}
2467 \latexrelease\DeclareFontSeriesChangeRule {ubex}{eb} {} {}
2468 \latexrelease\DeclareFontSeriesChangeRule {ubex}{ec} {} {}
2469 \latexrelease\DeclareFontSeriesChangeRule {ubex}{el} {} {}
2470 \latexrelease\DeclareFontSeriesChangeRule {ubex}{ex} {} {}
2471 \latexrelease\DeclareFontSeriesChangeRule {ubex}{l} {} {}
2472 \latexrelease\DeclareFontSeriesChangeRule {ubex}{m?} {} {}
2473 \latexrelease\DeclareFontSeriesChangeRule {ubex}{sb} {} {}
2474 \latexrelease\DeclareFontSeriesChangeRule {ubex}{sc} {} {}
2475 \latexrelease\DeclareFontSeriesChangeRule {ubex}{sl} {} {}
2476 \latexrelease\DeclareFontSeriesChangeRule {ubex}{sx} {} {}
2477 \latexrelease\DeclareFontSeriesChangeRule {ubex}{ub} {} {}
2478 \latexrelease\DeclareFontSeriesChangeRule {ubex}{uc} {} {}
2479 \latexrelease\DeclareFontSeriesChangeRule {ubex}{ul} {} {}
2480 \latexrelease\DeclareFontSeriesChangeRule {ubex}{ux} {} {}
2481 \latexrelease\DeclareFontSeriesChangeRule {ubex}{x} {} {}
2482 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{bx} {} {}
2483 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{b} {} {}
2484 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{c} {} {}
2485 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{eb} {} {}
2486 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{ec} {} {}
2487 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{el} {} {}
2488 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{ex} {} {}
2489 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{l} {} {}
2490 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{sb} {} {}
2491 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{sc} {} {}
2492 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{sl} {} {}
2493 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{sx} {} {}
2494 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{ub} {} {}
2495 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{uc} {} {}
2496 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{ul} {} {}
2497 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{ux} {} {}
2498 \latexrelease\DeclareFontSeriesChangeRule {ubsc}{x} {} {}
2499 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{?m} {} {}
2500 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{bx} {} {}
2501 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{b} {} {}

```

```

2502 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{c} {} {}
2503 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{eb} {} {}
2504 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{ec} {} {}
2505 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{el} {} {}
2506 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{ex} {} {}
2507 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{l} {} {}
2508 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{m?} {} {}
2509 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{sb} {} {}
2510 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{sc} {} {}
2511 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{sl} {} {}
2512 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{sx} {} {}
2513 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{ub} {} {}
2514 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{uc} {} {}
2515 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{ul} {} {}
2516 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{ux} {} {}
2517 \latexrelease\DeclareFontSeriesChangeRule {ubsx}{x} {} {}
2518 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{?m} {} {}
2519 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{bx} {} {}
2520 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{b} {} {}
2521 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{c} {} {}
2522 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{eb} {} {}
2523 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{ec} {} {}
2524 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{el} {} {}
2525 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{ex} {} {}
2526 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{l} {} {}
2527 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{m?} {} {}
2528 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{sb} {} {}
2529 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{sc} {} {}
2530 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{sl} {} {}
2531 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{sx} {} {}
2532 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{ub} {} {}
2533 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{uc} {} {}
2534 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{ul} {} {}
2535 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{ux} {} {}
2536 \latexrelease\DeclareFontSeriesChangeRule {ubuc}{x} {} {}
2537 \latexrelease\DeclareFontSeriesChangeRule {ubux}{?m} {} {}
2538 \latexrelease\DeclareFontSeriesChangeRule {ubux}{bx} {} {}
2539 \latexrelease\DeclareFontSeriesChangeRule {ubux}{b} {} {}
2540 \latexrelease\DeclareFontSeriesChangeRule {ubux}{c} {} {}
2541 \latexrelease\DeclareFontSeriesChangeRule {ubux}{eb} {} {}
2542 \latexrelease\DeclareFontSeriesChangeRule {ubux}{ec} {} {}
2543 \latexrelease\DeclareFontSeriesChangeRule {ubux}{el} {} {}
2544 \latexrelease\DeclareFontSeriesChangeRule {ubux}{ex} {} {}
2545 \latexrelease\DeclareFontSeriesChangeRule {ubux}{l} {} {}
2546 \latexrelease\DeclareFontSeriesChangeRule {ubux}{m?} {} {}
2547 \latexrelease\DeclareFontSeriesChangeRule {ubux}{sb} {} {}
2548 \latexrelease\DeclareFontSeriesChangeRule {ubux}{sc} {} {}
2549 \latexrelease\DeclareFontSeriesChangeRule {ubux}{sl} {} {}
2550 \latexrelease\DeclareFontSeriesChangeRule {ubux}{sx} {} {}
2551 \latexrelease\DeclareFontSeriesChangeRule {ubux}{ub} {} {}
2552 \latexrelease\DeclareFontSeriesChangeRule {ubux}{uc} {} {}
2553 \latexrelease\DeclareFontSeriesChangeRule {ubux}{ul} {} {}
2554 \latexrelease\DeclareFontSeriesChangeRule {ubux}{ux} {} {}
2555 \latexrelease\DeclareFontSeriesChangeRule {ubux}{x} {} {}

```

```

2556 \latexrelease\DeclareFontSeriesChangeRule {ubx}{bx} {} {}
2557 \latexrelease\DeclareFontSeriesChangeRule {ubx}{b}{bx}{ }
2558 \latexrelease\DeclareFontSeriesChangeRule {ubx}{c}{ubc}{ }
2559 \latexrelease\DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{ }
2560 \latexrelease\DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{ }
2561 \latexrelease\DeclareFontSeriesChangeRule {ubx}{el}{elx}{ }
2562 \latexrelease\DeclareFontSeriesChangeRule {ubx}{ex} {} {}
2563 \latexrelease\DeclareFontSeriesChangeRule {ubx}{l}{lx}{ }
2564 \latexrelease\DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{ }
2565 \latexrelease\DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{ }
2566 \latexrelease\DeclareFontSeriesChangeRule {ubx}{sl}{slx}{ }
2567 \latexrelease\DeclareFontSeriesChangeRule {ubx}{sx} {} {}
2568 \latexrelease\DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{ }
2569 \latexrelease\DeclareFontSeriesChangeRule {ubx}{uc} {} {}
2570 \latexrelease\DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{ }
2571 \latexrelease\DeclareFontSeriesChangeRule {ubx}{ux} {} {}
2572 \latexrelease\DeclareFontSeriesChangeRule {ubx}{x}{ubx}{ }
2573 \latexrelease\DeclareFontSeriesChangeRule {ub}{bx} {} {}
2574 \latexrelease\DeclareFontSeriesChangeRule {ub}{c}{ubc}{ }
2575 \latexrelease\DeclareFontSeriesChangeRule {ub}{eb} {} {}
2576 \latexrelease\DeclareFontSeriesChangeRule {ub}{ec}{ubec}{ }
2577 \latexrelease\DeclareFontSeriesChangeRule {ub}{ex} {} {}
2578 \latexrelease\DeclareFontSeriesChangeRule {ub}{sb} {} {}
2579 \latexrelease\DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{ }
2580 \latexrelease\DeclareFontSeriesChangeRule {ub}{sx} {} {}
2581 \latexrelease\DeclareFontSeriesChangeRule {ub}{uc} {} {}
2582 \latexrelease\DeclareFontSeriesChangeRule {ub}{ux} {} {}
2583 \latexrelease\DeclareFontSeriesChangeRule {ub}{x}{ubx}{ }
2584 \latexrelease\DeclareFontSeriesChangeRule {uc}{?m} {} {}
2585 \latexrelease\DeclareFontSeriesChangeRule {uc}{bx} {} {}
2586 \latexrelease\DeclareFontSeriesChangeRule {uc}{b} {} {}
2587 \latexrelease\DeclareFontSeriesChangeRule {uc}{eb} {} {}
2588 \latexrelease\DeclareFontSeriesChangeRule {uc}{el} {} {}
2589 \latexrelease\DeclareFontSeriesChangeRule {uc}{l} {} {}
2590 \latexrelease\DeclareFontSeriesChangeRule {uc}{m?} {} {}
2591 \latexrelease\DeclareFontSeriesChangeRule {uc}{sb} {} {}
2592 \latexrelease\DeclareFontSeriesChangeRule {uc}{sl} {} {}
2593 \latexrelease\DeclareFontSeriesChangeRule {uc}{ub} {} {}
2594 \latexrelease\DeclareFontSeriesChangeRule {uc}{ul} {} {}
2595 \latexrelease\DeclareFontSeriesChangeRule {ulc}{bx} {} {}
2596 \latexrelease\DeclareFontSeriesChangeRule {ulc}{b}{bc}{ }
2597 \latexrelease\DeclareFontSeriesChangeRule {ulc}{c}{ulc}{ }
2598 \latexrelease\DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{ }
2599 \latexrelease\DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
2600 \latexrelease\DeclareFontSeriesChangeRule {ulc}{el}{elc}{ }
2601 \latexrelease\DeclareFontSeriesChangeRule {ulc}{ex} {} {}
2602 \latexrelease\DeclareFontSeriesChangeRule {ulc}{l}{lc}{ }
2603 \latexrelease\DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{ }
2604 \latexrelease\DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
2605 \latexrelease\DeclareFontSeriesChangeRule {ulc}{sl}{slc}{ }
2606 \latexrelease\DeclareFontSeriesChangeRule {ulc}{sx} {} {}
2607 \latexrelease\DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{ }
2608 \latexrelease\DeclareFontSeriesChangeRule {ulc}{uc} {} {}
2609 \latexrelease\DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{ }

```

```

2610 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ux} {} {}
2611 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}
2612 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{bx} {} {}
2613 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{b} {} {}
2614 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{c} {} {}
2615 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{eb} {} {}
2616 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ec} {} {}
2617 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{el} {} {}
2618 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ex} {} {}
2619 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{l} {} {}
2620 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sb} {} {}
2621 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sc} {} {}
2622 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sl} {} {}
2623 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sx} {} {}
2624 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ub} {} {}
2625 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{uc} {} {}
2626 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ul} {} {}
2627 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ux} {} {}
2628 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{x} {} {}
2629 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{?m} {} {}
2630 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{bx} {} {}
2631 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{b} {} {}
2632 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{c} {} {}
2633 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{eb} {} {}
2634 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ec} {} {}
2635 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{el} {} {}
2636 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ex} {} {}
2637 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{l} {} {}
2638 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{m?} {} {}
2639 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sb} {} {}
2640 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sc} {} {}
2641 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sl} {} {}
2642 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sx} {} {}
2643 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ub} {} {}
2644 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{uc} {} {}
2645 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ul} {} {}
2646 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ux} {} {}
2647 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{x} {} {}
2648 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{bx} {} {}
2649 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{b} {} {}
2650 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{c} {} {}
2651 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{eb} {} {}
2652 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ec} {} {}
2653 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{el} {} {}
2654 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ex} {} {}
2655 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{l} {} {}
2656 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sb} {} {}
2657 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sc} {} {}
2658 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sl} {} {}
2659 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sx} {} {}
2660 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ub} {} {}
2661 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{uc} {} {}
2662 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ul} {} {}
2663 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ux} {} {}

```



```

2664 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{x} {} {}
2665 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{?m} {} {}
2666 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{bx} {} {}
2667 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{b} {} {}
2668 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{c} {} {}
2669 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{eb} {} {}
2670 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ec} {} {}
2671 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{el} {} {}
2672 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ex} {} {}
2673 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{l} {} {}
2674 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{m?} {} {}
2675 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sb} {} {}
2676 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sc} {} {}
2677 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sl} {} {}
2678 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sx} {} {}
2679 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ub} {} {}
2680 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{uc} {} {}
2681 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ul} {} {}
2682 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ux} {} {}
2683 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{x} {} {}
2684 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{?m} {} {}
2685 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{bx} {} {}
2686 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{b} {} {}
2687 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{c} {} {}
2688 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{eb} {} {}
2689 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ec} {} {}
2690 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{el} {} {}
2691 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ex} {} {}
2692 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{l} {} {}
2693 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{m?} {} {}
2694 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sb} {} {}
2695 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sc} {} {}
2696 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sl} {} {}
2697 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sx} {} {}
2698 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ub} {} {}
2699 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{uc} {} {}
2700 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ul} {} {}
2701 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ux} {} {}
2702 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{x} {} {}
2703 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{?m} {} {}
2704 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{bx} {} {}
2705 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{b} {} {}
2706 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{c} {} {}
2707 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{eb} {} {}
2708 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ec} {} {}
2709 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{el} {} {}
2710 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ex} {} {}
2711 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{l} {} {}
2712 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{m?} {} {}
2713 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sb} {} {}
2714 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sc} {} {}
2715 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sl} {} {}
2716 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sx} {} {}
2717 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ub} {} {}

```

```

2718 \latexrelease\DeclareFontSeriesChangeRule {ulux}{uc} {} {}
2719 \latexrelease\DeclareFontSeriesChangeRule {ulux}{ul} {} {}
2720 \latexrelease\DeclareFontSeriesChangeRule {ulux}{ux} {} {}
2721 \latexrelease\DeclareFontSeriesChangeRule {ulux}{x} {} {}
2722 \latexrelease\DeclareFontSeriesChangeRule {ulx}{bx} {} {}
2723 \latexrelease\DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
2724 \latexrelease\DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
2725 \latexrelease\DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
2726 \latexrelease\DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
2727 \latexrelease\DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
2728 \latexrelease\DeclareFontSeriesChangeRule {ulx}{ex} {} {}
2729 \latexrelease\DeclareFontSeriesChangeRule {ulx}{l}{lx}{}
2730 \latexrelease\DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
2731 \latexrelease\DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
2732 \latexrelease\DeclareFontSeriesChangeRule {ulx}{sl}{slx}{}
2733 \latexrelease\DeclareFontSeriesChangeRule {ulx}{sx} {} {}
2734 \latexrelease\DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
2735 \latexrelease\DeclareFontSeriesChangeRule {ulx}{uc} {} {}
2736 \latexrelease\DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
2737 \latexrelease\DeclareFontSeriesChangeRule {ulx}{ux} {} {}
2738 \latexrelease\DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}
2739 \latexrelease\DeclareFontSeriesChangeRule {ul}{bx} {} {}
2740 \latexrelease\DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
2741 \latexrelease\DeclareFontSeriesChangeRule {ul}{eb} {} {}
2742 \latexrelease\DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
2743 \latexrelease\DeclareFontSeriesChangeRule {ul}{ex} {} {}
2744 \latexrelease\DeclareFontSeriesChangeRule {ul}{sb} {} {}
2745 \latexrelease\DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
2746 \latexrelease\DeclareFontSeriesChangeRule {ul}{sx} {} {}
2747 \latexrelease\DeclareFontSeriesChangeRule {ul}{ub} {} {}
2748 \latexrelease\DeclareFontSeriesChangeRule {ul}{uc} {} {}
2749 \latexrelease\DeclareFontSeriesChangeRule {ul}{ux} {} {}
2750 \latexrelease\DeclareFontSeriesChangeRule {ul}{x}{ulx}{}
2751 \latexrelease\DeclareFontSeriesChangeRule {ux}{?m} {} {}
2752 \latexrelease\DeclareFontSeriesChangeRule {ux}{bx} {} {}
2753 \latexrelease\DeclareFontSeriesChangeRule {ux}{b} {} {}
2754 \latexrelease\DeclareFontSeriesChangeRule {ux}{eb} {} {}
2755 \latexrelease\DeclareFontSeriesChangeRule {ux}{el} {} {}
2756 \latexrelease\DeclareFontSeriesChangeRule {ux}{l} {} {}
2757 \latexrelease\DeclareFontSeriesChangeRule {ux}{m?} {} {}
2758 \latexrelease\DeclareFontSeriesChangeRule {ux}{sb} {} {}
2759 \latexrelease\DeclareFontSeriesChangeRule {ux}{sl} {} {}
2760 \latexrelease\DeclareFontSeriesChangeRule {ux}{ub} {} {}
2761 \latexrelease\DeclareFontSeriesChangeRule {ux}{ul} {} {}
2762 \latexrelease\DeclareFontSeriesChangeRule {x}{bx} {} {}
2763 \latexrelease\DeclareFontSeriesChangeRule {x}{c}{c}{}
2764 \latexrelease\DeclareFontSeriesChangeRule {x}{ec}{ec}{}
2765 \latexrelease\DeclareFontSeriesChangeRule {x}{el}{elx}{}
2766 \latexrelease\DeclareFontSeriesChangeRule {x}{l}{lx}{}
2767 \latexrelease\DeclareFontSeriesChangeRule {x}{sc}{sc}{}
2768 \latexrelease\DeclareFontSeriesChangeRule {x}{sl}{slx}{}
2769 \latexrelease\DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
2770 \latexrelease\DeclareFontSeriesChangeRule {x}{ul}{ulx}{}
2771 \latexrelease

```

```

2772 <latexrelease>\EndIncludeInRelease
2773 <latexrelease>\IncludeInRelease{0000/00/00}%
2774 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
2775 <latexrelease>
2776 <latexrelease>\let\DeclareFontSeriesChangeRule\@undefined
2777 <latexrelease>
2778 <latexrelease>\EndIncludeInRelease

```

1.3 Changing to a new series

```

2779 <*2ekernel | latexrelease>
2780 <latexrelease>\IncludeInRelease{2021/06/01}%
2781 <latexrelease> {\fontseries}{delay fontseries update}%

```

\fontseries The `\fontseries` command takes one argument which is the requested new font series. In the original implementation it simply saved the expanded value in `\f@series`. Now we do a bit more processing and look up the final value in the font series data base. This is done by `\merge@font@series`. But the lookup should be done within the target family and call to `\fontseries` might be followed by a `\fontfamily` call. So we delay the processing to `\selectfont` and only record the necessary action in `\delayed@f@adjustment`.

```

2782 \DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
2783   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
2784     {\delayed@f@adjustment\delayed@merge@font@series{#1}}}

```

(End of definition for \fontseries.)

\delayed@f@adjustment The macro holding the delayed action(s) for use in `\selectfont`.

```

2785 \let\delayed@f@adjustment\@empty

```

(End of definition for \delayed@f@adjustment.)

\fontseriesforce To change unconditionally to a new series you can use `\fontseriesforce`. Of course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```

2786 \DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
2787   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
2788     {\delayed@f@adjustment\edef\f@series{#1}}}

```

(End of definition for \fontseriesforce.)

\if@forced@series If the series gets forced we need to know that fact later on.

```

2789 \newif\if@forced@series

```

(End of definition for \if@forced@series.)

```

2790 </2ekernel | latexrelease>
2791 <latexrelease>\EndIncludeInRelease
2792 <latexrelease>\IncludeInRelease{2020/02/02}%
2793 <latexrelease> {\fontseries}{delay fontseries update}%
2794 <latexrelease>
2795 <latexrelease>\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
2796 <latexrelease> \merge@font@series{#1}}
2797 <latexrelease>\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
2798 <latexrelease> \edef\f@series{#1}}
2799 <latexrelease>\let\delayed@f@adjustment\@undefined
2800 <latexrelease>

```

For a roll forward we may have to define `\if@forced@series` but this needs doing in a way that $\TeX$ doesn't see it when skipping over conditionals.

```

2801 <latexrelease>\expandafter\newif\csname if@forced@series\endcsname
2802 <latexrelease>
2803 <latexrelease>\EndIncludeInRelease
2804 <latexrelease>\IncludeInRelease{0000/00/00}%
2805 <latexrelease>                {\fontseries}{delay fontseries update}%
2806 <latexrelease>
2807 <latexrelease>\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
2808 <latexrelease>\let\fontseriesforce\@undefined
2809 <latexrelease>
2810 <latexrelease>\EndIncludeInRelease
2811 <*2ekernel | latexrelease>
2812 <latexrelease>\IncludeInRelease{2020/02/02}%
2813 <latexrelease>    {\merge@font@series}{Merge series values}%

```

`\merge@font@series` We look up the data base value by expanding the right command twice. If no such value exist then the result will be `\relax` otherwise it will be the two brace groups: the desired result and the alternate result. The first case means that the third argument to `\merge@font@series` will be empty.

```

2814 \def\merge@font@series#1{%
2815   \expandafter\expandafter\expandafter
2816   \merge@font@series@
2817   \csname series@\f@series @#1\endcsname
2818   {#1}%
2819   \@nil
2820 }

```

(End of definition for \merge@font@series.)

`\merge@font@series@` This now defines the new `\f@series`:

```

2821 \def\merge@font@series@#1#2#3\@nil{%

```

If the third argument is empty there is no database entry for the combination and the second argument holds the new series so we return that.

Originally the test was simply `\ifx!#3!` but that actually dies if `#3` starts with a conditional and in the definition of `\AmSfont` that is actually the case.

```

2822 %\ifcat\expandafter X\detokenize{#1}X%
2823 \def\reserved@a{#3}%
2824 \ifx\reserved@a\@empty
2825   \set@target@series{#2}%
2826 \else

```

Otherwise we check if the desired result for the series (`#1`) exists for the font family and the current shape. All this happens inside `\selectfont` which has already taken care to load the `.fd`, file if necessary.

```

2827 \edef\reserved@a{\f@encoding /\f@family /#1/\f@shape}%
2828 \ifcsname \reserved@a \endcsname

```

If the desired result is available then we use that. However, we do need some post-processing because we need to drop surplus `ms` due to the way naming convention was designed in the '90s (sigh).

```

2829 \set@target@series{#1}%

```

If not, then we try the alternate result (#2).

```
2830     \else
2831     \ifcsname \f@encoding /\f@family /#2/\f@shape \endcsname
```

If the alternate result exist we use that and also issue a warning (or rather a log entry) that we didn't managed to change to the desired font.

```
2832     \set@target@series{#2}%
2833     \@font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
2834     \else
2835     \set@target@series{#3}%
2836     \@font@shape@subst@warning
2837   \fi
2838 \fi
2839 \fi
2840 }
```

It is possible that the previous font and the new one are actually identical (and the font was not found because it still needs loading) in which case a warning would look rather odd. So we make a quick check for that (which is the reason why we defined `\@reserveda` above instead of doing inline testing inside `\ifcsname`).

```
2841 \def\@font@shape@subst@warning{%
2842   \edef\reserved@b{\curr@fontshape}%
2843   \ifx\reserved@a\reserved@b \else
2844     \@font@warning{Font shape '\reserved@a' undefined\MessageBreak
2845                   using '\reserved@b' instead}%
2846   \fi
2847 }
```

(End of definition for `\merge@font@series@`.)

`\merge@font@series@without@substitution`
`\merge@font@series@without@substitution@`
`\delayed@merge@font@series`

`\merge@font@series@without@substitution` works like `\merge@font@series`, i.e., it looks up the combination in the rule base and if there exists an entry it uses it and if not it uses the new series value. However, it doesn't check if there is actually a font face with the new series value as `\merge@font@series` does. This simplified command is used in `\selectfont` at a point where other font attributes are not yet updated so that checking the font face might result incorrect in substitutions.

```
2848 \def\merge@font@series@without@substitution#1{%
2849   \expandafter\expandafter\expandafter
2850   \merge@font@series@without@substitution@
2851   \csname series@\f@series @#1\endcsname
2852   {#1}%
2853   \@nil
2854 }
2855 \def\merge@font@series@without@substitution@#1#2#3\@nil{%
2856   \def\reserved@a{#3}%
2857   \ifx\reserved@a\@empty
2858     \set@target@series{#2}%
2859   \else
2860     \set@target@series{#1}%
2861   \fi
2862 }
```

(End of definition for `\merge@font@series@without@substitution`,
`\merge@font@series@without@substitution@`, and `\delayed@merge@font@series`.)

`\delayed@merge@font@series` When we delay the merge action in `\fontseries` we first attempt to use merging without substitution. If that results in a non-existing font face the merge is redone in `\selectfont` using a version with substitution. See `\selectfont` for details.

```
2863 \let\delayed@merge@font@series\merge@font@series@without@substitution
```

(End of definition for `\delayed@merge@font@series`.)

`\maybe@load@fontshape` A small helper that we use a couple of times: try loading a fontshape (in a group because `\try@load@fontshape` normalizes catcodes and we also want to change `\typeout` so that it doesn't report missing .fd files on the terminal).

```
2864 \def\maybe@load@fontshape{%
2865   \begingroup
2866   \let \typeout \@font@info
2867   \try@load@fontshape
2868   \endgroup}
```

(End of definition for `\maybe@load@fontshape`.)

`\set@target@series` Finally the code for normalizing the `\f@series` value.

The combined series value determined by the mapping may still contain an `m` that we have to remove (as the .fd files use `c` not `mc` to denote a medium weight condensed series, etc.). We do this in all branches above because a user might have written

```
\DeclareFontSeriesChangeRule {m}{sc}{msc}{mc}
```

instead of using `sc` and `c` as needed in the .fd file.

```
2869 \def\set@target@series#1{%
```

We need to `\edef` the argument first in case it starts with a conditional. Then we check (and perhaps drop) an “m” from the value and assign the result to `\f@series`.

```
2870   \edef\f@series{#1}%
2871   \series@maybe@drop@one@m\f@series\f@series
2872 }
```

(End of definition for `\set@target@series`.)

`\series@maybe@drop@one@m` If the series value is in NFSS notation then it should not contain any “m” unless it is just an “m” by its own. So we need to drop surplus “m”s. But we better don't do this for full names, such as “semibold” as used by `autoinst`, for example. So we test against the possible explicit values that should drop an “m”. After that we assign the result to `#2` for further use.

```
2873 \def\series@maybe@drop@one@m#1{%
2874   \expandafter\series@maybe@drop@one@m@x\expandafter{#1}}
2875
2876 \def\series@maybe@drop@one@m@x#1#2{%
```

The code below is an inline version of the `\in@` macro without the group, so that it works in `\accent`.

```
2877   \def\in@@##1,#1,{}%
2878   \series@check@toks\expandafter{\in@@
2879     ,ulm,elm,lm,slm,mm,slm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{},{},#1,}%
2880   \edef\in@@{\the\series@check@toks}%
2881   \ifx\in@@\@empty
```

The default definition for `\bfdefault` etc is actually `b\@empty` so that we can detect if the user has changed the default. However that means a) the above test will definitely fail (maybe something to change) and b) we better use `\edef` on the next line to get rid of it as otherwise the test against `#2` (e.g., `\bfdef@ult`) will fail in other places.

```
2882 \edef#2{#1}%
2883 \else
2884 \edef#2{\expandafter\series@drop@one@m #1m\series@drop@one@m}%
2885 \fi
2886 }
```

As a precaution we use a private toks register not `\toks@` as that is no longer hidden inside the group.

```
2887 \newtoks\series@check@toks
```

(End of definition for \series@maybe@drop@one@m.)

`\series@drop@one@m` Drop up to two ms but keep one if that makes the series value empty. Actually, with the current implementation we know that there is at least one in the series value itself and we added one after it, so all we have to do is now returning `#1#2` and dropping the rest.

```
2888 \def\series@drop@one@m#1m#2m#3\series@drop@one@m{%
2889 % \ifx\relax#1#2\relax m\else#1#2\fi
2890 #1#2%
2891 }
```

(End of definition for \series@drop@one@m.)

Supporting rollback ...

```
2892 </2ekernel | latexrelease>
2893 <latexrelease>\EndIncludeInRelease
2894 <latexrelease>\IncludeInRelease{0000/00/00}%
2895 <latexrelease> {\merge@font@series}{Merge series values}%
2896 <latexrelease>
2897 <latexrelease>\let\merge@font@series\@undefined
2898 <latexrelease>\let\merge@font@series@\@undefined
2899 <latexrelease>\let\@font@shape@subst@warning\@undefined
2900 <latexrelease>\let\merge@font@series@without@substitution\@undefined
2901 <latexrelease>\let\merge@font@series@without@substitution@\@undefined
2902 <latexrelease>\let\delayed@merge@font@series\@undefined
2903 <latexrelease>\let\maybe@load@fontshape\@undefined
2904 <latexrelease>\let\set@target@series\@undefined
2905 <latexrelease>\let\series@maybe@drop@one@m\@undefined
2906 <latexrelease>\let\series@drop@one@m\@undefined
2907 <latexrelease>
2908 <latexrelease>\EndIncludeInRelease
```

2 Changing the shape

Shapes are also split in two axes (though it could be more if that is desirable), essentially building in an “sc” axis.

```
2909 <*2ekernel | latexrelease>
2910 <latexrelease>\IncludeInRelease{2020/02/02}%
2911 <latexrelease> {\ulcshape}{Font shape change rules}%
```

`\DeclareFontShapeChangeRule` The database for shapes is done in exactly the same way, only that it is much smaller and we usually have no alternative shape (or rather it is empty thus not used).

```
2912 \def\DeclareFontShapeChangeRule #1#2#3#4{%
2913   \@namedef{shape@#1@#2}{#{#3}#{#4}}}
```

(End of definition for \DeclareFontShapeChangeRule.)

There is kind of the same problem with returning back from `sc` to normal. It sort of needs its own letter. In `fontspec` this was solved by the first time `\upshape` changes `it` or `sl` back (so only `sc` remains) and second time it changes then `sc` back to normal. Maybe that's not a bad way to handle it, but decided for a slightly different approach: `n` always returns to "normal", ie resets everything and `up` changes italic or slanted to upright and `ulc` undoes small caps.

So we now offer `\normalshape` (using `\shapedefault`) which is normally the same as calling both `\ulcshape` and `\upshape`, only more efficient.

`\ulcshape` To request going back to upper/lowercase we need a new command. It uses `ulc` as shape name but this shape is virtual, i.e., it doesn't exist as a real shape, it is only used as part of the database table entries and thus only appears in the second argument there (but not in the first).

```
2914 \DeclareRobustCommand\ulcshape
2915   {\not@math@alphabet\ulcshape\relax
2916    \fontshape\ulcdefault\selectfont}
2917 \let\ulcdefault\@undefined      % for rollback
2918 \newcommand\ulcdefault{ulc}
```

(End of definition for \ulcshape, \textulc, and \ulcdefault.)

`\swshape` New command to select a swash shape. The standard rules put this in the same category as italics or slanted, i.e., if you ask for it then italics are undone. One could provide more complicated rules so that `it + sw` becomes `swit` but given that there are only very few fonts that have swash letters that level of flexibility (these days) would be just resulting in a lot of combinations that do not exist.

```
2919 \DeclareRobustCommand\swshape
2920   {\not@math@alphabet\swshape\relax
2921    \fontshape\swdefault\selectfont}
2922 \let\swdefault\@undefined      % for rollback
2923 \newcommand\swdefault{sw}
```

(End of definition for \swshape, \textsw, and \swdefault.)

`\sscshape` New commands to select spaced small capitals. There isn't a single free font that supports it. However, some commercial ones do, so we offer it. It is also possible to produce spaced small capitals from normal small capitals in OTF fonts using `otftotfm` with calls such as

```
otftotfm -e TEXMF/fonts/enc/dvips/base/txnansx.enc \
SourceSerifPro-Regular.otf -fkern -fliga --feature=smcp \
--letterspacing=80 SourceSerifPro-Regular-ssc-LY1
```

Michael Ummels kindly prepared the necessary rules for `ssc` ages ago, but until recently they managed to hide deep down in my inbox. I finally got around to integrating them (with a few changes) and I also took the opportunity to rationalize (a bit) the rules for the only other uncommon shape, `sw`.


```

2924 \DeclareRobustCommand\sscshape
2925         {\not@math@alphabet\sscshape\relax
2926          \fontshape\sscdefault\selectfont}
2927 \let\sscdefault\@undefined      % for rollback
2928 \newcommand\sscdefault{ssc}

(End of definition for \sscshape, \textssc, and \sscdefault.)
    Supporting rollback ...

2929 </2ekernel | latexrelease>
2930 <latexrelease>\EndIncludeInRelease
2931 <latexrelease>\IncludeInRelease{0000/00/00}%
2932 <latexrelease>    {\ulcshape}{Font shape change rules}%
2933 <latexrelease>
2934 <latexrelease>\let\DeclareFontShapeChangeRule\@undefined
2935 <latexrelease>\let\ulcshape\@undefined
2936 <latexrelease>\let\ulcdefault\@undefined
2937 <latexrelease>\let\swshape\@undefined
2938 <latexrelease>\let\swdefault\@undefined
2939 <latexrelease>\let\sscshape\@undefined
2940 <latexrelease>\let\sscdefault\@undefined
2941 <latexrelease>
2942 <latexrelease>\EndIncludeInRelease
2943 <*2ekernel>

```

2.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```

2944 </2ekernel>
2945 <*2ekernel | latexrelease>
2946 <latexrelease>\IncludeInRelease{2025/06/01}%
2947 <latexrelease>    {\DeclareFontShapeChangeRule}{Rules for ssc and sw}%
2948 %
2949 % \DeclareFontShapeChangeRule {n}{n}    {n}    {}
2950 \DeclareFontShapeChangeRule {n}{it}    {it}    {sl}
2951 \DeclareFontShapeChangeRule {n}{sl}    {sl}    {it}

```

If **sw** is requested but not available (not many font families offer it) we try to fallback to **it** instead of **sw**. That isn't always perfect, because some swash shapes are actually upright, but it is only a fallback and most of the time it would be better than **n**.

```

2952 \DeclareFontShapeChangeRule {n}{sw}    {sw}    {it}
2953 % \DeclareFontShapeChangeRule {n}{sc}    {sc}    {}
2954 \DeclareFontShapeChangeRule {n}{ulc}    {n}    {}
2955 \DeclareFontShapeChangeRule {n}{up}    {n}    {}

```

For the **ssc** shape we make the following general assumptions: if **ssc** $\langle X \rangle$ exists then **sc** $\langle X \rangle$ and $\langle X \rangle$ also exist in the font. If the **ssc** $\langle X \rangle$ shape doesn't exist but the user had asked for **ssc** we try to replace it by **sc** because we then assume that the current font family simply doesn't have any **ssc** shapes. However, if we are already in some **ssc** shape and a shape change is requested we know that at least some **ssc** shapes exist for the current font family, so rather than falling back to some **sc** shape we try to stay within **ssc** shapes in a fallback situation.

```

2956 \DeclareFontShapeChangeRule {n}{ssc}    {ssc}    {sc}

```

```

2957 % \DeclareFontShapeChangeRule {it}{n} {n} {}
2958 % \DeclareFontShapeChangeRule {it}{it} {it} {}
2959 \DeclareFontShapeChangeRule {it}{sl} {sl} {it}
2960 \DeclareFontShapeChangeRule {it}{sw} {sw} {it}

```

If neither `scit` nor `scsl` exist then `sc` will be used as a fallback albeit with a log entry, so except for the latter there will be no change for CM or Latin Modern fonts.

```

2961 \DeclareFontShapeChangeRule {it}{sc} {scit} {scsl}
2962 \DeclareFontShapeChangeRule {it}{ulc} {it} {}
2963 \DeclareFontShapeChangeRule {it}{up} {n} {}
2964 \DeclareFontShapeChangeRule {it}{ssc} {sscit} {scit}

2965 % \DeclareFontShapeChangeRule {sl}{n} {n} {}
2966 \DeclareFontShapeChangeRule {sl}{it} {it} {sl}
2967 % \DeclareFontShapeChangeRule {sl}{sl} {sl} {}
2968 \DeclareFontShapeChangeRule {sl}{sw} {sw} {it}
2969 \DeclareFontShapeChangeRule {sl}{sc} {scsl} {scit}
2970 \DeclareFontShapeChangeRule {sl}{ulc} {sl} {}
2971 \DeclareFontShapeChangeRule {sl}{up} {n} {}
2972 \DeclareFontShapeChangeRule {sl}{ssc} {sscsl} {scsl}

2973 % \DeclareFontShapeChangeRule {sc}{n} {n} {}
2974 \DeclareFontShapeChangeRule {sc}{it} {scit} {scsl}
2975 \DeclareFontShapeChangeRule {sc}{sl} {scsl} {scit}
2976 \DeclareFontShapeChangeRule {sc}{sw} {scsw} {scit}
2977 % \DeclareFontShapeChangeRule {sc}{sc} {sc} {}
2978 \DeclareFontShapeChangeRule {sc}{ulc} {n} {}

```

The next rule might be a bit surprising, and rightly so. It would be more correct if `sc` were not affected by `up`, so that it remains `sc` as shown in the commented out rule. However, for nearly three decades commands such as `\upshape` or `\textup` changed small caps back to the “normal” shape. So for backward compatibility we keep that behavior.

As a result you are currently typesetting in `scit` or `scsl` using `\upshape` twice will return you to the normal shape too, the first will change to `sc` and the second (because of the rule below) change that to `n`. This is the way `fontspec` implemented its version on this interface, so this rule means we are also compatible with the way `fontspec` behaved. Still it remains an oddity which I would rather liked to have avoided.

```

2979 % \DeclareFontShapeChangeRule {sc}{up} {sc} {}
2980 \DeclareFontShapeChangeRule {sc}{up} {n} {}
2981 % \DeclareFontShapeChangeRule {sc}{ssc} {ssc} {}

2982 % \DeclareFontShapeChangeRule {scit}{n} {n} {}
2983 \DeclareFontShapeChangeRule {scit}{it} {scit} {}
2984 \DeclareFontShapeChangeRule {scit}{sl} {scsl} {scit}
2985 \DeclareFontShapeChangeRule {scit}{sw} {scsw} {scit}
2986 \DeclareFontShapeChangeRule {scit}{sc} {scit} {}
2987 \DeclareFontShapeChangeRule {scit}{ulc} {it} {}

```

The next rule assumes that if `scit` exists then `it` exists as well. If not, the mechanism will save `ulc` in `\f@series`, which most certainly doesn’t exist. So when a font is later selected that would result in a substitution (so no harm done really). Alternatively, we could in this case use `n` as alternative, which may be a bit faster, but such a setup would be so weird in the first place that this isn’t worth the effort.

```

2988 \DeclareFontShapeChangeRule {scit}{up} {sc} {}
2989 \DeclareFontShapeChangeRule {scit}{ssc} {sscit} {scit}

```

```

2990 % \DeclareFontShapeChangeRule {scsl}{n} {n} {}
2991 \DeclareFontShapeChangeRule {scsl}{it} {scit} {scsl}
2992 \DeclareFontShapeChangeRule {scsl}{sl} {scsl} {}
2993 \DeclareFontShapeChangeRule {scsl}{sw} {scsw} {scsl}
2994 \DeclareFontShapeChangeRule {scsl}{sc} {scsl} {}
2995 \DeclareFontShapeChangeRule {scsl}{ulc} {sl} {}
2996 \DeclareFontShapeChangeRule {scsl}{up} {sc} {}
2997 \DeclareFontShapeChangeRule {scsl}{ssc} {sscsl} {scsl}

2998 % \DeclareFontShapeChangeRule {scsw}{n} {n} {}
2999 \DeclareFontShapeChangeRule {scsw}{it} {scit} {scsw}
3000 \DeclareFontShapeChangeRule {scsw}{sl} {scsl} {}
3001 \DeclareFontShapeChangeRule {scsw}{sw} {scsw} {}
3002 \DeclareFontShapeChangeRule {scsw}{sc} {scsw} {}
3003 \DeclareFontShapeChangeRule {scsw}{ulc} {sw} {}
3004 \DeclareFontShapeChangeRule {scsw}{up} {sc} {}
3005 \DeclareFontShapeChangeRule {scsw}{ssc} {sscsw} {scsw}

3006 % \DeclareFontShapeChangeRule {sw}{n} {n} {}
3007 % \DeclareFontShapeChangeRule {sw}{it} {it} {}
3008 % \DeclareFontShapeChangeRule {sw}{sl} {sl} {}
3009 % \DeclareFontShapeChangeRule {sw}{sw} {sw} {}
3010 \DeclareFontShapeChangeRule {sw}{sc} {scsw} {scit}
3011 \DeclareFontShapeChangeRule {sw}{ulc} {sw} {}
3012 \DeclareFontShapeChangeRule {sw}{up} {n} {}
3013 \DeclareFontShapeChangeRule {sw}{ssc} {sscsw} {scsw}

3014 % \DeclareFontShapeChangeRule {ssc}{n} {n} {}
3015 \DeclareFontShapeChangeRule {ssc}{it} {sscit} {sscsl}
3016 \DeclareFontShapeChangeRule {ssc}{sl} {sscsl} {sscit}
3017 \DeclareFontShapeChangeRule {ssc}{sw} {sscsw} {sscit}
3018 \DeclareFontShapeChangeRule {ssc}{sc} {sc} {}
3019 % \DeclareFontShapeChangeRule {ssc}{ssc} {ssc} {}
3020 \DeclareFontShapeChangeRule {ssc}{ulc} {n} {}

```

We implement the same logic as for `sc`, see above. The `ssc` shape doesn't have to care about 30 years of history, but it would be surprising if `\sscshape\upshape` did not work like `\scshape\upshape`.

```

3021 % \DeclareFontShapeChangeRule {ssc}{up} {ssc} {}
3022 \DeclareFontShapeChangeRule {ssc}{up} {n} {}

3023 % \DeclareFontShapeChangeRule {sscit}{n} {n} {}
3024 \DeclareFontShapeChangeRule {sscit}{it} {sscit} {}
3025 \DeclareFontShapeChangeRule {sscit}{sl} {sscsl} {sscit}
3026 \DeclareFontShapeChangeRule {sscit}{sw} {sscsw} {sscit}
3027 \DeclareFontShapeChangeRule {sscit}{ssc} {sscit} {}
3028 \DeclareFontShapeChangeRule {sscit}{sc} {scit} {}
3029 \DeclareFontShapeChangeRule {sscit}{ulc} {it} {}
3030 \DeclareFontShapeChangeRule {sscit}{up} {ssc} {}

3031 % \DeclareFontShapeChangeRule {sscsl}{n} {n} {}
3032 \DeclareFontShapeChangeRule {sscsl}{it} {sscit} {sscsl}
3033 \DeclareFontShapeChangeRule {sscsl}{sl} {sscsl} {}
3034 \DeclareFontShapeChangeRule {sscsl}{sw} {sscsw} {sscit}
3035 \DeclareFontShapeChangeRule {sscsl}{sc} {scsl} {}
3036 \DeclareFontShapeChangeRule {sscsl}{ulc} {sl} {}
3037 \DeclareFontShapeChangeRule {sscsl}{up} {ssc} {}

```


just switching to that shape or switching away from it; so we mimic that by providing the corresponding trivial rules to overrule those from above when we roll back.

```

3090 <latexrelease>\DeclareFontShapeChangeRule {n}{ssc} {ssc} {}
3091 <latexrelease>\DeclareFontShapeChangeRule {it}{ssc} {ssc} {}
3092 <latexrelease>\DeclareFontShapeChangeRule {sl}{ssc} {ssc} {}
3093 <latexrelease>\DeclareFontShapeChangeRule {sw}{ssc} {ssc} {}
3094 <latexrelease>
3095 <latexrelease>\DeclareFontShapeChangeRule {scit}{ssc}{ssc} {}
3096 <latexrelease>\DeclareFontShapeChangeRule {scsl}{ssc}{ssc} {}
3097 <latexrelease>\DeclareFontShapeChangeRule {scsw}{ssc}{ssc} {}
3098 <latexrelease>
3099 <latexrelease>\DeclareFontShapeChangeRule {ssc}{it} {it} {}
3100 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sl} {sl} {}
3101 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sw} {sw} {}
3102 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sc} {sc} {}
3103 <latexrelease>\DeclareFontShapeChangeRule {ssc}{ulc} {n} {}
3104 <latexrelease>\DeclareFontShapeChangeRule {ssc}{up} {n} {}
3105 <latexrelease>
3106 <latexrelease>\DeclareFontShapeChangeRule {sscit}{it} {it} {}
3107 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sl} {sl} {}
3108 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sw} {sw} {}
3109 <latexrelease>\DeclareFontShapeChangeRule {sscit}{ssc} {ssc} {}
3110 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sc} {sc} {}
3111 <latexrelease>\DeclareFontShapeChangeRule {sscit}{ulc} {n} {}
3112 <latexrelease>\DeclareFontShapeChangeRule {sscit}{up} {ssc} {}
3113 <latexrelease>
3114 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{it} {it} {}
3115 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sl} {sl} {}
3116 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sw} {sw} {}
3117 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sc} {sc} {}
3118 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{ulc} {n} {}
3119 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{up} {ssc} {}
3120 <latexrelease>
3121 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{it} {it} {}
3122 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sl} {sl} {}
3123 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sw} {sw} {}
3124 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{ssc} {ssc} {}
3125 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sc} {sc} {}
3126 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{ulc} {n} {}
3127 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{up} {ssc} {}
3128 <latexrelease>\EndIncludeInRelease

```

Before 2020-02-02 we don't really care about any existing shape change rules since the mechanism isn't used.

```

3129 <latexrelease>\IncludeInRelease{0000/00/00}%
3130 <latexrelease>      {\DeclareFontShapeChangeRule}{Rules for ssc and sw}%
3131 <latexrelease>\EndIncludeInRelease

```

2.2 Changing to a new shape

```

3132 <*2ekernel | latexrelease>
3133 <latexrelease>\IncludeInRelease{2021/06/01}%
3134 <latexrelease>      {\fontshape}{Font shape change}%

```

`\fontshape` Again the `\fontshape` now has to do a lookup to get to its new value in `\f@shape`. The method is exactly the same as in `\fontseries`.

```
3135 \DeclareRobustCommand\fontshape[1]
3136     {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
3137      {\delayed@f@adjustment\delayed@merge@font@shape{#1}}}
```

(End of definition for \fontshape.)

`\fontshapeforce` The unconditional version:

```
3138 \DeclareRobustCommand\fontshapeforce[1]
3139     {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
3140      {\delayed@f@adjustment\edef\f@shape{#1}}}
```

(End of definition for \fontshapeforce.)

Supporting rollback ...

```
3141 </2ekernel | latexrelease>
3142 <latexrelease>\EndIncludeInRelease
3143 <latexrelease>\IncludeInRelease{2020/02/02}%
3144 <latexrelease>    {\fontshape}{Font shape change}%
3145 <latexrelease>
3146 <latexrelease>\DeclareRobustCommand\fontshape[1]{\merge@font@shape{#1}}
3147 <latexrelease>\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{#1}}
3148 <latexrelease>
3149 <latexrelease>\EndIncludeInRelease
3150 <latexrelease>\IncludeInRelease{0000/00/00}%
3151 <latexrelease>    {\fontshape}{Font shape change}%
3152 <latexrelease>
3153 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
3154 <latexrelease>\let\fontshapeforce\@undefined
3155 <latexrelease>
3156 <latexrelease>\EndIncludeInRelease
3157 <*2ekernel | latexrelease>
3158 <latexrelease>\IncludeInRelease{2020/02/02}%
3159 <latexrelease>    {\merge@font@shape}{Font shape change rules}%
```

`\merge@font@shape` Look up the database entry (if existing) and act accordingly.

```
3160 \def\merge@font@shape#1{%
3161     \expandafter\expandafter\expandafter
3162     \merge@font@shape@
3163     \csname shape@\f@shape @#1\endcsname
3164     {#1}%
3165     \@nil
3166 }
```

(End of definition for \merge@font@shape.)

`\merge@font@shape@` Same game now, except that we look at shapes not series values and we can set the shape without the complication of dropping “m”s from the name as we had to for the series.

```
3167 \def\merge@font@shape@#1#2#3\@nil{%
3168     \def\reserved@a{#3}%
3169     \ifx\reserved@a\@empty
3170         \edef\f@shape{#2}%
3171     \else
```

`\reserved@a` is used in `\@font@shape@subst@warning` so we have to define it in addition to do the `\ifcstype` test

```

3172 \edef\reserved@a{\f@encoding /\f@family /\f@series/#1}%
3173 \ifcstype \reserved@a\endcstype
3174 \edef\f@shape{#1}%
3175 \else
3176 \ifcstype \f@encoding /\f@family /\f@series/#2\endcstype
3177 \edef\f@shape{#2}%
3178 \@font@shape@subst@warning
3179 \else
3180 \edef\f@shape{#3}%
3181 \@font@shape@subst@warning
3182 \fi
3183 \fi
3184 \fi
3185 }

```

(End of definition for \merge@font@shape@.)

`\merge@font@shape@without@substitution`
`\merge@font@shape@without@substitution@`
`\delayed@merge@font@shape`

See definition of `\selectfont` for how these macros are used.

```

3186 \def\merge@font@shape@without@substitution#1{%
3187 \expandafter\expandafter\expandafter
3188 \merge@font@shape@without@substitution@
3189 \cstype shape@\f@shape @#1\endcstype
3190 {#1}%
3191 \@nil
3192 }
3193 \def\merge@font@shape@without@substitution@#1#2#3\@nil{%
3194 \def\reserved@a{#3}%
3195 \ifx\reserved@a\@empty
3196 \edef\f@shape{#2}%
3197 \else
3198 \edef\f@shape{#1}%
3199 \fi
3200 }
3201 \let\delayed@merge@font@shape\merge@font@shape@without@substitution

```

*(End of definition for \merge@font@shape@without@substitution,
\merge@font@shape@without@substitution@, and \delayed@merge@font@shape.)*

`\normalshape` `\normalshape` resets both sub-axes if the default rules are used.

```

3202 \protected\def\normalshape
3203 {\not@math@alphabet\normalshape\relax
3204 \fontshape\shapedefault\selectfont}%

```

(End of definition for \normalshape.)

3 Make sure we win ...

This code implements one aspect of what the package `fontaxes` provide. So its redefinitions for the various shape commands, such as `\itshape` should no longer happen. We therefore force the standard definitions at `\AtBeginDocument` (later when this is defined). Once `fontaxes` is no longer doing such redefinitions that could be taken out again.

We use a separate macro so that we can easily disable this (in case of rollback).

`\reinstall@nfss@defs` I use `\protected` here not `\DeclareRobustCommand` to avoid extra status lines.

```

3205 \def\reinstall@nfss@defs{%
3206   \protected\def\upshape
3207     {\not@math@alphabet\upshape\relax
3208      \fontshape\updefault\selectfont}%
3209   \protected\def\slshape
3210     {\not@math@alphabet\slshape\relax
3211      \fontshape\sldefault\selectfont}%
3212   \protected\def\scshape
3213     {\not@math@alphabet\scshape\relax
3214      \fontshape\scdefault\selectfont}%
3215   \protected\def\itshape
3216     {\not@math@alphabet\itshape\mathit
3217      \fontshape\itdefault\selectfont}%
3218   \protected\def\ulcshape
3219     {\not@math@alphabet\ulcshape\relax
3220      \fontshape\ulc\selectfont}%
3221   \protected\def\swshape
3222     {\not@math@alphabet\swshape\relax
3223      \fontshape\swdefault\selectfont}%
3224   \protected\def\sscshape
3225     {\not@math@alphabet\sscshape\relax
3226      \fontshape\sscdefault\selectfont}%
3227 }

```

(End of definition for \reinstall@nfss@defs.)

Supporting rollback ...

```

3228 </2ekernel | latexrelease>
3229 <latexrelease>\EndIncludeInRelease
3230 <latexrelease>\IncludeInRelease{0000/00/00}%
3231 <latexrelease>  {\merge@font@shape}{Font shape change rules}%
3232 <latexrelease>
3233 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
3234 <latexrelease>\let\fontshapeforce\@undefined
3235 <latexrelease>
3236 <latexrelease>\let\merge@font@shape\@undefined
3237 <latexrelease>\let\merge@font@shape@\@undefined
3238 <latexrelease>
3239 <latexrelease>\let\merge@font@shape@without@substitution\@undefined
3240 <latexrelease>\let\merge@font@shape@without@substitution@\@undefined
3241 <latexrelease>\let\delayed@merge@font@shape\@undefined
3242 <latexrelease>
3243 <latexrelease>\let\normalshape\@undefined
3244 <latexrelease>

```

This is always called in `\document` so don't make it undefined.

```

3245 <latexrelease>
3246 <latexrelease>\let\reinstall@nfss@defs\relax
3247 <latexrelease>\EndIncludeInRelease

```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```

3248 <*2ekernel | latexrelease>
3249 <latexrelease>\IncludeInRelease{2020/10/01}%

```



```

3250 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3251 \g@addto@macro\@kernel@after@begindocument@before
3252                {\reinstall@nfss@defs\init@series@setup}
3253 </2ekernel | latexrelease>
3254 <latexrelease>\EndIncludeInRelease

    The initialization was introduced in 2020/02/02 but
3255 <latexrelease>\IncludeInRelease{2020/02/02}%
3256 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3257 <latexrelease>\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
3258 <latexrelease>\EndIncludeInRelease

3259 <latexrelease>\IncludeInRelease{0000/00/00}%
3260 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3261 <latexrelease>\EndIncludeInRelease
3262 <*2ekernel>
3263 </2ekernel>

```

File 26

ltfsstrc.dtx

1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the `tracefnt` package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the `tracefnt` package is loaded.

debugshow In addition to **infoshow** show also changing of math fonts as far as possible (this option can produce a large amount of output).

loading Show the name of external fonts when they are loaded. This option shows only “newly” loaded fonts not those already preloaded in the format or the class file before the `tracefnt` package became active.

pausing Turn all font warnings into errors so that L^AT_EX will stop.

2 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

When this file is processed directly by L^AT_EX this will produce the documentation as well.

```
1 <*driver>
2 \documentclass{ltxdoc}
3
4
5 %\OnlyDescription % comment out for implementation details
6
7 \begin{document}
8   \DocInput{ltfsstrc.dtx}
9 \end{document}
10 </driver>
```

3 The Implementation

Warning: Read the macro documentation with a grain of salt. It is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

If we are making a package file it is a good idea to test whether we are running under 2e. This code is actually placed at the very beginning of this file for easier maintenance, thus commented out here.

```
11 <*package>
12 %\NeedsTeXFormat{LaTeX2e}
13 %\ProvidesPackage{tracefnt}[??/??/?? v?.??]
14 %
15 %                               Standard LaTeX package (font tracing)
16 </package>
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.³⁴

```
16 <+debug> \input trace.sty
```

4 Handling Options

`\tracingfonts` Here is the definition of the integer register for the font trace. As a default in a package file we use 1 to give error messages if fonts are substituted. If this code is used for debugging or tracing reasons in the format file (i.e. in `fam.dtx`) we use 0 as the default. But if no font trace is used we build a definition that will produce a warning message.

```
17 <*2ekernel>
18 \message{NFSS tracing,}
19 \def\tracingfonts{%
20   \@font@warning{Command \noexpand\tracingfonts
21     not provided.\MessageBreak
22     Use the ‘tracefnt’ package.\MessageBreak Command found:}%
23   \count@}
24 </2ekernel>
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten by the next line if one of these modules are included.

```
25 <*package, trace, debug>
26 \newcount\tracingfonts
27 \tracingfonts=0
28 </package, trace, debug>
```

(End of definition for \tracingfonts.)

The option `errorshow` turns off all warnings so that only real errors are shown. `warningshow` corresponds to the NFSS default (when `tracefnt` is not loaded). `infoshow` is the default for this package here; and `debugshow`, `loading`, and `pausing` extend the amount of information even further.

```
29 <*package>
30 \DeclareOption{errorshow}{%
31   \def\@font@info#1{%
32     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
33   }
34 </package>
```

³⁴This package is not in distribution at the moment (and probably doesn't any longer work). Think of this part of the code as being historical artifacts.

```

33         {LaTeX Font Info: \space\space\space#1}}%
34     \def\@font@warning#1{%
35         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
36         {LaTeX Font Warning: #1}}%
37     }
38 \DeclareOption{warningshow}{%
39     \def\@font@info#1{%
40         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
41         {LaTeX Font Info: \space\space\space#1}}%
42     \def\@font@warning#1{%
43         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
44         {LaTeX Font Warning: #1}}%
45     }
46 \DeclareOption{infoshow}{%
47     \def\@font@info#1{%
48         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
49         {LaTeX Font Info: \space\space\space#1}}%
50     \def\@font@warning#1{%
51         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
52         {LaTeX Font Warning: #1}}%
53     }
54 \DeclareOption{loading}{%
55     \tracingfonts\tw@
56 }
57 \DeclareOption{debugshow}{%
58     \ExecuteOptions{infoshow}%
59     \tracingfonts\thr@@
60 }
61 \DeclareOption{pausing}{%
62     \def\@font@warning#1{%
63         \GenericError
64         {(Font)\@spaces\@spaces\@spaces\space\space}%
65         {LaTeX Font Warning: #1}%
66         {See the LaTeX Companion for details.}%
67         {I'll stop for every LaTeX Font Warning because
68         you requested\MessageBreak the 'pausing' option
69         to the tracefnt package.}}%
70     }

```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```

71 \ExecuteOptions{infoshow}
72 \ProcessOptions
73 \endpackage

```

We also need a default definition inside the kernel:

```

74 \ifx\kernel
75 \def\@font@info#1{%
76     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
77     {LaTeX Font Info: \space\space\space#1}}%
78 \def\@font@warning#1{%
79     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
80     {LaTeX Font Warning: #1}}%
81 \fi

```

5 Macros common to fam.tex and tracefmt.sty

In the first versions of `tracefmt.dtx` some macros of `fam.dtx`³⁵ were redefined to include the extra tracing information. Now these macros are all defined in this file (i.e. removed from `fam.dtx`) and different production versions can be obtained simply by specifying a different set of modules to include when generating `lftss.dtx`.

5.1 General font loading

`\extract@font` This macro organizes the font loading. It first calls `\get@external@font` which will return in `\external@font` the name of the external font file (the `.tfm`) as it was determined by the NFSS tables.

```
82 <*2ekernel|package>
83 \def\extract@font{%
84     \get@external@font
```

Then the external font is loaded and assigned to the font identifier stored inside `\font@name` (for this reason we need `\expandafter`).

```
85     \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
86 <*trace>
87     \ifnum \tracingfonts >\@ne
88     \@font@info{External font '\external@font'
89                 loaded as\MessageBreak \font@name}\fi
90 </trace>
```

Finally we call the corresponding “loading action” macros to finish things. First the font is locally selected to allow the use of `\font` inside the loading action macros.

```
91     \font@name \relax
```

The next two lines execute the “loading actions” for the family and then for the individual font shape.

```
92     \csname \f@encoding+\f@family\endcsname
93     \csname\curr@fontshape\endcsname
94     \relax
95     }
96 </2ekernel|package>
```

The `\relax` at the end needs to be explained. This is inserted to prevent `TEX` from scanning too far when it is executing the replacement text of the loading code macros.

(End of definition for `\extract@font`.)

`\get@external@font` This function tries to find an external font name. It will place the name into the macro `\external@font`. If no font is found it will return the one that was defined via `\DeclareErrorFont`.

```
97 <*2ekernel>
98 \def\get@external@font{%
```

We don’t know the external font name at the beginning.

```
99     \let\external@font\@empty
100     \edef\font@info{\expandafter\expandafter\expandafter\string
101                     \csname \curr@fontshape \endcsname}%
102     \try@size@range
```

³⁵This file is currently not distributed in documented form. Its code is part of `lftss.dtx`.

If this failed, we'll try to substitute another size of the same font. This is done by the `\try@size@substitution` macro. It “knows about” `\do@extract@font`, `\font@name`, `\f@size`, and so on.

```

103 \ifx\external@font\@empty
104 \try@size@substitution
105 \ifx\external@font\@empty
106 \latex@error{Font \expandafter \string\font@name\space
107 not found}\@eha
108 \error@fontshape
109 \get@external@font
110 \fi\fi
111 }
112 \endkernel

```

(End of definition for `\get@external@font`.)

```

113 <*2ekernel | latexrelease | package>
114 <latexrelease> \IncludeInRelease{2021/06/01}%
115 <latexrelease> {\selectfont}{Add hook to \selectfont}%

```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```

116 \DeclareRobustCommand\selectfont
117 {%

```

When `debug` is specified we actually want something like ‘undebug’. The font selection is now stable so that using `\tracingall` on some other macros will show us a lot of unwanted information about font loading. Therefore we disable tracing during font loading as long as `\tracingfonts` is less than 4.

```

118 <+debug> \pushtracing
119 <+debug> \ifnum\tracingfonts<4 \tracingoff
120 <+debug> \else \tracingon\p@selectfont \fi

```

If `\baselinestretch` was redefined by the user it will not longer match its internal counterpart `\f@linespread`. If so we call `\set@fontsize` to prepare `\size@update`.

```

121 \ifx\f@linespread\baselinestretch \else
122 \set@fontsize\baselinestretch\f@size\f@baselineskip \fi

```

The series and shape updates are only prepared by `\fontseries` and `\fontshape` but not executed until after we are ready to change the font face. This way they happen after a possibly new family is set which is important because they look at the available font faces in that family and alter the selection based on availability. Several calls to `\fontseries` or `\fontshape` are delayed in the order in which they appear, so that by switching them one can work around missing intermediate font faces and avoid substitutions.

We first attempt to do the merge without any substitution. As we might end up with a non-existing font face we may have to restart and therefore save the current values of `\f@series` and `\f@shape` before the merge.

But first we make a quick test to see if there are any delayed actions, because if not it is pointless to make all the assignments and try loading a missing fontshape.

```

123 \ifx\delayed@f@adjustment\@empty
124 \else
125 \let\f@shape@savd\f@shape
126 \let\f@series@savd\f@series

```

Then we run the delayed adjustments (which use the `\..@without@substitution` commands):

```
127      \delayed@f@adjustment
```

We then check if the resulting combination is valid but for this we have to make sure that the appropriate `.fd` is loaded if that hasn't happened so far.

```
128      \maybe@load@fontshape
```

```
129      \ifcsname \f@encoding/\f@family/\f@series/\f@shape \endcsname
```

If this macro is defined then we are good and no further action is necessary.

Otherwise the combination is not valid, so we redo the merge but this time with substitutions.

```
130      \else
```

```
131          \let\f@shape\f@shape@saved
```

```
132          \let\f@series\f@series@saved
```

```
133          \let\delayed@merge@font@shape\merge@font@shape
```

```
134          \let\delayed@merge@font@series\merge@font@series
```

```
135          \delayed@f@adjustment
```

```
136          \let\delayed@merge@font@shape\merge@font@shape@without@substitution
```

```
137          \let\delayed@merge@font@series\merge@font@series@without@substitution
```

```
138      \fi
```

Now the series and shape values are updated and we clear `\delayed@f@adjustment`. This is important because on the next execution of `\selectfont` we should not mistakenly redo the delayed actions if there wasn't any series or shape change.

```
139      \let\delayed@f@adjustment\@empty
```

```
140      \fi
```

If the series was forced we should now cancel that in case the next series change is done with some low-level setting to `\f@series`.

```
141      \@forced@seriesfalse
```

Then we generate the internal name of the font by concatenating *family*, *series*, *shape*, and current *size*, with slashes as delimiters between them. This is much more readable than standard L^AT_EX's `\twbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
142      \xdef\font@name{%
```

```
143          \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
144      \pickup@font
```

Then we select the font.

```
145      \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in `every` but using a different interface).

```
146      \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
147      \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
148 \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
149 <+debug> \poptracing
150 }
```

(End of definition for `\selectfont`.)

selectfont Declare the hook used in `selectfont` in the kernel, but not inside the `tracefnt` package.

```
151 <-trace> \NewHook{selectfont}
```

(End of definition for `selectfont`.)

If `\tracingfonts` is greater than 2 we also show the font switch inside `\selectfont`. We do this by adding this code to the hook in the `tracefnt` package: macro might redefine `\font@name`.

```
152 <*trace>
153 \AddToHook{selectfont}
154   {\ifnum \tracingfonts>\tw@
155     \font@info{Switching to \font@name}\fi}
156 </trace>
157 </2ekernel | latexrelease | package>
158 <latexrelease> \EndIncludeInRelease
```

With `\selectfont` having different definitions in different kernels we also have to provide them in the `tracefnt` package to support rollback. In packages that works a bit differently and therefore we have to provide an empty block there.

```
159 <package> \IncludeInRelease{2021/06/01}%
160 <package> \selectfont{\Add hook to \selectfont}%
161 <package> \EndIncludeInRelease

162 <latexrelease | package> \IncludeInRelease{0000/00/00}%
163 <latexrelease | package> \selectfont{\Add hook to \selectfont}%
164 <latexrelease | package>
165 <latexrelease | package> \DeclareRobustCommand\selectfont
166 <latexrelease | package>   {%
167 <latexrelease | package>   \ifx\f@linespread\baselinestretch \else
168 <latexrelease | package>   \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
169 <latexrelease | package>   \xdef\font@name{%
170 <latexrelease | package>   \curname\curr@fontshape/\f@size\endcurname}%
171 <latexrelease | package>   \pickup@font
172 <latexrelease | package>   \font@name
173 <latexrelease | package>   \size@update
174 <latexrelease | package>   \enc@update
175 <latexrelease | package>   }
176 <latexrelease | package>
177 <latexrelease | package> \EndIncludeInRelease
```

\set@fontsize The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
178 <*2ekernel | package>
179 \def\set@fontsize#1#2#3{%
180   \@defaultunits\@tempdimb#2pt\relax\@nnil
```



```

181 \edef\f@size{\strip@pt\@tempdimb}%
182 \@defaultunits\@tempskipa#3pt\relax\@nnil
183 \edef\f@baselineskip{\the\@tempskipa}%
184 \edef\f@linespread{#1}%

```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```

185 \let\baselinestretch\f@linespread

```

Additional processing will happen within `\selectfont`. For this reason the macro `\size@update` (which will be called in `\selectfont`) will be defined to be:

```

186 \def\size@update{%

```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```

187 \baselineskip\f@baselineskip\relax
188 \baselineskip\f@linespread\baselineskip
189 \normalbaselineskip\baselineskip

```

then to set up a new `\strutbox`

```

190 \setbox\strutbox\hbox{%
191 \vrule\@height.7\baselineskip
192 \@depth.3\baselineskip
193 \@width\z@}%

```

We end with a bit of tracing information.

```

194 \ifnum \tracingfonts>\tw@
195 \ifnum \tracingfonts>\tw@
196 \ifx\f@linespread\@empty
197 \let\reserved@a\@empty
198 \else
199 \def\reserved@a{\f@linespread x}%
200 \fi
201 \@font@info{Changing size to \f@size/\reserved@a
202 \f@baselineskip}%
203 \aftergroup\type@restoreinfo \fi
204 \iftracingfonts\tracingfontsize\fi

```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```

205 \let\size@update\relax}%
206 }

```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let`!

(End of definition for \set@fontsize.)

\size@update Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```

207 \let\size@update\relax

```

(End of definition for \size@update.)

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```

208 <*trace>
209   \def\type@restoreinfo{%
210     \ifx\f@linespread\@empty
211       \let\reserved@a\@empty
212     \else
213       \def\reserved@a{\f@linespread x}%
214     \fi
215     \@font@info{Restoring size to
216               \f@size/\reserved@a\f@baselineskip}}
217 </trace>

```

(End of definition for `\type@restoreinfo`.)

`\glb@settings` The macro `\glb@settings` globally selects all math fonts for the current size if necessary.

`\glb@currsiz` `\glb@currsiz` `\def\glb@settings{%`

When `\glb@settings` gains control a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every *math group* the new assignments are appended to `\math@fonts`. But this happens only if the `\math@fonts` switch is set to true. However, we always set up the correct math sizes for script and scriptscript fonts since they may be needed even if we don't set up the whole math machinery.

Here we set the math size, script size and scriptscript size. If the `S@...` macro is not defined we have to first calculate the three sizes.

```

219   \expandafter\ifx\csname S@\f@size\endcsname\relax
220     \calculate@math@sizes
221   \fi

```

The effect of this is that `\calculate@math@sizes` may or may not define the `S@...` macro. In the first case the next time the same size is requested this macro is used, otherwise `\calculate@math@sizes` is called again. This also sets the `\math@fonts` switch. If it is true we must switch the math fonts.

```

222   \csname S@\f@size\endcsname
223   \ifmath@fonts
224 <*trace>
225     \ifnum \tracingfonts>\tw@
226       \@font@info{Setting up math fonts for
227                 \f@size/\f@baselineskip}\fi
228 </trace>

```

Inside a group we execute the macro for the current math *version*. This sets `\math@fonts` to a list of `\textfont...` assignments. `\getanddefine@fonts` (which may be called at this point) needs the `\escapechar` parameter to be set to `-1`.

```

229   \begingroup
230     \escapechar\m@ne
231     \csname mv@\math@version \endcsname

```

Then we set `\globaldefs` to 1 so that all following changes are done globally. The math font assignments recorded in `\math@fonts` are executed and `\glb@currsiz` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```

232     \globaldefs\@ne
233     \math@fonts

```

Finally we execute any code that is supposed to happen whenever the math font setup changes. This register will be executed in local mode which means that everything that is supposed to have any effect should be done globally inside. We can't execute it within `\globaldefs\@one` as we don't know what ends up inside this register, e.g., it might contain calculations which use some local registers to calculate the final (global) value.

Otherwise we announce that the math fonts are not set up for this size.

(End of definition for \glb@settings and \glb@currsiz.)

(End of definition for \baselinestretch.)

(End of definition for \every@math@size.)

TeX uses the math fonts that are current when the end of a formula is reached. If we don't want to keep font setups local to every formula (which would result in an enormous overhead), we have to be careful not to end up with the wrong setup in case formulas are nested, e.g., we need to be able to handle

This is handled in the following way:

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.
3. If both are the same we assume that we can use the current math font setup without adjustment.
4. If they differ we call `\glb@settings` which changes the math font setup and updates `\glb@currsz`.
 - (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\glb@settings` is executed again in the outer formula, so that the old setup is automatically restored.
 - (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
 - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
 - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesettable. Compared to original implementation in NFSS2 the new algorithm has the overhead of one test per formula to see if the current math setup is valid (in the original algorithm the setup was always valid, thus no test was necessary).
- In nested formulae the math font setup is restored in the outer formula by a series of `\aftergroup` commands and checks. Compared to the original algorithm this involves additional checks ($2 \times$ (non-math levels) per inner formula).

5.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4 except that instead of a switch the macro `\init@restore@glb@settings` is used.

```

251 <*2ekernel | package>
252 \def\check@mathfonts{%
253   \ifx \glb@currsz \f@size
254   <*trace>
255     \ifnum \tracingfonts>\thr@@
256       \@font@info{*** MATH: no change \f@size\space
257         curr/global (\curr@math@size/\glb@currsz)}\fi
258   </trace>
259   \else
260   <*trace>
261     \ifnum \tracingfonts>\thr@@
262       \@font@info{*** MATH: setting up \f@size\space
263         curr/global (\curr@math@size/\glb@currsz)}\fi
264   </trace>

```

```

265     \glb@settings
266     \init@restore@glb@settings
267   \fi
268   \let\curr@math@size\font@size
269   \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
270 }

```

(End of definition for \check@mathfonts.)

\init@restore@glb@settings This macro does by default nothing but get redefined inside \check@mathfonts to initiate fontsize restoring in nested formulas.

```

271 <-trace>\let\init@restore@glb@settings\relax
272 <*trace>
273 \def\init@restore@glb@settings{%
274     \ifnum \tracingfonts>\thr@@
275     \font@info{*** MATH: no resetting (not in
276         nested math)}\fi
277 }
278 </trace>

```

(End of definition for \init@restore@glb@settings.)

\restglb@settings This macro will be executed the first time after the current formula.

```

279 \def\restglb@settings{%
280 <*trace>
281     \ifnum \tracingfonts>\thr@@
282     \font@info{*** MATH: restoring}\fi
283 </trace>
284     \begingroup
285     \let\font@size\curr@math@size
286     \ifx\glb@currsz \font@size
287 <*trace>
288     \ifnum \tracingfonts>\thr@@
289     \font@info{*** MATH: ... already okay (\font@size)}\fi
290 </trace>
291     \else
292 <*trace>
293     \ifnum \tracingfonts>\thr@@
294     \font@info{*** MATH: ... to \font@size}\fi
295 </trace>
296     \glb@settings
297   \fi
298 \endgroup
299 }

```

(End of definition for \restglb@settings.)

5.2.3 Other code for math

\use@mathgroup The \use@mathgroup macro should be used in user macros to select a math group. Depending on whether or not the margid option is in force it has two or three arguments. For this reason it should be called as the last macro.

First we test if we are inside math mode since we don't want to apply a useless definition.

```

300 \def\use@mathgroup#1#2{\relax\ifmmode

```

```

301 <*trace>
302   \ifnum \tracingfonts>\tw@
303     \count@#2\relax
304     \@font@info{Using \noexpand\mathgroup
305               (\the\count@) #2}\fi
306 </trace>

```

If so we first call the ‘=’ macro (i.e. argument three) to set up special things for the selected math group. Then we call `\mathgroup` to select the group given by argument two and finally we place `#1` (i.e., the argument of the `<math alphabet identifier>`) at the end. This part of the code is surrounded by two commands which behave like `\begingroup` and `\endgroup` if we want `<math alphabet identifier>`s but will expand into `\@empty` if we want simply switches to a new math group. Since argument number 2 may be a digit instead of a control sequence we add a `\relax`. Otherwise something like `\mit{1}` would switch to math group 11 (and back) instead of printing an oldstyle 1.

```

307   \math@bgroup
308     \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
309     #1\fi
310     \mathgroup#2\relax

```

Before we reinsert the swallowed token (arg. three) into the input stream, in the case that the `<math alphabet identifier>` isn’t called in math mode, we remove the `\fi` with the `\expandafter` trick. This is necessary if the token is actually an macro with arguments. In such a case the `\fi` will be misinterpreted as the first argument which would be disastrous.

```

311   \expandafter\math@egroup\fi}%

```

The surrounding macros equal `\begingroup` and `\endgroup`. But using internal names makes it possible to overwrite their meaning in certain cases. This is for example used in $\mathcal{MS}$ -TeX macros for placing accents.

(End of definition for `\use@mathgroup`.)

`\math@egroup` If the `margid` option is in force (which can be tested by looking at the definition of `\math@bgroup`) we change the `\math@egroup` command a bit to display the current `<math group number>` after it closes the scope of `<math alphabet>` with `\endgroup`.

```

312 <*trace>
313   \ifx\math@bgroup\bgroup
314     \def\math@egroup#1{#1\egroup
315       \ifnum \tracingfonts>\tw@
316         \@font@info{Restoring \noexpand\mathgroup
317               (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
318               }\fi}
319   \fi
320 </trace>

```

(End of definition for `\math@egroup`.)

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the `<math group number>` and the *family/series/shape* name as a control sequence.

```

321 \def\getanddefine@fonts#1#2{%

```

First we turn of tracing when `\tracingfonts` is less than 4.

```

322 <+debug> \pushtracing
323 <+debug> \ifnum\tracingfonts<4 \tracingoff
324 <+debug> \else \tracingon\getanddefine@fonts \fi

325 <*trace>
326 \ifnum \tracingfonts>\tw@
327 \count@#1\relax
328 \font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
329 \string#2 \tf@size/\sf@size/\ssf@size}\fi
330 /trace>

```

We append the current `\tf@size` to `#2` to obtain the font name.³⁶ Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```

331 \xdef\font@name{\csname \string#2/\tf@size\endcsname}%

```

Then we call `\pickup@font` to load it if necessary. We remember the internal name as `\textfont@name`.

```

332 \pickup@font \let\textfont@name\font@name

```

Same game for `\scriptfont` and `\scriptscriptfont`, but here we additionally transform the fontname through `\transform@scriptfont` to allow different family names in math scripts.

```

333 \xdef\font@name{\csname \transform@scriptfont {sf}{#2}/\sf@size\endcsname}%
334 \pickup@font \let\scriptfont@name\font@name
335 \xdef\font@name{\csname \transform@scriptfont {ssf}{#2}/\ssf@size\endcsname}%
336 \pickup@font

```

Then we append the new `\textfont...` assignments to the `\math@fonts`.

```

337 \edef\math@fonts{\math@fonts
338 \textfont#1\textfont@name
339 \scriptfont#1\scriptfont@name
340 \scriptscriptfont#1\font@name}%

```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```

341 <+debug> \poptracing
342 }

```

(End of definition for `\getanddefine@fonts`.)

`\transform@scriptfont` `\transform@scriptfont` has two arguments: the name of the script size we transform the font for (always `sf` or `ssf`) and the *family/series/shape* name as a `csname`. It expands to a new *family/series/shape* name which will be used for this script. For this `\transform@scriptfont` mostly delegates to `\__nfss_transform_scriptfont:nw`, which searches for a replacement marker `\__nfss_mapped_scriptfont_family_{scriptsize}_{encoding}` with a remapped encoding and family.

```

343 \ExplSyntaxOn
344 \cs_set:Npn \transform@scriptfont #1 #2 {
345 \exp_last_unbraced:Nno
346 \__nfss_transform_scriptfont:nw
347 {#1}
348 {\token_to_str:N #2}
349 }

```

³⁶One might ask why this expansion does not generate a macro name that starts with an additional `\` character. The solution is that `\escapechar` is set to `-1` before `\getanddefine@fonts` is called.

```

350 \cs_set:Npn \__nfss_transform_scriptfont:nw #1 #2 / #3 / {
351   \cs_if_exist_use:cF { __nfss_mapped_scriptfont_family_ #1 _ #2 / #3 } { #2 / #3 }
352   /
353 }

```

(End of definition for `\transform@scriptfont`.)

`\DeclareMathScriptfontMapping` `\DeclareMathScriptfontMapping` can be used to define replacement font families which will be used for math subscripts or subsubscripts when a specific font family is used as the main math font. For this `\DeclareMathScriptfontMapping` takes three pairs of encoding and family names as arguments: The first pair identifies the base font for which associated script and scriptscript variants should be declared. The second pair describes the family which should be used as scriptfonts (e.g. in subscripts). Finally the third pair describes the family which should be used as scriptscriptfonts (e.g. in nested subscripts).

```

354 \cs_set:Npn \DeclareMathScriptfontMapping #1 #2 #3 #4 #5 #6 {
355   \cs_gset:cpn { __nfss_mapped_scriptfont_family_sf_ #1 / #2 } { #3 / #4 }
356   \cs_gset:cpn { __nfss_mapped_scriptfont_family_ssf_ #1 / #2 } { #5 / #6 }
357 }
358 \ExplSyntaxOff
359 </2ekernel|package>

```

(End of definition for `\DeclareMathScriptfontMapping`.)

6 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\@gobble` which discards the following argument, otherwise it expands to `\@firstofone` which reproduces its argument.

```

360 <*2ekernel>
361 \def\ifnot@nil#1{\def\reserved@a{#1}%
362   \ifx\reserved@a\@nnil \expandafter\@gobble
363   \else \expandafter\@firstofone\fi}

```

(End of definition for `\ifnot@nil`.)

`\remove@to@nnil` Three other auxiliary macros will be needed in the following: `\remove@to@nnil` gobbles up everything up to, and including, the next `\@nnil` token, and `\remove@angles` and `\remove@star` do the same for the character `>` and `*`, respectively, instead of `\@nnil`.

```

364 \def\remove@to@nnil#1\@nnil{}
365 \def\remove@angles#1>{\set@simple@size@args}
366 \def\remove@star#1*{#1}

```

(End of definition for `\remove@to@nnil`, `\remove@angles`, and `\remove@star`.)

`\extract@sizefn` This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```

367 \def\extract@sizefn#1*#2\@nil{%
368   \if>#2>\set@size@funct@args#1\@nil
369     \let\sizefn@info\@empty
370   \else\expandafter\set@size@funct@args\remove@star#2\@nil
371     \def\sizefn@info{#1}\fi
372 }

```


(End of definition for `\extract@sizefn`.)

`\try@simple@size` This function tries to extract the given size (specified by `\f@size`) for the requested font shape. The font information must already be present in `\font@info`. The central macro that does the real work is `\extract@fontinfo`. We will first give a simple example how this macro works, and describe it in full generality later.

Assume that the requested parameters are: *encoding scheme* ‘OT1’, *family* ‘cm’, *series* ‘sansserif’, *shape* ‘normal’, and *size* ‘12’. The corresponding font definitions have already been extracted from the macro `\OT1/cm/sansserif/normal` and stored in `font@info`. (Otherwise `\extract@fontinfo` doesn’t get called.) This information consists of a token list made of characters of category code 12 of the form

```
<10*>cmss10<12*>cmss12<17*>cmss17
```

For reasonable packages one usually needs more sizes but this is sufficient to get the flavour. We will define a macro `\extract@fontinfo` to find the external font name (‘cmss12’) for us:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \set@simple@size@args#3<#4\@nnil
  \execute@size@function{#2}}}
```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

#1 will contain all characters before `<12*>`, #2 will be empty, #3 will be exactly `cmss12`, and #4 will be `17>cmss17`. The expansion is therefore

```
\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}
```

This means: the default (empty) size function will be executed, with its optional argument set to empty and its mandatory argument set to `cmss12` by `\set@simple@size@args`. As we discussed earlier, the effect of the default size function is to load the given external font (`cmss12`) at the specified size (12)—which is exactly what was intended.

But this is only part of the whole story. It may be that the size requested does not occur in the token list `\font@info`. And the simple definition of `\extract@fontinfo` we gave above does not allow to specify give more than one size specification in front of the external font name.

Let’s address these two problems separately. The first one is solved with the following trick: We define `\extract@fontinfo` as follows:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \ifnot@nil{#3}%
  {\set@simple@size@args#3<#4\@nnil
   \execute@size@function{#2}}%
  }%}
```

How does this work? We call `\extract@fontinfo` via

```
\expandafter\extract@fontinfo\font@info<12*>\@nil\@nnil
```

i.e. by appending `<12*>\@nil<\@nnil`. If the size ('12' in this case) appears in `\font@info` everything works as explained above, the only difference being that argument #4 of `\extract@fontinfo` additionally gets the tokens `<12*>\@nil<\@nnil`. However, if the size is not found everything up to the final `<12*>` is in argument #1, #3 gets `\@nil`, and #2 and #4 are empty. The macro `\ifnot@nil` will discard the calls to `\set@simple@size@args` and `execute@size@function`, and hence `\font@info` will continue to be equal to `\@empty`. This means that no simple size specification matching the requested size could be found.

The second problem (more than one simple size specification for one external font name) will be addressed in `\set@simple@size@args` below.

The macros are hidden inside other control sequences so that we have to build `\extract@fontinfo` in several steps.

So here's the actual definition of `\extract@font` in `\try@simple@size`.

373 % % this could be replaced by `\try@size@range` making the subst slower!

374 \def\try@simple@size{%

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

375 \def\reserved@a{\def\extract@fontinfo####1}%

Now we can define `\extract@fontinfo`. Here we handle a small but convenient variation: in case of the default (empty) size function it is allowed to omit the `*` character.

376 \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%

377 \ifnot@nil{##2}%

378 {\set@simple@size@args##2<##3\@nnil

379 \execute@size@function\sizefn@info

380 }%}

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

381 \expandafter\expandafter

382 \expandafter\extract@fontinfo\expandafter\font@info

383 \expandafter<\f@size>\@nil<\@nnil

384 }

(End of definition for `\try@simple@size`.)

`\set@simple@size@args` As promised above, the macro `\set@simple@size@args` will handle the case of several size specifications in a row. If another size specification follows, the very first token of its argument list is the character `<`. By starting the definition as follows,

385 \def\set@simple@size@args#1<{%

parameter #1 is empty in this case, and contains the size function's arguments otherwise. We distinguish these two cases (Note that the character `<` cannot appear in #1) by calling `\remove@angles` for empty #1 and `\extract@sizefn` otherwise. In the latter case we have to take care of the remaining character tokens and discard them. This is done by `\remove@to@nnil`. Note also the use of Kabelschacht's method.

386 \if<#1<%

387 \expandafter\remove@angles

388 \else

389 \extract@sizefn#1*\@nil

390 \expandafter\remove@to@nnil

391 \fi}

(End of definition for `\set@simple@size@args`.)

Now, we are through with the case of a simple size, except for calling the size function. This will be handled later, as it is the same mechanism for all types of size specification. We will now proceed to macros for extraction of size range specification.

`\extract@rangefontinfo` `\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens `<\@nil->`. It looks for font ranges with font size functions. It's operation is rather simple: it discards everything up to the next size specification and passes this on to `\is@range` for inspection. The specification (parameter #2) is inserted again, in case it is needed later.

```
392 \def\extract@rangefontinfo#1<#2>{%
393     \is@range#2->\@nil#2>}
```

(End of definition for `\extract@rangefontinfo`.)

`\is@range` `\is@range` is again a sort of dispatcher macro: if the size specification it is looking at is not a range specification it discards it and calls `\extract@rangefontinfo` to continue the search. Otherwise it calls `\check@range` to check the requested size against the specified range.

From the way `\is@range` is called inside `\extract@rangefontinfo` we see that #2 is the character `>` if the size specification found is a simple one (that does not contain a `-` character). This is checked easily enough and `\extract@rangefontinfo` called again. Note that the extra tokens inserted after the `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```
394 \def\is@range#1-#2\@nil{%
395     \if>#2\expandafter\check@single\else
396     \expandafter\check@range\fi}
```

(End of definition for `\is@range`.)

`\check@range` `\check@range` takes lower bound as parameter #1, upper bound as #2, size function as #3 and the size function's arguments as #4. If #3 is the special token `\@nil \font@info` is exhausted and we can stop searching.

```
397 \def\check@range#1-#2>#3<#4\@nnil{%
398     \ifnot@nil{#3}{%
```

If #3 wasn't `\@nil` we have a range. We start by assuming that we have to recurse. Note that we have to reinsert an `<` as it was already removed by scanning.

```
399     \def\reserved@f{\extract@rangefontinfo<#4\@nnil}%
```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If `\upper@bound` is zero after the assignment we set it to `\maxdimen` (upper open range). We need to use a `<dimen>` register for the scan since we may have a decimal number as the boundary.

```
400     \upper@bound0#2\p@
401     \ifdim\upper@bound=\z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against `\f@size`. If it is larger or equal than `\f@size` this range is no good and we have to recurse.

```
402     \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If \f@size is smaller than the boundary we have to recurse.

```

403      \lower@bound0#1\p@
404      \ifdim \f@size \p@<\lower@bound
405      \else

```

If both tests are passed we can try executing the size function.

```

406      \set@simple@size@args#3<#4\@nnil
407      \execute@size@function\sizefn@info

```

If the function was successful it should have left an external font name in \external@font. We use this to see if we can stop scanning. Otherwise we recurse.

```

408      \ifx\external@font\@empty
409      \else
410      \let\reserved@f\@empty
411      \fi
412      \fi
413      \fi
414      \reserved@f}}

```

(End of definition for \check@range.)

\lower@bound We use two dimen registers \lower@bound and \upper@bound to store the lower and upper endpoints of the range we found.

```

415 \newdimen\lower@bound
416 \newdimen\upper@bound

```

(End of definition for \lower@bound and \upper@bound.)

\check@single \check@single takes the size as parameter #1, size function as #2 and the size function's arguments as #3. We can assume that there is always something in the pipeline since the very last entry is a faked range (see above).

```

417 \def\check@single#1>#2<#3\@nnil{%

```

We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```

418      \def\reserved@f{\extract@rangefontinfo<#3\@nnil}%

```

Now we check the size against \f@size. If it is not equal \f@size it is no good and we have to recurse.

```

419      \ifdim \f@size \p@=#1\p@

```

Otherwise if this test is passed we can try executing the size function.

```

420      \set@simple@size@args#2<#3\@nnil
421      \execute@size@function\sizefn@info

```

If the function was successful it should have left an external font name in \external@font. We use this to see if we can stop scanning. Otherwise we recurse.

```

422      \ifx\external@font\@empty
423      \else
424      \let\reserved@f\@empty
425      \fi
426      \fi
427      \reserved@f}

```

(End of definition for \check@single.)

\set@size@funct@args This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token \@nil.

```

428 \def\set@size@funct@args{\@ifnextchar[%
429   \set@size@funct@args@\set@size@funct@args@[]}}
430 \def\set@size@funct@args@[#1]#2\@nil{%
431   \def\mandatory@arg{#2}%
432   \def\optional@arg{#1}}
433 \</2ekernel>

```

(End of definition for \set@size@funct@args and \set@size@funct@args@.)

\DeclareSizeFunction This function defines a new size function hiding the internal from the designer. The body of the size function may use \optional@arg and \mandatory@arg denoting the optional and mandatory argument that may follow the size specification <...>.

```

434 \<2ekernel>
435 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
436 \@onlypreamble\DeclareSizeFunction
437 \</2ekernel>

```

(End of definition for \DeclareSizeFunction.)

\execute@size@function This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a \relax).

```

438 \<2ekernel | package>
439 \def\execute@size@function#1{%
440 \<trace>
441   \ifundefined{s@fct@#1}%
442     {\errmessage{Undefined font size function #1}%
443      \s@fct@}%
444     {\csname s@fct@#1\endcsname}%
445 \</trace>
446 \<-trace>   \csname s@fct@#1\endcsname
447 }
448 \</2ekernel | package>

```

(End of definition for \execute@size@function.)

\try@size@range This macro tries to find a suitable range for requested size (specified by \f@size) in \font@info. All the relevant action is done in \extract@rangefontinfo. All that needs to be done is to stuff in the token list in \font@info so that \extract@rangefontinfo can inspect it. Note the <-*\@nil>< token at the end to stop scanning.

```

449 \<2ekernel>
450 \def\try@size@range{%
451   \expandafter\extract@rangefontinfo\font@info <-*\@nil>\@nnil
452 }

```

(End of definition for \try@size@range.)

\try@size@substitution This is the last thing that can be tried. If the desired \f@size is found neither among the simple size specifications nor in one of the ranges the whole list of size specifications is searched for a nearby simple size.

```

453 \gdef\try@size@substitution{%

```

First we do some initializations. `\@tempdimb` will hold the difference between the wanted size and the best solution found so far, so we initialize it with `\maxdimen`. The macro `\best@size` will hold the best size found, nothing found is indicated by the empty value.

```
454 \tempdimb \maxdimen
455 \let \best@size \@empty
```

Now we loop over the specification

```
456 \expandafter \try@simples \font@info <\number\@M>\@nil<\@nnil
457 }
```

(End of definition for \try@size@substitution.)

`\font@submax` The macro `\font@submax` records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at `\end{document}`. The macro `\fontsubfuzz` contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```
458 \def\font@submax{0pt}
459 \def\fontsubfuzz{.4pt}
460 </2ekernel>
461 <+package>\def\fontsubfuzz{0pt}
```

(End of definition for \font@submax and \fontsubfuzz.)

`\try@simples` `\try@simples` goes through a font shape definition in the input until it recognizes the tokens `<*\@nil>`. It looks for simple sizes to determine the two closest sizes. It is assumed that simple sizes are in increasing order.

```
462 <*2ekernel>
463 \gdef\try@simples#1<#2>{%
464 \tryif@simple#2->\tryif@simple}
```

(End of definition for \try@simples.)

`\tryis@simple` `\tryis@simple` is similar to `\is@range`. If it sees a simple size, it checks it against the value of `\f@size` and sets `\lower@font@size` or `\higher@font@size`. In the latter case, it stops the iteration. By adding `<\number\@M>` at the end of the line we always have an end point. This is a hack which probably should be corrected.

First it checks whether it is finished already, then whether the size specification in question is a simple one.

```
465 \gdef\tryif@simple#1-#2\tryif@simple{%
```

Most common case for `\reserved@f` first:

```
466 \let \reserved@f \try@simples
467 \if>#2%
```

If so, it compares it to the value of `\f@size`. This is done using a dimen register since there may be fractional numbers.

```
468 \dimen@ #1\p@
469 \ifdim \dimen@<\@M\p@
```

If `\dimen@` is `\M\p@` we have reached the end of the fontspec (hopefully) otherwise we compare the value with `\f@size` and compute in `\@tempdimc` the absolute value of the difference between the two values.

```

470     \ifdim \f@size\p@<\dimen@
471         \@tempdimc \dimen@
472         \advance\@tempdimc -\f@size\p@
473     \else
474         \@tempdimc \f@size\p@
475         \advance\@tempdimc -\dimen@
476     \fi

```

The result is then compared with the smallest difference we have encountered, if the new value (in `\@tempdimc` is smaller) we have found a size which is a better approximation so we make it the `\best@size` and adjust `\@tempdimb`.

```

477     \ifdim \@tempdimc<\@tempdimb
478         \@tempdimb \@tempdimc
479         \def \best@size{#1}%
480     \fi

```

When we have reached the end of the fontspec we substitute the best size found (if any). We code this inline to save macro space; in the past this was done by a macro called `\subst@size`.

```

481     \else

```

This macro substitutes the size recorded in `\best@size` for the unavailable size `\f@size`. `\subst@size` `\font@submax` records the maximum difference between desired size and selected size in the whole run.

```

482 % \subst@size          %% coded inline
483 % \def\subst@size{%
484     \ifx \external@font\@empty
485         \ifx \best@size\@empty
486         \else
487             \ifdim \@tempdimb>\font@submax \relax
488             \xdef \font@submax {\the\@tempdimb}%
489         \fi
490         \let \f@user@size \f@size
491         \let \f@size \best@size
492         \ifdim \@tempdimb>\fontsubfuzz\relax
493             \@font@warning{Font\space shape\space
494                 'curr@fontshape'\space in\space size\space
495                 <\f@user@size>\space not\space available\MessageBreak
496                 size\space <\f@size>\space substituted}%
497         \fi
498         \try@simple@size
499         \do@subst@correction
500     \fi
501 \fi
502 % %}

```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```

503     \let \reserved@f \remove@to@nnil
504     \fi
505 \fi

```

If it's a range iterate also.

```
506 \reserved@f}
```

(End of definition for `\tryis@simple` and `\subst@size`.)

6.1 Sizefunctions

In the following we define some useful size functions.

`\s@fct@` This is the default size function. Mandatory argument is an external font name, optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
507 \DeclareSizeFunction{}\empty@sfcnt\@font@warning}
508 \DeclareSizeFunction{s}\empty@sfcnt\@font@info}

509 \def\empty@sfcnt#1{%
510     \@tempdimb \f@size\p@
511     \ifx\optional@arg\empty
512     \else
513         \@tempdimb \optional@arg\@tempdimb
514         #1{Font\space shape\space '\curr@fontshape'\space
515             will\space be\MessageBreak
516             scaled\space to\space size\space \the\@tempdimb}%
517     \fi
518     \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}
```

(End of definition for `\s@fct@`.)

`\s@fct@gen` This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
519 \DeclareSizeFunction{gen}\gen@sfcnt\@font@warning}
520 \DeclareSizeFunction{sgen}\gen@sfcnt\@font@info}

521 \def\gen@sfcnt{%
522     \edef\mandatory@arg{\mandatory@arg\f@size}%
523     \empty@sfcnt}
```

(End of definition for `\s@fct@gen` and `\s@fct@sgen`.)

`\s@fct@genb` This size function is similar to `gen`, but for fonts where the size is encoded in the font name in centipoints, as in the DC fonts version 1.2. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```
524 \DeclareSizeFunction{genb}\genb@sfcnt\@font@warning}
525 \DeclareSizeFunction{sgenb}\genb@sfcnt\@font@info}

526 \def\genb@sfcnt{%
527     \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@@}%
528     \empty@sfcnt}
```

(End of definition for `\s@fct@genb` and `\s@fct@sgenb`.)

`\genb@x` The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centi-points.

```
529 \def\genb@x#1.#2.#3\@@{\two@digits{#1}\genb@y#200\@@}
530 \def\genb@y#1#2#3\@@{#1#2}
```

(End of definition for `\genb@x` and `\genb@y`.)

`\s@fct@sub` This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```
531 \DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
532 \DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}

533 \def\sub@sfcnt#1{%
534   \edef\mandatory@arg{\f@encoding/\mandatory@arg}%
```

Next action is split the arg into its individual components and allow for a late font shape load.

```
535   \begingroup
536   \expandafter\split@name\mandatory@arg/\@nil
537   \try@load@fontshape
538   \endgroup
```

Then we record the current `\f@size` since it may get clobbered.

```
539   \let\f@user@size\f@size
```

Then we check whether this new combination is defined and give an error message if not. In this case we also switch to `\error@fontshape`.

```
540   \expandafter
541   \ifx\csname\mandatory@arg\endcsname\relax
542     \errmessage{No\space declaration\space for\space
543               shape\space \mandatory@arg}%
544     \error@fontshape
545   \else
```

Otherwise we warn the user about the substitution taking place.

```
546     #1{Font\space shape\space '\curr@fontshape'\space in\space
547       size\space <\f@size>\space not\space available\MessageBreak
548       Font\space shape\space '\mandatory@arg'\space tried\space
549       instead}%
550     \expandafter\split@name\mandatory@arg/\@nil
551   \fi
```

Then we restart the font specification scan by calling `\get@external@font`.

```
552   \edef\f@size{\f@user@size}%
553   \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
554   \do@subst@correction
555 }
```

(End of definition for `\s@fct@sub`.)

`\@font@aliasinfo` Sometimes a substitution is only done to map a long font name to a standard shape or series, e.g.,

```
DeclareFontShape{T1}{Roboto-LF}{b}{it}{<-> alias * Roboto-LF/bold/it}{}
```

Using the `ssub` function in that case will give a strange (and incorrect) warning. As an alternative we therefore offer the size function `alias`. It will still add some info into the `.log` file, but no longer complains that the font shape is not available. It is implemented by grabbing the default warning text and replacing it with a new one.

```
556 </2ekernel>
557 <*2ekernel | latexrelease>
558 <latexrelease>\IncludeInRelease{2020/02/02}%
559 <latexrelease>{\font@aliasinfo}{alias size function}%
560 \DeclareSizeFunction{alias}{\sub@sfcnt\font@aliasinfo}
561 \def\font@aliasinfo#1{%
562   \font@info{Font\space shape\space '\curr@fontshape'\space
563     aliased\space to\MessageBreak '\mandatory@arg'}%
564 }
565 </2ekernel | latexrelease>
566 <latexrelease>\EndIncludeInRelease
567 <latexrelease>\IncludeInRelease{0000/00/00}%
568 <latexrelease>{\font@aliasinfo}{alias size function}%
569 <latexrelease>\let\s@sfcnt@alias\@undefined
570 <latexrelease>\let\font@aliasinfo\@undefined
571 <latexrelease>
572 <latexrelease>\EndIncludeInRelease
573 <*2ekernel>
```

(End of definition for \font@aliasinfo.)

\s@sfcnt@subf The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```
574 \DeclareSizeFunction{subf}{\sub@sfcnt\font@warning}
575 \DeclareSizeFunction{ssubf}{\sub@sfcnt\font@info}
576 \def\sub@sfcnt#1{%
577   #1{Font\space shape\space '\curr@fontshape'\space in\space
578     size\space \f@size\space not\space available\MessageBreak
579     external\space font\space '\mandatory@arg'\space used}%
580   \empty@sfcnt#1%
581 }
```

(End of definition for \s@sfcnt@subf.)

\s@sfcnt@fixed The `fixed` size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the ‘at’ size for the font. Otherwise the font is loaded at its design size.

```
582 \DeclareSizeFunction{fixed}{\fixed@sfcnt\font@warning}
583 \DeclareSizeFunction{sfixed}{\fixed@sfcnt\font@info}
584 \def\fixed@sfcnt#1{%
585   \ifx\optional@arg\@empty
586     \let\external@font\mandatory@arg
587   \else
588     \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
589   \fi
590 }
```

```

589 \fi
590 #1{External\space font\space '\external@font'\space loaded\space
591     for\space size\MessageBreak
592     <\f@size>}%
593 }
594 </2ekernel>

```

(End of definition for \sfct@fixed.)

File 27

ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original ‘NFSS1’ release of the Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was “current” only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via `latexrelease` package so that backward compatibility remains ensured for very old documents.

```
1 <*\latexrelease>
2 \IncludeInRelease{2015/01/01}{\new@fontshape}%
3                                     {NFSS version1 commands}%
4 \let\new@fontshape\@undefined
5 \let\warn@rel@i\@undefined
6 \let\scan@fontshape\@undefined
7 \let\scan@@fontshape\@undefined
8 \let\subst@fontshape\@undefined
9 \let\extra@def\@undefined
10 \let\default@mextra\@undefined
11 \let\preload@sizes\@undefined
12 \let\err@rel@i\@undefined
13 \let\newmathalphabet\@undefined
14 \let\newmathalphabet@\@undefined
15 \let\newmathalphabet@@@\@undefined
16 \let\if@no@font@opt\@undefined
17 \let\@no@font@optfalse\@undefined
18 \let\define@mathalphabet\@undefined
19 \let\define@mathgroup\@undefined
20 \let\addtoversion\@undefined
21 \EndIncludeInRelease
```

In older releases we provide the original definitions.

```
22 \IncludeInRelease{0000/00/00}{\new@fontshape}%
23                                     {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
24 \gdef\new@fontshape#1#2#3#4{%
25     \warn@rel@i\new@fontshape\DeclareFontShape
26     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
27     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
28 \@onlypreamble\new@fontshape
```

(End of definition for \new@fontshape.)

`\warn@rel@i` The warning message used above.

```
29 \gdef\warn@rel@i#1#2{%
30     \@font@warning{*** NFSS release 1 command
```

```

31          \noexpand#1found\MessageBreak
32      *** Update by using release 2 command
33          \string#2.\MessageBreak
34      *** Recovery is probably possible}%
35 }%
36  \@onlypreamble\warn@rel@i

```

(End of definition for \warn@rel@i.)

\scan@fontshape This will scan the old font shape definition syntax.

```

37  \gdef\scan@fontshape{%
38      \let\reserved@f\@empty
39      \let\reserved@e\@empty %           holds last info
40      \scan@@fontshape
41  }%
42  \@onlypreamble\scan@fontshape

```

(End of definition for \scan@fontshape.)

\scan@@fontshape

```

43  \gdef\scan@@fontshape#1>#2#3<{%
44      \ifx\@nil#1%
45          \edef\reserved@f{\reserved@f\reserved@e}%
46      \else
47          \def\reserved@b{#1}%           nick names
48          \def\reserved@c{#3}%
49          \in@{ at}{#3}%
50          \ifin@
51              \in@{pt}{#3}% not a proof but a good chance
52          \ifin@

```

We grab also everything after pt and discard it if people have forgotten to place a percent sign there.

```

53      \def\reserved@a##1 at##2pt##3\@nil{%
54          \def\reserved@b{##2}%
55          \def\reserved@c{##1}%
56      }%
57      \reserved@a#3\@nil
58      \fi
59      \fi
60      \ifnum 0<0#2
61          \edef\reserved@d{subf*\reserved@c}%
62          \ifcase #2\or
63              \or
64              \else
65                  \errmessage{*** What's this? NFSS release 0? ***}%
66              \fi
67          \else
68              \edef\reserved@d{#2\reserved@c}%
69          \fi
70          \ifx\reserved@d\reserved@e
71              \edef\reserved@f{\reserved@f<\reserved@b>}%
72          \else
73              \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
74              \let\reserved@e\reserved@d

```

```

75     \fi
76     \expandafter\scan@@fontshape
77     \fi
78 }%
79 \@onlypreamble\scan@@fontshape

```

(End of definition for \scan@@fontshape.)

\subst@fontshape This is now also handled by the extend syntax of `\DeclareFontShape`.

```

80 \gdef\subst@fontshape#1#2#3#4#5#6{%
81     \warn@rel@i\subst@fontshape\DeclareFontShape
82     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}%
83 \@onlypreamble\subst@fontshape

```

(End of definition for \subst@fontshape.)

\extra@def This was replaced by `\DeclareFontFamily`.

```

84 \gdef\extra@def#1#2#3{%
85     \warn@rel@i\extra@def\DeclareFontFamily
86     \DeclareFontFamily{U}{#1}{}%
87 }%
88 \@onlypreamble\extra@def

```

(End of definition for \extra@def.)

\default@mextra The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```

89 \gdef\default@mextra{%
90     \warn@rel@i\default@mextra\DeclareFontEncodingDefaults

```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```

91     \DeclareFontEncodingDefaults\relax
92 }%
93 \@onlypreamble\default@mextra

```

(End of definition for \default@mextra.)

\preload@sizes The new interface is `\DeclarePreloadSizes`.

```

94 \gdef\preload@sizes{%
95     \warn@rel@i\preload@sizes\DeclarePreloadSizes
96     \DeclarePreloadSizes U%
97 }%
98 \@onlypreamble\preload@sizes

```

(End of definition for \preload@sizes.)

\err@rel@i This macro is used in cases where emulation with NFSS2 features is not really possible.

```

99 \gdef\err@rel@i#1#2{%
100     \latex@error{*** NFSS release 1 command \noexpand#1found%
101         ^^J*** Recovery not possible. Use \string#2}%
102     {The new release of NFSS doesn't support the
103     \noexpand#1command^^Jany longer.
104     Please upgrade your file to the syntax of NFSS
105     release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```
106 \batchmode\input.\relax
107 }%
108 \@onlypreamble\err@rel@i
```

(End of definition for \err@rel@i.)

```
\newmathalphabet \newmathalphabet is the old form.
\newmathalphabet@@ 109 \gdef\newmathalphabet{%
\newmathalphabet@@@ 110 \ifno@font@opt
111 \latex@error{*** NFSS release 1 command
112 \noexpand\newmathalphabet found%
113 ^^J \space*** Automatic recovery not possible.%
114 ^^J \space*** TYPE H for Help%
115 }%
116 {Please look at the file usrguide.tex for hints on
117 how to resolve this problem.}%
118 \else
119 \warn@rel@i\newmathalphabet\DeclareMathAlphabet
120 \fi
121 \@ifstar\newmathalphabet@@@
122 \newmathalphabet@@}%
123 \gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{-}{-}}%
124 \gdef\newmathalphabet@@@#1#2#3#4{%
125 \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
126 \@onlypreamble\newmathalphabet
127 \@onlypreamble\newmathalphabet@@
128 \@onlypreamble\newmathalphabet@@@
```

(End of definition for \newmathalphabet, \newmathalphabet@@, and \newmathalphabet@@@.)

```
\ifno@font@opt
\@no@font@optfalse 129 \global\let\ifno@font@opt\iftrue
130 \gdef\@no@font@optfalse{\let\ifno@font@opt\iffalse}%
131
```

(End of definition for \ifno@font@opt and \@no@font@optfalse.)

```
\define@mathalphabet This is a case where dying is best.
131 \gdef\define@mathalphabet{%
132 \err@rel@i\define@mathalphabet\DeclareMathAlphabet
133 }%
134 \@onlypreamble\define@mathalphabet
```

(End of definition for \define@mathalphabet.)

```
\define@mathgroup And here is another one
135 \gdef\define@mathgroup{%
136 \err@rel@i\define@mathgroup\DeclareSymbolFont
137 }%
138 \@onlypreamble\define@mathgroup
```

(End of definition for \define@mathgroup.)

```

\addtoversion \addtoversion is the old form.

139 \def\addtoversion#1#2{%
140   \warn@rel@i\addtoversion\SetMathAlphabet
141   \SetMathAlphabet#2{#1}{U}}%
142 \@onlypreamble\addtoversion

(End of definition for \addtoversion.)
    Finishing off this huge \IncludeInRelease argument:
143 \EndIncludeInRelease
144 </latexrelease>

```


File 28

ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Interface Commands

```

1 <*2kernel>
2 \message{NFSS declarative interface,}

\in@ \@in is a utility macro with two arguments. It determines whether its first argument
\ifin@ occurs in its second and sets the switch \ifin@ accordingly. The first argument may not
contain braces nor # (more precisely, tokens of category code 1, 2, or 6).

3 \def\in@#1#2%
4 {%
5   \begingroup
6     \def\in@@##1#1{%
7       \toks@\expandafter{\in@@#2{}-{}#1}%
8       \edef\in@@{\the\toks@}%
9       \expandafter\endgroup
10      \ifx\in@@\@empty
11        \in@false
12      \else
13        \in@true
14      \fi
15    }
16 \newif\ifin@

```

(End of definition for \in@ and \ifin@.)

Before the `\begin{document}` command several $\langle\textit{math versions}\rangle$ and $\langle\textit{math alphabet identifiers}\rangle$ may be declared. In principle, there should be exactly one family/series/shape combination be declared for each version/alphabet pair. But we want to allow for defaults as well for automagical filling of holes.

While building the tables for math alphabet identifiers and math versions we keep several lists:

- the list of all math versions, `\version@list`, each entry prefixed by the control sequence `\version@elt`, i.e. this list has the following form

$$\begin{aligned} &\backslash\text{version@elt}\langle\text{version}_1\rangle\backslash\text{version@elt}\langle\text{version}_2\rangle\ldots \\ &\hspace{15em}\backslash\text{version@elt}\langle\text{version}_n\rangle \end{aligned}$$

- the list of all math alphabet identifiers. Here every entry has the form:

$$\begin{aligned} &\backslash\text{group@elt}\langle\textit{math group number}\rangle \\ &\{\{\langle\textit{default family}\rangle\}\{\langle\textit{default series}\rangle\}\{\langle\textit{default shape}\rangle\}\}. \end{aligned}$$

- Each defined math alphabet identifier holds a list containing Information about the *versions* for which it is defined. This list has a more complicated structure: it looks as follows:

```

\set@alpha<the alphabet identifier itself>
  \reserved@c<math version><font info>
  ...
\@nil

```

where ** is either `\reserved@e` (if the combination is not defined yet) or

```

{{<family>}{<series>}{<shape>}}

```

`\version@list` We initialize the version list to be empty.

```

17 \let\version@list=\@empty
18 \@onlypreamble\version@list

```

(End of definition for `\version@list`.)

`\version@elt`

```

19 \let\version@elt\relax
20 \@onlypreamble\version@elt

```

(End of definition for `\version@elt`.)

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```

21 %\def\new@mathversion#1{%

```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```

22 % \expandafter\in@\expandafter#1\expandafter{\version@list}%
23 % \ifin@

```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```

24 % \latex@error{Math version
25 % '\expandafter\@gobblefour\string#1'
26 % already defined}\@eha

```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```

27 % \else
28 % \global\expandafter\newcount\csname c@\expandafter
29 % \gobble\string#1\endcsname
30 % \global\csname c@\expandafter
31 % \gobble\string#1\endcsname\@ne
32 % \def\version@elt{\noexpand\version@elt\noexpand}%
33 % \edef\version@list{\version@list\version@elt#1}%

```

Then we prepare to enter the new version into all math alphabet identifier lists. Remember that these lists use `\reserved@c` as delimiter, and that there appears the control sequence `\reserved@e` that must not be expanded. Therefore we take suitable precautions.

```
34 % \def\reserved@c{\noexpand\reserved@c\noexpand}%
35 % \let\reserved@e\relax
```

We will now go through the `\alpha@list` to process every *math alphabet identifier* in turn. Since this list has `\group@elt` as a delimiter we define this control sequence. It has three arguments as every entry consists of three items (as explained above).

```
36 % \def\group@elt##1##2##3{%
```

The first of these arguments is the *math alphabet identifier*. We redefine it by appending the information about the new version at the end of the list contained in it. However, there is one subtlety: the definitions for `\reserved@c` and `\reserved@e` made above prevent the main part of the list from being expanded. But we still have to take care of the header and the trailer. To do this we remove the trailer by means of the macro `\remove@nil` which also protect the header from being expanded. Its definition is given below. Now we can prepare to add the new version.

```
37 % \edef##1{\expandafter\remove@nil##1%
38 % \reserved@c
39 % #1%
40 % \reserved@e
41 % \noexpand\@nil}}%
```

Finally we call `\alpha@list` which will now execute the macro `\group@elt` once for every defined *math alphabet identifier*. And that's all for now.

```
42 % \alpha@list
43 % \fi}
```

(End of definition for \new@mathversion.)

`\alpha@list` As we explained above every entry in `\alpha@list` has the form

```
\alpha@elt
<alphabet identifier><internal group number><default font assignments>...
```

We initialize it to `\@empty`.

```
44 \let\alpha@list\@empty
45 \@onlypreamble\alpha@list
```

(End of definition for \alpha@list.)

`\alpha@elt`

```
46 \let\alpha@elt\relax
47 \@onlypreamble\alpha@elt
```

(End of definition for \alpha@elt.)

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
48 \count18=-1
```

(End of definition for \newgroup.)

`\stepcounter`

(End of definition for \stepcounter.)

`\select@group` We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`. However, if we use oldstyle syntax where the math alphabet doesn't have arguments (ie if `\math@bgroup` is not `\bgroup`) we need to get rid of the extra group.

```

49 </2ekernel>
50 <latexrelease>\IncludeInRelease{2015/01/01}
51 <latexrelease>          {\select@group}{\select@group}%
52 <*2ekernel | latexrelease>
53 \def\select@group#1#2#3#4{%
54   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
55   {%
56     \ifmmode
57       \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
58         \begingroup
59         \escapechar\m@ne
60         \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
61         \globaldefs\@ne \math@fonts
62         \endgroup
63         \init@restore@version
64         \xdef#1{\noexpand\use@mathgroup\noexpand#2%
65           {\number\csname c@mv@\math@version\endcsname}}}%
66         \global\advance\csname c@mv@\math@version\endcsname\@ne
67       \else
68         \let#1\relax
69         \@latex@error{Too many math alphabets used in
70           version \math@version}%
71         \@eha
72       \fi
73     \else \expandafter\non@alpherr\fi
74     #1{#4}%
75   }%
76 }
77 </2ekernel | latexrelease>
78 <latexrelease>\EndIncludeInRelease
79 <latexrelease>\IncludeInRelease{0000/00/00}
80 <latexrelease>          {\select@group}{\select@group}%
81 <latexrelease>\def\select@group#1#2#3#4{%
82   <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
83   <latexrelease> {%
84     <latexrelease> \ifmmode
85     <latexrelease>   \ifnum\csname c@mv@\math@version\endcsname<\sixt@n
86     <latexrelease>     \begingroup
87     <latexrelease>     \escapechar\m@ne
88     <latexrelease>     \getanddefine@fonts
89     <latexrelease>       {\csname c@mv@\math@version\endcsname}#3%
90     <latexrelease>     \globaldefs\@ne \math@fonts
91     <latexrelease>     \endgroup
92     <latexrelease>     \init@restore@version
93     <latexrelease>     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
94       <latexrelease>       {\number\csname c@mv@\math@version\endcsname}}}%
95     <latexrelease>     \global\advance\csname c@mv@\math@version\endcsname\@ne
96     <latexrelease>   \else
97     <latexrelease>     \let#1\relax
98     <latexrelease>     \@latex@error{Too many math alphabets used in
99     <latexrelease>       version \math@version}%

```

```

100 <latexrelease>      \@eha
101 <latexrelease>      \fi
102 <latexrelease> \else \expandafter\non@alpherr\fi
103 <latexrelease> #1{#4}%
104 <latexrelease> }%
105 <latexrelease>}
106 <latexrelease>\EndIncludeInRelease
107 <*2ekernel>

108 \@onlypreamble\restore@mathversion

```

(End of definition for \select@group.)

\init@restore@version

```

109 \def\init@restore@version{%
110     \global\let\init@restore@version\relax
111     \xdef\restore@mathversion
112         {\expandafter\noexpand\csname mv@\math@version\endcsname
113          \global\csname c@mv@\math@version\endcsname
114          \number\csname c@mv@\math@version\endcsname\relax}%
115     \aftergroup\dorestore@version
116 }
117 \@onlypreamble\init@restore@version

```

(End of definition for \init@restore@version.)

\non@alpherr

```

118 \gdef\non@alpherr#1{\@latex@error{%

```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

```

119     \string#1allowed only in math mode}\@ehd}

```

(End of definition for \non@alpherr.)

\dorestore@version

```

120 \def\dorestore@version
121 {\ifmmode
122     \aftergroup\dorestore@version
123 \else
124     \gdef\init@restore@version{%
125         \global\let\init@restore@version\relax
126         \xdef\restore@mathversion
127             {\expandafter\noexpand\csname mv@\math@version\endcsname
128              \global\csname c@mv@\math@version\endcsname
129              \number\csname c@mv@\math@version\endcsname\relax}%
130         \aftergroup\dorestore@version
131     }%
132     \begingroup
133         \let\getanddefine@fonts\@gobbletwo
134         \restore@mathversion
135     \endgroup
136     \fi}%
137 \@onlypreamble\dorestore@version

```

(End of definition for \dorestore@version.)

`\c@localmathalphabets` To avoid hitting the “no more math fams available” limit of 16, we keep a defined number of math alphabets flexible/local. If we have to allocate any of those we roll back the allocation after the formula has ended, so the next formula can use other alphabets in the slot(s). This makes the processing a bit slower if you are working at the limit, but that is better than dying with “out of memory”.

```

138 \endkernel
139 \latexrelease\IncludeInRelease{2021/11/15}
140 \latexrelease {\document@select@group}{\document@select@group}%
141 \*2ekernel | latexrelease

```

We don’t really undo the declaration on rollback (as that would be hard to maintain), so rolling forward needs to check if the declaration was already made.

```

142 \ifx\c@localmathalphabets\undefined

```

There is no need to have this counter as part of the include checkpoints, given that it makes little sense to alter its settings mid document. All we want is the ability to change it using the `\setcounter` interface.

By default we keep two math fams flexible.

```

143 \newcount\c@localmathalphabets
144 \setcounter{localmathalphabets}{2}
145 \fi

```

(End of definition for \c@localmathalphabets.)

`\document@select@group` The `\document@select@group` command is the version of `\select@group` (inside math versions) that is used in the document body to set up math alphabets (if used).

```

146 \def\document@select@group#1#2#3#4{%
147 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
148 {%
149 \ifmmode

```

We first check if there is still room for allocating another mathgroup. If there is, we check if it can be globally allocated or if we have reached the limit which is given by `\e@mathgroup@top` with `\c@localmathalphabets` subtracted.

```

150 \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
151 \ifnum \numexpr\e@mathgroup@top-\c@localmathalphabets
152 >\csname c@mv@\math@version\endcsname
153 \else

```

If we are past this point we freeze the current state of the math version so that we can return to it after the formula has ended. Of course, that should be done only once, so we check if `\mv@<version>@frozen` already exists.

```

154 \ifcsname mv@\math@version @frozen\endcsname \else

```

We have to pass the current value of `\math@version` not the macro itself, because some of the processing is delayed to a point where the value may have changed again—not doing this caused a puzzling error in one setup.

```

155 \expandafter\freeze@math@version\expandafter{\math@version}%
156 \fi
157 \fi
158 \begingroup
159 \escapechar\m@ne
160 \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
161 \globaldefs\@ne \math@fonts

```

```

162 \endgroup
163 \expandafter\extract@alph@from@version
164 \csname mv@\math@version\expandafter\endcsname
165 \expandafter{\number\csname
166 c@mv@\math@version\endcsname}%
167 #1%
168 \global\advance\csname c@mv@\math@version\endcsname\@ne
169 \else
170 \let#1\relax
171 \@latex@error{Too many math alphabets used in
172 version \math@version}%
173 \@eha
174 \fi

```

Extra \expandafter to remove the \expandafter added below

```

175 \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

We surround \select@group with braces so that functions using it can be used directly after `_` or `^`.

If the legacy interface is used, e.g., `\sf -1` the math alphabet #1 does not take an argument so we better do not surround #4 with braces, because then we get `{\relax}` into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of `\math@bgroup`.

```

176 \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
177 }%
178 }
179 </2ekernel | latexrelease>
180 <latexrelease>\EndIncludeInRelease
181 <latexrelease>\IncludeInRelease{2020/10/01}
182 <latexrelease> {\document@select@group}{\document@select@group}%
183 <latexrelease>
184 <latexrelease>\def\document@select@group#1#2#3#4{%
185 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
186 <latexrelease> {%
187 <latexrelease> \ifmmode
188 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
189 <latexrelease> \begingroup
190 <latexrelease> \escapechar\m@ne
191 <latexrelease> \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
192 <latexrelease> \globaldefs\@ne \math@fonts
193 <latexrelease> \endgroup
194 <latexrelease> \expandafter\extract@alph@from@version
195 <latexrelease> \csname mv@\math@version\expandafter\endcsname
196 <latexrelease> \expandafter{\number\csname
197 <latexrelease> c@mv@\math@version\endcsname}%
198 <latexrelease> #1%
199 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
200 <latexrelease> \else
201 <latexrelease> \let#1\relax
202 <latexrelease> \@latex@error{Too many math alphabets used
203 <latexrelease> in version \math@version}%
204 <latexrelease> \@eha
205 <latexrelease> \fi
206 <latexrelease> \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

```

207 <latexrelease> \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
208 <latexrelease> }%
209 <latexrelease> }
210 <latexrelease> \EndIncludeInRelease
211 <latexrelease> \IncludeInRelease{2015/01/01}
212 <latexrelease> {\document@select@group}{\document@select@group}%
213 <latexrelease>
214 <latexrelease> \def\document@select@group#1#2#3#4{%
215 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
216 <latexrelease> {%
217 <latexrelease> \ifmmode
218 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
219 <latexrelease> \begingroup
220 <latexrelease> \escapechar\m@ne
221 <latexrelease> \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
222 <latexrelease> \globaldefs\@ne \math@fonts
223 <latexrelease> \endgroup
224 <latexrelease> \expandafter\extract@alph@from@version
225 <latexrelease> \csname mv@\math@version\expandafter\endcsname
226 <latexrelease> \expandafter{\number\csname
227 <latexrelease> c@mv@\math@version\endcsname}%
228 <latexrelease> #1%
229 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
230 <latexrelease> \else
231 <latexrelease> \let#1\relax
232 <latexrelease> \@latex@error{Too many math alphabets used
233 <latexrelease> in version \math@version}%
234 <latexrelease> \@eha
235 <latexrelease> \fi
236 <latexrelease> \else \expandafter\non@alpherr\fi
237 <latexrelease> #1{#4}%
238 <latexrelease> }%
239 <latexrelease> }
240 <latexrelease> \EndIncludeInRelease
241 <latexrelease>
242 <latexrelease> \IncludeInRelease{0000/00/00}
243 <latexrelease> {\document@select@group}{\document@select@group}%
244 <latexrelease>
245 <latexrelease> \def\document@select@group#1#2#3#4{%
246 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
247 <latexrelease> {%
248 <latexrelease> \ifmmode
249 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
250 <latexrelease> \begingroup
251 <latexrelease> \escapechar\m@ne
252 <latexrelease> \getanddefine@fonts
253 <latexrelease> {\csname c@mv@\math@version\endcsname}#3%
254 <latexrelease> \globaldefs\@ne \math@fonts
255 <latexrelease> \endgroup
256 <latexrelease> \expandafter\extract@alph@from@version
257 <latexrelease> \csname mv@\math@version\expandafter\endcsname
258 <latexrelease> \expandafter{\number\csname
259 <latexrelease> c@mv@\math@version\endcsname}%
260 <latexrelease> #1%

```



```

261 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
262 <latexrelease> \else
263 <latexrelease> \let#1\relax
264 <latexrelease> \@latex@error{Too many math alphabets used
265 <latexrelease> in version \math@version}%
266 <latexrelease> \@eha
267 <latexrelease> \fi
268 <latexrelease> \else \expandafter\non@alpherr\fi
269 <latexrelease> #1{#4}%
270 <latexrelease> }%
271 <latexrelease> }
272 <latexrelease> \EndIncludeInRelease
273 <*2ekernel>

```

(End of definition for \document@select@group.)

\freeze@math@version This command stores the current state of the math version and sets things up to return to it after each formula from now on. We use L3 programming layer code to set it up.

```

274 </2ekernel>
275 <*2ekernel | latexrelease>
276 <latexrelease> \IncludeInRelease{2022/11/01}%
277 <latexrelease> {\freeze@math@version}{freeze math version}%
278 \ExplSyntaxOn
279 \cs_new_protected:Npn\freeze@math@version #1 {

```

Save the current \mv@<version> code and the number of allocated mathgroups inside.

```

280 \@font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
281 #1.\MessageBreak
282 Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
283 (local:~ \int_use:N\c@localmathalphabets) }
284 \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
285 \tl_gset:ce { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }

```

Here is the definition of \mv@<version>@reset. If there has been no new math alphabet allocation, doing a reset would just cause a lot of unnecessary processing, so we do a quick check upfront for this.

```

286 \cs_gset:cpn{mv@#1@reset}
287 {

```

If we are back at top-level, or more precisely outside of any (nested) math, we may have to reset the math version back to its frozen version, if that exists and has fewer alphabets allocated.

```

288 \int_compare:nNnTF \@math@level = 0
289 {
290 \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
291 { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
292 {
293 \@font@info{ Undo~ math~ alphabet~ allocation~ in~ version~ #1 }

```

If the undo is necessary, we restore the \mv@<version> code.

```

294 \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
295 \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }

```

But we also should undo changes to the top-level math alphabet definitions. We therefore run this code with a modified definition for `\getanddefine@fonts` because there is no need to do anything to the symbol fonts that are permanently allocated. However, we do this only if the resetting was for the current math version, because otherwise we would give the top-level definitions the values for the last math version being reset (e.g., bold if there are just two).

```

296         \tl_if_eq:NnTF \math@version {#1}
297         {
298             \group_begin:
299             \cs_set_eq:NN \getanddefine@fonts \use_none:n
300             \use:c {mv@ \math@version }
301             \group_end:
302         }
303     {

```

Once we have hit the boundary and have to revert to the frozen version that happens quite often, but typically only for the normal version. Thus, it is better to report only when we do not revert the top-level definition as this is the exceptional case and happens only if several math versions are used in parallel.

```

304         \@font@info{ ...~ but~ do~ not~ reset~ the~
305             top-level~ math~ alphabet~ definitions }
306     }
307 }
308 {

```

If there was no change, we report that in the log (but this branch could go completely).

```

309         \@font@info{ No~ math~ alphabet~ change~
310             to~ frozen~ version~ #1 }
311     }
312 }
313 {
314     \@font@info{ Nested~ math:~ keeping~ math~ alphabet~
315         allocation~ in~ version~ #1}
316 }

```

If this is executed after a math display, we may have to arrange for ignoring spaces, because they are now hidden if the tokens from above intervene. This is signaled by the 2e switch `@ignore` which is set in `\frozen@everymath` and `\frozen@everydisplay`.

This is all 2e code so we use that syntax.

```

317     \if@ignore \ignorespaces \fi
318 }
319 }
320 \ExplSyntaxOff
321 \</2ekernel | latexrelease>
322 \<latexrelease>\EndIncludeInRelease
323 \<latexrelease>\IncludeInRelease{2021/11/15}
324 \<latexrelease>         {\freeze@math@version}{freeze math version}%
325 \<latexrelease>
326 \<latexrelease>\ExplSyntaxOn
327 \<latexrelease>\cs_set_protected:Npn\freeze@math@version #1 {
328 \<latexrelease>     \@font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
329 \<latexrelease>         #1.\MessageBreak
330 \<latexrelease>         Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
331 \<latexrelease>         (local:~ \int_use:N\c@localmathalphabets)      }

```

```

332 <latexrelease> \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
333 <latexrelease> \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }
334 <latexrelease> \group_insert_after:N \__nfss_init_mv_freeze:N
335 <latexrelease> \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
336 <latexrelease> \tl_gput_right:No \check@mathfonts
337 <latexrelease> {
338 <latexrelease> \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
339 <latexrelease> }
340 <latexrelease> \cs_gset:cpn{mv@#1@reset}
341 <latexrelease> {
342 <latexrelease> \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
343 <latexrelease> { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
344 <latexrelease> {
345 <latexrelease> \@font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}
346 <latexrelease> \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
347 <latexrelease> \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
348 <latexrelease> \group_begin:
349 <latexrelease> \cs_set_eq:NN \getanddefine@fonts \use_none:nn
350 <latexrelease> \use:c {mv@#1}
351 <latexrelease> \group_end:
352 <latexrelease> }
353 <latexrelease> {
354 <latexrelease> \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
355 <latexrelease> }
356 <latexrelease> \if@ignore \ignorespaces \fi
357 <latexrelease> }
358 <latexrelease>}
359 <latexrelease>\cs_set_protected:Npn \__nfss_init_mv_freeze:N #1 {%
360 <latexrelease> \mode_if_math:T { \group_insert_after:N \__nfss_init_mv_freeze:N
361 <latexrelease> \group_insert_after:N } #1
362 <latexrelease>}
363 <latexrelease>\ExplSyntaxOff
364 <latexrelease>
365 <latexrelease>\EndIncludeInRelease
366 <*2ekernel>

```

(End of definition for \freeze@math@version.)

\process@table

```

367 </2ekernel>
368 <*2ekernel | latexrelease>
369 <latexrelease> \IncludeInRelease{2025/11/01}%
370 <latexrelease> {\process@table}{Support metafamily}%
371 \def\process@table{%
372 \def\cdp@elt##1##2##3##4{%
373 \@font@info{Checking defaults for
374 ##1/##2/##3/##4}%
375 \expandafter
376 \ifx\csname##1/##2/##3/##4\endcsname\relax

```

Grouping is important for two reasons, first \cdp@elt will get redefined if \Declare... functions are executed within the external .fd file and secondly \try@load@fontshape changes a lot of catcodes without surrounding itself with a group.

```

377 \begingroup
378 \def\f@encoding{##1}\def\f@family{##2}%

```

```

379         \try@load@fontshape
380     \endgroup
381 \fi
382 \expandafter
383 \ifx\csname##1/##2/##3/##4\endcsname\relax
384     \@latex@error{This NFSS system isn't set up properly}%
385     {For encoding scheme ##1 the defaults
386     ##2/##3/##4 do not form a valid font shape}%
387 \else
388     \@font@info{... okay}%
389 \fi}%
390 \cldp@list

```

Now we make sure that `\error@fontshape` is okay.

```

391 \begingroup
392     \escapechar\m@ne
393     \error@fontshape
394     \expandafter\ifx\csname \curr@fontshape\endcsname\relax
395         \begingroup
396             \try@load@fontshape
397         \endgroup
398     \fi
399     \expandafter\ifx\csname \curr@fontshape\endcsname\relax
400         \@latex@error{This NFSS system isn't set up properly}%
401         {The system maintainer forgot to specify a suitable
402         substitution
403         font shape using the \noexpand\DeclareErrorFont
404         command}%
405     \fi
406 \endgroup

```

Set `\select@group` to its meaning used within the document body.

```

407 \let\select@group\document@select@group

```

Install the default font attributes as they are currently pointing to error font face. We can speed up the process by just using `\edef`, thereby avoiding all kind of extra processing. Don't use `\reset@font` for this since that would trigger `\selectfont`, but run the `normalfont` hook, since this is really what we are doing here (except for the `\selectfont=`).

```

408 \fontencoding\encodingdefault
409 \edef\f@family{\familydefault}%
410 \edef\f@series{\seriesdefault}%
411 \edef\f@shape{\shapedefault}%

412 \set@current@meta@family
413 \UseHook{normalfont}%

```

Drop stuff not longer needed.

```

414 \everyjob{}%
415 }
416 </2ekernel | latexrelease>
417 <latexrelease>\EndIncludeInRelease

418 <latexrelease>\IncludeInRelease{0000/00/00}%
419 <latexrelease>          {\process@table}{Support metafamily}%
420 <latexrelease>

```

```

421 <latexrelease>\def\process@table{%
422 <latexrelease>    \def\cdp@elt##1##2##3##4{%
423 <latexrelease>        \@font@info{Checking defaults for
424 <latexrelease>            ##1/##2/##3/##4}%
425 <latexrelease>        \expandafter
426 <latexrelease>        \ifx\csname##1/##2/##3/##4\endcsname\relax
427 <latexrelease>            \begingroup
428 <latexrelease>                \def\f@encoding{##1}\def\f@family{##2}%
429 <latexrelease>                \try@load@fontshape
430 <latexrelease>            \endgroup
431 <latexrelease>        \fi
432 <latexrelease>        \expandafter
433 <latexrelease>        \ifx\csname##1/##2/##3/##4\endcsname\relax
434 <latexrelease>            \@latex@error{This NFSS system isn't set up properly}%
435 <latexrelease>                {For encoding scheme ##1 the defaults
436 <latexrelease>                ##2/##3/##4 do not form a valid font shape}%
437 <latexrelease>        \else
438 <latexrelease>            \@font@info{... okay}%
439 <latexrelease>        \fi}%
440 <latexrelease>    \cdp@list
441 <latexrelease>    \begingroup
442 <latexrelease>        \escapechar\m@ne
443 <latexrelease>        \error@fontshape
444 <latexrelease>        \expandafter\ifx\csname \curr@fontshape\endcsname\relax
445 <latexrelease>            \begingroup
446 <latexrelease>                \try@load@fontshape
447 <latexrelease>            \endgroup
448 <latexrelease>        \fi
449 <latexrelease>        \expandafter\ifx\csname \curr@fontshape\endcsname\relax
450 <latexrelease>            \@latex@error{This NFSS system isn't set up properly}%
451 <latexrelease>                {The system maintainer forgot to specify a suitable
452 <latexrelease>                substitution
453 <latexrelease>                font shape using the \noexpand\DeclareErrorFont
454 <latexrelease>                command}%
455 <latexrelease>        \fi
456 <latexrelease>    \endgroup
457 <latexrelease>    \let\select@group\document@select@group
458 <latexrelease>    \fontencoding\encodingdefault
459 <latexrelease>    \edef\f@family{\familydefault}%
460 <latexrelease>    \edef\f@series{\seriesdefault}%
461 <latexrelease>    \edef\f@shape{\shapedefault}%
462 <latexrelease>    \everyjob{}%
463 <latexrelease>}
464 <latexrelease>\EndIncludeInRelease
465 <*2ekernel>

466 \onlypreamble\process@table

(End of definition for \process@table.)

467 %\onlypreamble\set@mathradical

```

\DeclareMathVersion

```

468 </2ekernel>
469 <*2ekernel | latexrelease>

```

```

470 <latexrelease>\IncludeInRelease{2022/11/01}%
471 <latexrelease>          {\DeclareMathVersion}{local alphabets}%
472 \def\DeclareMathVersion#1{%

```

When declaring a new math version we need to instantiate an L3 variable that is used when we freeze the version, because too many alphabets got allocated. If we don't do this, L3 programming layer complains if it is run in checking mode.

```

473   \namedef{g__nfss_frozen_mv_#1_tl}{}%

```

We also extend `\check@mathfonts` to call a version reset (once frozen) after a formula has finished.

```

474   \expandafter\ifx\csname mv@#1\endcsname \relax
475     \expandafter \g@addto@macro \expandafter \check@mathfonts
476       \expandafter {\expandafter \aftergroup \csname mv@#1@reset\endcsname}%

```

Initially this macro does nothing. It is, however, important that it doesn't stop any `\ignorespaces`, so we make it expandable and not `\relax`.

```

477   \namedef{mv@#1@reset}{}%
478 \fi

479   \expandafter\new@mathversion\csname mv@#1\endcsname}
480 \@onlypreamble\DeclareMathVersion
481 </2ekernel | latexrelease>
482 <latexrelease>\EndIncludeInRelease

483 <latexrelease>\IncludeInRelease{2021/11/15}%
484 <latexrelease>          {\DeclareMathVersion}{local alphabets}%
485 <latexrelease>\def\DeclareMathVersion#1{%
486 <latexrelease>   \namedef{g__nfss_frozen_mv_#1_tl}{}%
487 <latexrelease>   \expandafter\new@mathversion\csname mv@#1\endcsname}
488 <latexrelease>\EndIncludeInRelease
489 <latexrelease>\IncludeInRelease{0000/00/00}%
490 <latexrelease>          {\DeclareMathVersion}{local alphabets}%
491 <latexrelease>\def\DeclareMathVersion#1{%
492 <latexrelease>   \expandafter\new@mathversion\csname mv@#1\endcsname}
493 <latexrelease>\EndIncludeInRelease
494 <*2ekernel>

```

(End of definition for \DeclareMathVersion.)

`\new@mathversion`

```

495 \def\new@mathversion#1{%
496   \expandafter\in@\expandafter#1\expandafter{\version@list}%
497   \ifin@
498     \@font@info{Redeclaring math version
499               '\expandafter@gobblefour\string#1'}%
500   \else
501     \expandafter\newcount\csname c@\expandafter
502               \gobble\string#1\endcsname
503     \def\version@elt{\noexpand\version@elt\noexpand}%
504     \edef\version@list{\version@list\version@elt#1}%
505   \fi

```

`\toks@` is used to gather all tokens for the math version. `\count@` will be used to count the math groups we add to this version.

```

506   \toks@{}%
507   \count@\z@

```

Now we loop over `\group@list` to add all math groups defined so far to the version and at the same time to count them.

```

508 \def\group@elt##1##2{%
509     \advance\count@\@ne
510     \addto@hook\toks@{\getanddefine@fonts##1##2}%
511     }%
512 \group@list

```

We set the counter for this math version to the number of math groups found in `\group@list`.

```

513 \global\csname c@\expandafter\@gobble\string#1\endcsname\count@

```

Now we loop over `\alpha@list` to add all math alphabets known so far. We have to distinguish the case that an alphabet by default should produce an error in new versions.

```

514 \def\alpha@elt##1##2##3{%
515     \ifx##2\no@alphabet@error
516     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
517         {\no@alphabet@error##1}}%
518     \else
519     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
520         {\select@group##1##2##3}}%
521     \fi
522     }%
523 \alpha@list

```

Finally we define the math version to expand to the contents of `\toks@`.

```

524 \xdef#1{\the\toks@}%
525 }
526 \@onlypreamble\new@mathversion

```

(End of definition for \new@mathversion.)

`\DeclareSymbolFont` First drop any surplus `m` from the series argument then do what has been done since 1994.

```

527 </2ekernel>
528 <*2ekernel | latexrelease>
529 <latexrelease> \IncludeInRelease{2022/11/01}%
530 <latexrelease> { \DeclareSymbolFont } { maybe drop m } %
531 \def\DeclareSymbolFont #1#2#3#4#5{%
532     \def\reserved@a{\DeclareSymbolFont@m@dropped{#1}{#2}{#3}}%
533     \edef\reserved@b{#4}%
534     \series@maybe@drop@one@m\reserved@b\reserved@b
535     \expandafter\reserved@a\expandafter{\reserved@b}{#5}%
536 }
537 \def\DeclareSymbolFont@m@dropped #1#2#3#4#5{%
538     \@tempswafalse
539     \edef\reserved@b{#2}%
540     \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
541         \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
542     \cdp@list
543     \if@tempswa
544         \@ifundefined{sym#1}{%

```

```

545 \ifnum\count18<15 %
546 \expandafter\new@mathgroup\csname sym#1\endcsname
547 \expandafter\new@symbolfont\csname sym#1\endcsname
548 {#2}{#3}{#4}{#5}%
549 \else
550 \latex@error{Too many symbol fonts declared}\@eha
551 \fi
552 }%
553 {%
554 \@font@info{Redeclaring symbol font ‘#1’}%

```

Update the group list.

```

555 \def\group@elt##1##2{%
556 \noexpand\group@elt\noexpand##1%
557 \expandafter\ifx\csname sym#1\endcsname##1%
558 \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
559 \else
560 \noexpand##2%
561 \fi}%
562 \xdef\group@list{\group@list}%

```

Update the version list.

```

563 \def\version@elt##1{%
564 \expandafter
565 \SetSymbolFont@expandafter##1\csname#2/#3/#4/#5\expandafter
566 \endcsname \csname sym#1\endcsname
567 }%
568 \version@list
569 }%
570 \else
571 \latex@error{Encoding scheme ‘#2’ unknown}\@eha
572 \fi
573 }
574 \@onlypreamble\DeclareSymbolFont
575 </2ekernel | latexrelease>
576 <latexrelease>\EndIncludeInRelease
577 <latexrelease>\IncludeInRelease{0000/00/00}%
578 <latexrelease> \{\DeclareSymbolFont\{maybe drop m\}%
579 <latexrelease>
580 <latexrelease>\let\DeclareSymbolFont\DeclareSymbolFont@m@dropped
581 <latexrelease>\let\DeclareSymbolFont@m@dropped\@undefined
582 <latexrelease>
583 <latexrelease>\EndIncludeInRelease
584 <*2ekernel>

```

(End of definition for \DeclareSymbolFont.)

\group@list

```

585 \let\group@list\@empty
586 \@onlypreamble\group@list

```

(End of definition for \group@list.)

\group@elt

```

587 \let\group@elt\relax
588 \@onlypreamble\group@elt

```


(End of definition for \group@elt.)

\new@symbolfont

```

589 \def\new@symbolfont#1#2#3#4#5{%
590   \toks@\expandafter{\group@list}%
591   \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
592     \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
593   \def\version@elt##1{\toks@\expandafter{##1}%
594     \edef##1{\the\toks@\noexpand\getanddefine@fonts
595       #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
596     \global\advance\csname c@\expandafter
597       \gobble\string##1\endcsname\@ne
598     }%
599   \version@list
600 }
601 \@onlypreamble\new@symbolfont

```

(End of definition for \new@symbolfont.)

\SetSymbolFont First drop any surplus m from the series argument then do what has been done since 1994.

```

602 </2ekernel>
603 <*2ekernel | latexrelease>
604 <latexrelease> \IncludeInRelease{2022/11/01}%
605 <latexrelease> {\SetSymbolFont}{maybe drop m}%
606 \def\SetSymbolFont #1#2#3#4#5#6{%
607   \def\reserved@a{\SetSymbolFont@m@dropped{#1}{#2}{#3}{#4}}%
608   \edef\reserved@b{#5}%
609   \series@maybe@drop@one@m\reserved@b\reserved@b
610   \expandafter\reserved@a\expandafter{\reserved@b}{#6}%
611 }
612 \def\SetSymbolFont@m@dropped#1#2#3#4#5#6{%
613   \@tempswafalse
614   \edef\reserved@b{#3}%
615   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
616     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
617   \cdp@list
618   \if@tempswa
619     \expandafter\SetSymbolFont@
620     \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
621       \endcsname \csname sym#1\endcsname
622   \else
623     \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
624   \fi
625 }
626 \@onlypreamble\SetSymbolFont
627 </2ekernel | latexrelease>
628 <latexrelease> \EndIncludeInRelease
629 <latexrelease> \IncludeInRelease{0000/00/00}%
630 <latexrelease> {\SetSymbolFont}{maybe drop m}%
631 <latexrelease>
632 <latexrelease> \let\SetSymbolFont\SetSymbolFont@m@dropped
633 <latexrelease> \let\SetSymbolFont@m@dropped\@undefined
634 <latexrelease>

```

```

635 \latexrelease\EndIncludeInRelease
636 \*2ekernel)

```

(End of definition for \SetSymbolFont.)

\SetSymbolFont@

```

637 \def\SetSymbolFont@#1#2#3{%
638   \expandafter\in@\expandafter#1\expandafter{\version@list}%
639   \ifin@
640     \expandafter\in@\expandafter#3\expandafter{\group@list}%
641     \ifin@
642       \begingroup
643         \expandafter\get@cdp\string#2\@nil\reserved@a
644         \toks@{}%
645         \def\install@mathalphabet##1##2{%
646           \addto@hook\toks@{\install@mathalphabet##1{##2}}%
647         }%
648         \def\getanddefine@fonts##1##2{%
649           \ifnum##1=#3%
650             \addto@hook\toks@{\getanddefine@fonts#3#2}%
651             \expandafter\get@cdp\string##2\@nil\reserved@b
652             \ifx\reserved@a\reserved@b\else
653               \@font@info{Encoding '\reserved@b' has changed
654                 to '\reserved@a' for symbol font\MessageBreak
655                 '\expandafter\@gobblefour\string#3' in the
656                 math version '\expandafter
657                 \@gobblefour\string#1'}%
658             \fi
659             \@font@info{%
660               Overwriting symbol font
661               '\expandafter\@gobblefour\string#3' in
662               version '\expandafter
663               \@gobblefour\string#1'\MessageBreak
664               \@spaces \expandafter\@gobble\string##2 -->
665               \expandafter\@gobble\string#2}%
666             \else
667               \addto@hook\toks@{\getanddefine@fonts##1##2}%
668             \fi}%
669         #1%
670         \xdef#1{\the\toks@}%
671       \endgroup
672     \else
673       \latex@error{Symbol font '\expandafter\@gobblefour\string#3'
674         not defined}\@eha
675     \fi
676   \else
677     \latex@error{Math version '\expandafter\@gobblefour\string#1'
678       is not
679       defined}{You probably misspelled the name of the math
680       version.^^JOr you have to specify an additional package.}%
681   \fi
682 }
683 \@onlypreamble\SetSymbolFont@

```

(End of definition for \SetSymbolFont.)

`\get@cdp`

```
684 \def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
685 \@onlypreamble\get@cdp
```

(End of definition for \get@cdp.)

`\DeclareMathAlphabet`

```
686 \def\DeclareMathAlphabet#1#2#3#4#5{%
687   \@tempswafalse
688   \edef\reserved@b{#2}%
689   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
690     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
691   \cdp@list
692   \if@tempswa
693     \expandafter\ifx
694       \csname\expandafter\@gobble\string#1\endcsname
695       \relax
696       \new@mathalphabet#1{#2}{#3}{#4}{#5}%
697   \else
```

Check if it is already a math alphabet.

```
698   \edef\reserved@a{\noexpand\in@{\string\select@group}%
699     {\expandafter\meaning\csname \expandafter
700       \@gobble\string#1\space\endcsname}}%
701   \reserved@a
702   \ifin@
703     \@font@info{Redeclaring math alphabet \string#1}%
704     \def\version@elt##1{%
705       \expandafter\SetMathAlphabet@\expandafter
706         ##1\csname#2/#3/#4/#5\expandafter\endcsname
707
708       \csname M@#2\expandafter\endcsname
709       \csname \expandafter\@gobble\string#1\space\endcsname#1}%
710     \version@list
711   \else
```

Check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`.

```
711   \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
712     {\expandafter\meaning\csname \expandafter
713       \@gobble\string#1\space\endcsname}}%
714   \reserved@a
715   \ifin@
```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```
716     \@font@info{Redeclaring math alphabet \string#1}%
717     \new@mathalphabet#1{#2}{#3}{#4}{#5}%
```

Otherwise panic.

```
718   \else
719     \@latex@error{Command ‘\string#1’ already defined}\@eha
720   \fi
721   \fi
722   \fi
723   \else
724     \@latex@error{Encoding scheme ‘#2’ unknown}\@eha
```

```

725 \fi
726 }
727 \@onlypreamble\DeclareMathAlphabet

```

(End of definition for \DeclareMathAlphabet.)

\new@mathalphabet

```

728 \def\new@mathalphabet#1#2#3#4#5{%
729   \toks@{\expandafter{\alpha@list}}%
730   \edef#1{\expandafter\noexpand\csname \expandafter
731     \@gobble\string#1\space\endcsname
732     \if/#5/%
733       \noexpand\no@alphabet@error
734       \noexpand\no@alphabet@error
735     \else
736       \expandafter\noexpand\csname M@#2\endcsname
737       \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
738     \fi
739   }%
740   \toks2\expandafter{#1}%
741   \edef\alpha@list{\the\toks@ \noexpand\alpha@elt\the\toks2}%
742   \def\version@elt##1{\toks@{\expandafter{##1}}%
743     \edef##1{\the\toks@ \install@mathalphabet
744       \expandafter\noexpand
745       \csname \expandafter\@gobble
746         \string#1\space\endcsname
747       {\if/#5/%
748         \noexpand\no@alphabet@error
749         \noexpand#1%
750       \else
751         \noexpand\select@group\the\toks2
752       \fi}}%
753   }%
754   \version@list
755   \expandafter\edef\csname \expandafter\@gobble
756     \string#1\space\endcsname{\if/#5/%
757     \noexpand\no@alphabet@error
758     \noexpand#1%
759   \else
760     \noexpand\select@group\the\toks2
761   \fi}%
762   \edef#1{\noexpand\protect
763     \expandafter\noexpand\csname \expandafter
764       \@gobble\string#1\space\endcsname}%
765 }
766 \@onlypreamble\new@mathalphabet

```

(End of definition for \new@mathalphabet.)

\SetMathAlphabet

```

767 \def\SetMathAlphabet#1#2#3#4#5#6{%
768   \@tempswafalse
769   \edef\reserved@b{#3}%
770   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
771     \ifx\reserved@b\reserved@c \@tempwattrue\fi}%

```

```

772 \cdp@list
773 \if@tempswa
774 \expandafter\SetMathAlphabet@
775 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
776 \endcsname \csname M@#3\expandafter\endcsname
777 \csname \expandafter\@gobble\string#1\space\endcsname#1%
778 \else
779 \@latex@error{Encoding scheme '#3' unknown}\@eha
780 \fi
781 }
782 \@onlypreamble\SetMathAlphabet

```

(End of definition for \SetMathAlphabet.)

\SetMathAlphabet@

```

783 \def\SetMathAlphabet@#1#2#3#4#5{%
784 \expandafter\in@\expandafter#1\expandafter{\version@list}%
785 \ifin@
786 \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
787 \ifin@
788 \begingroup
789 \toks@{%
790 \def\getanddefine@fonts##1##2{%
791 \addto@hook\toks@{\getanddefine@fonts##1##2}%
792 }%
793 \def\reserved@c##1##2##3##4{% % for message below
794 \expandafter\@gobble\string##4}%
795 \def\install@mathalphabet##1##2{%
796 \ifx##1#4%
797 \addto@hook\toks@
798 {\install@mathalphabet#4{\select@group#4#3#2}}%
799 \@font@info{Overwriting math alphabet
800 '\string#5' in version '\expandafter
801 \@gobblefour\string#1'\MessageBreak
802 \@spaces \reserved@c##2 -->
803 \expandafter\@gobble\string#2}%
804 \else
805 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
806 \fi
807 }%
808 #1%
809 \xdef#1{\the\toks@}%
810 \endgroup
811 \else

```

If the math alphabet was defined via \DeclareSymbolFontAlphabet we have remove its external definition and add it as a normal math alphabet to every version before trying to change it in one version.

```

812 \edef\reserved@a{%
813 \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
814 \reserved@a
815 \ifin@
816 \def\reserved@b##1\use@mathgroup##2##3{%
817 \def\reserved@b{##3}\def\reserved@c{##2}}%
818 \expandafter\reserved@b#4%

```

```

819 \begingroup
820 \def\install@mathalphabet##1##2{%
821   \addto@hook\toks@{\install@mathalphabet##1{##2}}%
822   }%
823 \def\getanddefine@fonts##1##2{%
824   \addto@hook\toks@{\getanddefine@fonts##1##2}%
825   \ifnum##1=\reserved@b
826     \expandafter
827     \addto@hook\expandafter\toks@
828     \expandafter\expandafter\install@mathalphabet
829     \expandafter#4\expandafter
830     {\expandafter\select@group\expandafter
831      #4\reserved@c##2}}%
832   \fi
833   }%
834 \def\version@elt##1{%
835   \toks@{}%
836   ##1%
837   \xdef##1{\the\toks@}%
838   }%
839 \version@list
840 \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

841 \expandafter\gdef\expandafter\alpha@list\expandafter
842   {\alpha@list
843    \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
844 \gdef#4{\no@alphabet@error #5}% fake things :-)

```

Then call the internal setting routine again:

```

845 \SetMathAlphabet@{#1}{#2}{#3}#4#5%
846 \else
847   \@latex@error{Command ‘\string#5’ not defined as a
848     math alphabet}%
849   {Use \noexpand\DeclareMathAlphabet to define it.}%
850 \fi
851 \fi
852 \else
853   \@latex@error{Math version ‘\expandafter\@gobblefour\string#1’
854     is not
855     defined}{You probably misspelled the name of the math
856     version.^^JOr you have to specify an additional package.}%
857 \fi
858 }
859 \@onlypreamble\SetMathAlphabet@

```

(End of definition for \SetMathAlphabet@.)

`\DeclareMathAccent` Could do with more checks like allowing single number in #4 lowercase in #4 etc

```

860 </2ekernel>
861 <*2ekernel | latexrelease>
862 <latexrelease> \IncludeInRelease{2019/10/01}%
863 <latexrelease> {DeclareMathAccent}{Make math accents robust}%
864 \def\DeclareMathAccent#1#2#3#4{%
865   \expandafter\in@\csname sym#3\expandafter\endcsname

```

```

866     \expandafter{\group@list}%
867 \ifin@
868     \begingroup
869     \count\z@=#4\relax
870     \count\tw@\count\z@
871     \divide\count\z@\sist@@n
872     \count@\count\z@
873     \multiply\count@\sist@@n
874     \advance\count\tw@-\count@
875     \if\relax\noexpand#1% is command?
876     \edef\reserved@a{\noexpand\in@
877         {\expandafter\@gobble\string\mathaccent}
878         {\expandafter\meaning
879         \csname\expandafter\@gobble\string#1\space\endcsname}}%
880     \reserved@a
881     \ifin@
882     \expandafter\let
883     \csname\expandafter\@gobble\string#1\space\endcsname
884     \@undefined
885     \expandafter\set@mathaccent
886     \csname sym#3\endcsname#1#2%
887     {\hexnumber@\count\z@}\hexnumber@\count\tw@}%
888     \@font@info{Redeclaring math accent \string#1}%
889 \else
890     \expandafter\ifx
891     \csname\expandafter\@gobble\string#1\endcsname
892     \relax
893     \expandafter\set@mathaccent
894     \csname sym#3\endcsname#1#2%
895     {\hexnumber@\count\z@}\hexnumber@\count\tw@}%
896     \else
897     \@latex@error{Command '\string#1' already defined}\@eha
898     \fi
899 \fi
900 \else
901 \@latex@error{Not a command name: '\noexpand#1'}\@eha
902 \fi
903 \endgroup
904 \else
905 \@latex@error{Symbol font '#3' is not defined}\@eha
906 \fi
907 }
908 </2ekernel | latexrelease>
909 <latexrelease>\EndIncludeInRelease
910 <latexrelease>\IncludeInRelease{0000/00/00}%
911 <latexrelease>{DeclareMathAccent}{Make math accents robust}%
912 <latexrelease>\def\DeclareMathAccent#1#2#3#4{%
913 <latexrelease> \expandafter\in@\csname sym#3\expandafter\endcsname
914 <latexrelease> \expandafter{\group@list}%
915 <latexrelease> \ifin@
916 <latexrelease> \begingroup
917 <latexrelease> \count\z@=#4\relax
918 <latexrelease> \count\tw@\count\z@
919 <latexrelease> \divide\count\z@\sist@@n

```

```

920 <latexrelease> \count@ \count\z@
921 <latexrelease> \multiply \count@ \sixt@@n
922 <latexrelease> \advance \count\tw@ - \count@
923 <latexrelease> \if \relax \noexpand#1% is command?
924 <latexrelease> \edef \reserved@a { \noexpand \in@
925 <latexrelease> \expandafter \@gobble \string \mathaccent } { \meaning#1 } }%
926 <latexrelease> \reserved@a
927 <latexrelease> \ifin@
928 <latexrelease> \expandafter \set@mathaccent
929 <latexrelease> \csname sym#3 \endcsname #1#2%
930 <latexrelease> { \hexnumber@ { \count\z@ } \hexnumber@ { \count\tw@ } }%
931 <latexrelease> \@font@info { Redeclaring math accent \string#1 }%
932 <latexrelease> \else
933 <latexrelease> \expandafter \ifx
934 <latexrelease> \csname \expandafter \@gobble \string#1 \endcsname
935 <latexrelease> \relax
936 <latexrelease> \expandafter \set@mathaccent
937 <latexrelease> \csname sym#3 \endcsname #1#2%
938 <latexrelease> { \hexnumber@ { \count\z@ } \hexnumber@ { \count\tw@ } }%
939 <latexrelease> \else
940 <latexrelease> \@latex@error { Command ‘ \string#1 ’ already defined } \@eha
941 <latexrelease> \fi
942 <latexrelease> \fi
943 <latexrelease> \else
944 <latexrelease> \@latex@error { Not a command name: ‘ \noexpand#1 ’ } \@eha
945 <latexrelease> \fi
946 <latexrelease> \endgroup
947 <latexrelease> \else
948 <latexrelease> \@latex@error { Symbol font ‘ #3 ’ is not defined } \@eha
949 <latexrelease> \fi
950 <latexrelease> }
951 <latexrelease> \EndIncludeInRelease
952 <*2ekernel>

```

953 \onlypreamble \DeclareMathAccent

(End of definition for \DeclareMathAccent.)

\set@mathaccent

```

954 </2ekernel>
955 <*2ekernel | latexrelease>
956 <latexrelease> \IncludeInRelease { 2019/10/01 }%
957 <latexrelease> { \set@mathaccent } { makemath accents robust }%
958 \def \set@mathaccent#1#2#3#4{%
959 \xdef#2 { \mathaccent " \mathchar@type#3 \hexnumber@#1#4 \relax }%
960 \MakeRobust#2%
961 }
962 \onlypreamble \set@mathaccent
963 </2ekernel | latexrelease>
964 <latexrelease> \EndIncludeInRelease
965 <latexrelease> \IncludeInRelease { 0000/00/00 }%
966 <latexrelease> { \set@mathaccent } { makemath accents robust }%
967 <latexrelease>
968 <latexrelease> \def \set@mathaccent#1#2#3#4{%
969 <latexrelease> \xdef#2 { \mathaccent " \mathchar@type#3 \hexnumber@#1#4 \relax }%

```



```

970 <latexrelease>
971 <latexrelease>\EndIncludeInRelease
972 <*2ekernel>

```

(End of definition for \set@mathaccent.)

\DeclareMathSymbol

```

973 \def\DeclareMathSymbol#1#2#3#4{%
974   \expandafter\in@\csname sym#3\expandafter\endcsname
975   \expandafter{\group@list}%
976   \ifin@
977     \begingroup
978       \count\z@=#4\relax
979       \count\tw@\count\z@
980       \divide\count\z@\sist@@n
981       \count@\count\z@
982       \multiply\count@\sist@@n
983       \advance\count\tw@-\count@
984       \if\relax\noexpand#1% is command?

```

Store the command name with a space attached inside \reserved@@b in case we look at a robust definition.

```

985       \edef\reserved@b{\expandafter\noexpand
986         \csname\expandafter@gobble\string#1\space\endcsname}%

```

Test both #1 and #1_␣ for containing mathchar.

```

987       \edef\reserved@a
988         {\noexpand\in@{\expandafter@gobble\string\mathchar}%
989          {\meaning#1\expandafter\meaning\reserved@b}}%
990       \reserved@a

```

Drop #1_␣ in case it was defined before.

```

991       \global\expandafter\let\reserved@b\@undefined
992       \ifin@
993         \expandafter\set@mathsymbol
994         \csname sym#3\endcsname#1#2%
995         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
996         \@font@info{Redeclaring math symbol \string#1}%
997       \else
998         \expandafter\ifx
999         \csname\expandafter@gobble\string#1\endcsname
1000         \relax
1001         \expandafter\set@mathsymbol
1002         \csname sym#3\endcsname#1#2%
1003         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1004       \else
1005         \@latex@error{Command ‘\string#1’ already defined}\@eha
1006       \fi
1007     \fi
1008   \else
1009     \expandafter\set@mathchar
1010     \csname sym#3\endcsname#1#2
1011     {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1012   \fi
1013 \endgroup

```

```

1014 \else
1015 \latexerror{Symbol font ‘#3’ is not defined}\@eha
1016 \fi
1017 }
1018 \@onlypreamble\DeclareMathSymbol

```

(End of definition for \DeclareMathSymbol.)

\set@mathchar

```

1019 \def\set@mathchar#1#2#3#4{%
1020 \global\mathcode‘#2="\mathchar@type#3\hexnumber@#1#4\relax}
1021 \@onlypreamble\set@mathchar

```

(End of definition for \set@mathchar.)

\set@mathsymbol

```

1022 \def\set@mathsymbol#1#2#3#4{%
1023 \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
1024 \@onlypreamble\set@mathsymbol

```

(End of definition for \set@mathsymbol.)

```

1025 %\def\mathsymbol#1#2#3{%
1026 % \@tempcnta=#3\relax
1027 % \@tempcntb\@tempcnta
1028 % \divide\@tempcnta\sixt@@n
1029 % \count@\@tempcnta
1030 % \multiply\count@\sixt@@n
1031 % \advance\@tempcntb-\count@
1032 % \mathchar"\mathchar@type#1\hexnumber@#2%
1033 % \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}
1034 %
1035 %\def\DeclareMathAlphabetCharacter#1#2#3{%
1036 % \DeclareMathSymbol{#1}7{#2}{#3}

```

\DeclareMathDelimiter

```

1037 \def\DeclareMathDelimiter#1{%
1038 \if\relax\noexpand#1%
1039 \expandafter\@DeclareMathDelimiter
1040 \else
1041 \expandafter\@xxDeclareMathDelimiter
1042 \fi
1043 #1}
1044 \@onlypreamble\DeclareMathDelimiter

```

(End of definition for \DeclareMathDelimiter.)

\@xxDeclareMathDelimiter

This macro checks if the second arg is a “math type” such as `\mathopen`. The undocumented original code didn’t use math types when the delimiter was a single letter. For this reason the coding is a bit strange as it tries to support the undocumented syntax for compatibility reasons.

```

1045 \def\@xxDeclareMathDelimiter#1#2#3#4{%

```

7 is the default value returned in the case that `\mathchar@type` is passed something unexpected, like a math symbol font name. We locally move `\mathalpha` out of the way so if you use that the right branch is taken. This will still fail if an explicit number 7 is used!

```
1046 \begingroup
1047 \let\mathalpha\mathord
1048 \ifnum7=\mathchar@type{#2}%
1049 \endgroup
```

If this branch is taken we have old syntax (5 arguments).

```
1050 \expandafter\@firstofone
1051 \else
```

If this branch is taken `\mathchar@type` is different from 7 so we assume new syntax. In this case we also use the arguments to set up the letter as a math symbol for the case where it is not used as a delimiter.

```
1052 \endgroup
1053 \DeclareMathSymbol#1{#2}{#3}{#4}%
```

Then we arrange that `\@xDeclareMathDelimiter` only gets #1, #3, #4 ... as it does not expect a math type as argument.

```
1054 \expandafter\@firstoftwo
1055 \fi
1056 {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
1057 \@onlypreamble\@xxDeclareMathDelimiter
```

(End of definition for \@xxDeclareMathDelimiter.)

`\@DeclareMathDelimiter`

```
1058 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
1059 \expandafter\in@\csname sym#3\expandafter\endcsname
1060 \expandafter{\group@list}%
1061 \ifin@
1062 \expandafter\in@\csname sym#5\expandafter\endcsname
1063 \expandafter{\group@list}%
1064 \ifin@
1065 \begingroup
1066 \count\z@=#4\relax
1067 \count\tw@\count\z@
1068 \divide\count\z@\sist@@n
1069 \count@\count\z@
1070 \multiply\count@\sist@@n
1071 \advance\count\tw@-\count@
1072 \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1073 %
1074 \count\z@=#6\relax
1075 \count\tw@\count\z@
1076 \divide\count\z@\sist@@n
1077 \count@\count\z@
1078 \multiply\count@\sist@@n
1079 \advance\count\tw@-\count@
1080 \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1081 %
1082 \edef\reserved@a{\noexpand\in@
1083 {\expandafter\@gobble\string\delimiter}{\meaning#1}}%
```

```

1084     \reserved@a
1085     \ifin@
1086         \expandafter\set@mathdelimiter
1087             \csname sym#3\expandafter\endcsname
1088             \csname sym#5\endcsname#1#2%
1089             \reserved@c\reserved@d
1090             \@font@info{Redeclaring math delimiter \string#1}%
1091     \else
1092         \expandafter\ifx
1093             \csname\expandafter\@gobble\string#1\endcsname
1094             \relax
1095         \expandafter\set@mathdelimiter
1096             \csname sym#3\expandafter\endcsname
1097             \csname sym#5\endcsname#1#2%
1098             \reserved@c\reserved@d
1099     \else
1100         \@latex@error{Command ‘\string#1’ already defined}\@eha
1101     \fi
1102 \fi
1103 \endgroup
1104 \else
1105     \@latex@error{Symbol font ‘#5’ is not defined}\@eha
1106 \fi
1107 \else
1108     \@latex@error{Symbol font ‘#3’ is not defined}\@eha
1109 \fi
1110 }

```

1111 \@onlypreamble\@DeclareMathDelimiter

(End of definition for \@DeclareMathDelimiter.)

\@xDeclareMathDelimiter

```

1112 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
1113     \expandafter\in@\csname sym#2\expandafter\endcsname
1114     \expandafter{\group@list}%
1115     \ifin@
1116         \expandafter\in@\csname sym#4\expandafter\endcsname
1117         \expandafter{\group@list}%
1118     \ifin@
1119         \begingroup
1120             \count\z@=#3\relax
1121             \count\tw@\count\z@
1122             \divide\count\z@\sixt@@n
1123             \count@\count\z@
1124             \multiply\count@\sixt@@n
1125             \advance\count\tw@-\count@
1126             \edef\reserved@c{\hexnumber@\count\z@}\hexnumber@\count\tw@}%
1127         %
1128         \count\z@=#5\relax
1129         \count\tw@\count\z@
1130         \divide\count\z@\sixt@@n
1131         \count@\count\z@
1132         \multiply\count@\sixt@@n
1133         \advance\count\tw@-\count@

```

```

1134 \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1135 \expandafter\set@mathdelimiter
1136 \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
1137 \reserved@c\reserved@d
1138 \endgroup
1139 \else
1140 \@latex@error{Symbol font ‘#4’ is not defined}\@eha
1141 \fi
1142 \else
1143 \@latex@error{Symbol font ‘#2’ is not defined}\@eha
1144 \fi
1145 }
1146 \@onlypreamble\@xDeclareMathDelimiter

```

(End of definition for \@xDeclareMathDelimiter.)

`\set@mathdelimiter` We have to end the definition of a math delimiter like `\lfloor` with a space and not with `\relax` as we did before, because otherwise constructs involving `\abovewithdelims` will prematurely end (pr/1329)

```

1147 </2ekernel>
1148 <*2ekernel | latexrelease>
1149 <latexrelease>\IncludeInRelease{2019/10/01}%
1150 <latexrelease> \set@mathdelimiter}{make delimiters robust}%
1151 \def\set@mathdelimiter#1#2#3#4#5#6{%

```

We use `\protected` not `\MakeRobust` so that `\bigl\lfloor` etc. works inside the argument of `\protected@edef`.

```

1152 \protected
1153 \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1154 \hexnumber@#2#6 }%
1155 % \MakeRobust#3%
1156 }
1157 \@onlypreamble\set@mathdelimiter
1158 </2ekernel | latexrelease>
1159 <latexrelease>\EndIncludeInRelease
1160 <latexrelease>\IncludeInRelease{0000/00/00}%
1161 <latexrelease> \set@mathdelimiter}{make delimiters robust}%
1162 <latexrelease>
1163 <latexrelease>\def\set@mathdelimiter#1#2#3#4#5#6{%
1164 <latexrelease> \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1165 <latexrelease> \hexnumber@#2#6 }}
1166 <latexrelease>
1167 <latexrelease>\EndIncludeInRelease
1168 <*2ekernel>

```

(End of definition for \set@mathdelimiter.)

`\set@@mathdelimiter`

```

1169 \def\set@@mathdelimiter#1#2#3#4#5{%
1170 \global\delcode‘#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
1171 \@onlypreamble\set@@mathdelimiter

```

(End of definition for \set@@mathdelimiter.)

`\DeclareMathRadical`

1172 `\def\DeclareMathRadical#1#2#3#4#5{%`
 Below is a crude fix to make this macro work if #1 is undefined or `\relax`. Should be improved!

```
1173   \expandafter\ifx
1174     \csname\expandafter\@gobble\string#1\endcsname
1175     \relax
1176     \let#1\radical
1177   \fi
1178   \edef\reserved@a{\noexpand\in@
1179     {\expandafter\@gobble\string\radical}{\meaning#1}}%
1180   \reserved@a
1181   \ifin@
1182     \expandafter\in@\csname sym#2\expandafter\endcsname
1183     \expandafter{\group@list}%
1184   \ifin@
1185     \expandafter\in@\csname sym#4\expandafter\endcsname
1186     \expandafter{\group@list}%
1187   \ifin@
1188     \begingroup
1189       \count\z@=#3\relax
1190       \count\tw@\count\z@
1191       \divide\count\z@\sist@@n
1192       \count@\count\z@
1193       \multiply\count@\sist@@n
1194       \advance\count\tw@-\count@
1195       \edef\reserved@c{%
1196         \hexnumber@\count\z@}\hexnumber@\count\tw@}%
1197       \count\z@=#5\relax
1198       \count\tw@\count\z@
1199       \divide\count\z@\sist@@n
1200       \count@\count\z@
1201       \multiply\count@\sist@@n
1202       \advance\count\tw@-\count@
1203       \edef\reserved@d{%
1204         \hexnumber@\count\z@}\hexnumber@\count\tw@}%
1205
```

Coded inline instead of using `\set@mathradical`

```
1205 %       \expandafter\set@mathradical
1206 %       \csname sym#2\expandafter\endcsname
1207 %       \csname sym#4\endcsname#1%
1208 %       \reserved@c\reserved@d
1209       \xdef#1{\radical"\expandafter\hexnumber@
1210         \csname sym#2\endcsname\reserved@c
1211         \expandafter\hexnumber@
1212         \csname sym#4\endcsname\reserved@d
1213         \relax}%
1214     \endgroup
1215   \else
1216     \@latex@error{Symbol font ‘#4’ is not defined}\@eha
1217   \fi
1218 \else
1219   \@latex@error{Symbol font ‘#2’ is not defined}\@eha
1220 \fi
```

```

1221 \else
1222 \latexerror{Command '\string#1' already defined}\@eha
1223 \fi
1224 }
1225 \@onlypreamble\DeclareMathRadical

(End of definition for \DeclareMathRadical.)
Definition below was wrong it contained \delimiter !

def\set@mathradical#1#2#3#4#5{%
\xdef#3{\radical"\hexnumber@#1#4\hexnumber@#2#5\relax}}

\mathalpha just a dummy currently
1226 \let\mathalpha\relax

(End of definition for \mathalpha.)

\mathchar@type
1227 \def\mathchar@type#1{%
1228 \ifodd 2#1#1 #1\else % is this non-negative number?
1229 \ifx#1\mathord 0\else
1230 \ifx#1\mathop 1\else
1231 \ifx#1\mathbin 2\else
1232 \ifx#1\mathrel 3\else
1233 \ifx#1\mathopen 4\else
1234 \ifx#1\mathclose 5\else
1235 \ifx#1\mathpunct 6\else
1236 7% % anything else is variable ord
1237 \fi
1238 \fi
1239 \fi
1240 \fi
1241 \fi
1242 \fi
1243 \fi
1244 \fi}
1245 \@onlypreamble\mathchar@type

(End of definition for \mathchar@type.)

\DeclareSymbolFontAlphabet

1246 \def\DeclareSymbolFontAlphabet#1#2{%
1247 \expandafter\DeclareSymbolFontAlphabet@
1248 \csname \expandafter\@gobble\string#1\space\endcsname{#2}#1}
1249 \@onlypreamble\DeclareSymbolFontAlphabet

(End of definition for \DeclareSymbolFontAlphabet.)

\DeclareSymbolFontAlphabet@
1250 \def\DeclareSymbolFontAlphabet@#1#2#3{%
We use the switch \if@tempswa to decide if we can declare this symbol font alphabet.
1251 \if@tempswatru

```

First check if #2 is known to be a symbol font

```
1252 \expandafter\in@\csname sym#2\expandafter\endcsname
1253 \expandafter{\group@list}%
1254 \ifin@
```

Check if #1 is defined as a math alphabet defined via \DeclareMathAlphabet:

```
1255 \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
1256 \ifin@
```

If so remove it from the \alpha@list and from all math version macros.

```
1257 \@font@info{Redeclaring math alphabet \string#3}%
1258 \toks@{}%
1259 \def\alpha@elt##1##2##3{%
1260 \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
1261 \alpha@list
1262 \xdef\alpha@list{\the\toks@}%

```

Now we loop over all versions and remove the math alphabet:

```
1263 \def\version@elt##1{%
1264 \begingroup
1265 \toks@{}%
1266 \def\getanddefine@fonts####1####2{%
1267 \addto@hook\toks@{\getanddefine@fonts####1####2}}%
1268 \def\install@mathalphabet####1####2{%
1269 \ifx####1#1\else
1270 \addto@hook\toks@{\install@mathalphabet
1271 #####1{####2}}\fi}%
1272 ##1%
1273 \xdef##1{\the\toks@}%
1274 \endgroup
1275 }%
1276 \version@list
1277 \else

```

If #3 is not defined as a math alphabet check if it is defined at all:

```
1278 \expandafter\ifx
1279 \csname\expandafter\@gobble\string#1\space\endcsname
1280 \relax

```

If it is undefined, fine otherwise check if it is a math alphabet defined via \DeclareSymbolFontAlphabet:

```
1281 \else
1282 \edef\reserved@a{%
1283 \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
1284 \reserved@a
1285 \ifin@
1286 \@font@info{Redeclaring math alphabet \string#3}%
1287 \else

```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
1288 \@tempswafalse
1289 \@latex@error{Command ‘\string#3’ already defined}\@eha
1290 \fi
1291 \fi
1292 \fi
1293 \else

```


Since the symbol font is not known we better skip defining this alphabet.

```

1294     \@tempswafalse
1295     \@latex@error{Unknown symbol font '#2'}\@eha
1296     \fi
1297     \if@tempswa

```

When we reach this point we are allowed to define #1 to be a symbol font math alphabet. This means that we have to set it to

```
\use@mathgroup <math-settings> \sym<name>
```

The `<math-settings>` are the one for the encoding that is used in the font shape where `\sym<name>` is pointing to. This means that we have to get it from the information stored in `\group@list`. Thus we loop through that list after defining `\group@elt` in a suitable way.

```

1298     \def\group@elt##1##2{%
1299         \expandafter\ifx\csname sym#2\endcsname##1%
1300         \expandafter\reserved@a\string##2\@nil
1301         \fi}%
1302     \def\reserved@a###2/##3\@nil{%
1303         \def\reserved@a{##2}}%
1304     \group@list
1305     \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
1306     \edef#1{\the\toks@
1307         \noexpand\use@mathgroup
1308         \expandafter\noexpand\csname M@\reserved@a\endcsname
1309         \csname sym#2\endcsname}%
1310     \def#3{\protect#1}%
1311     \fi
1312 }
1313 \@onlypreamble\DeclareSymbolFontAlphabet@
1314 </2ekernel>

```

(End of definition for \DeclareSymbolFontAlphabet@.)

File 29

ltfssini.dtx

This file contains the top level L^AT_EX interface to the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

1 NFSS Initialization

Finally, there are six commands that are to be used in L^AT_EX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

```
1 <*2kernel>
2 \message{NFSS initialization,}
```

1.1 Providing math *versions*

L^AT_EX provides two *versions*. We call them normal and bold, respectively.

```
3 \DeclareMathVersion{normal}
4 \DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of `\rmfamily` etc. in math mode.

(Actually most are now defined further down in the file.)

First the changes to another *family*:

```
5 %\DeclareRobustCommand\rmfamily
6 %      {\not@math@alphabet\rmfamily\mathrm}
7 %      \fontfamily\rmdefault\selectfont}
8 %\DeclareRobustCommand\sffamily
9 %      {\not@math@alphabet\sffamily\mathsf}
10 %      \fontfamily\sfddefault\selectfont}
11 %\DeclareRobustCommand\ttfamily
12 %      {\not@math@alphabet\ttfamily\mathtt}
13 %      \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
14 %\DeclareRobustCommand\bfseries
15 %      {\not@math@alphabet\bfseries\mathbf}
16 %      \fontseries\bfdefault\selectfont}
17 %\DeclareRobustCommand\mdseries
18 %      {\not@math@alphabet\mdseries\relax}
19 %      \fontseries\mddefault\selectfont}
20 \DeclareRobustCommand\upshape
21      {\not@math@alphabet\upshape\relax}
22      \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
23 \DeclareRobustCommand\slshape
24      {\not@math@alphabet\slshape\relax}
25      \fontshape\sldefault\selectfont}
26 \DeclareRobustCommand\scshape
```

```

27         {\not@math@alphabet\scshape\relax
28          \fontshape\scdefault\selectfont}
29 \DeclareRobustCommand\itshape
30     {\not@math@alphabet\itshape\mathit
31      \fontshape\itdefault\selectfont}

```

2 Custom series settings for main document families

This section was introduced 2020/02/02 and for now we support a full rollback (may need splitting later).

```

32 </2ekernel>
33 <*2ekernel | latexrelease>
34 <latexrelease> \IncludeInRelease{2021/11/15}%
35 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%

```

One problem with the NFSS approach of handling the series axis turned out to be that (especially with respect to “boldness”) different font families implemented different strategies. For example, with Computer Modern fonts you normally only have **bx** whereas most PostScript fonts offered only **b** but not **bx**. As a result L^AT_EX’s standard setting for `\bfdefault` didn’t work with such fonts, but if it got changed to produce **b**, then that didn’t work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font `.fd` such that if a **bx** series got requested the **b** series got used. While this works in that particular case, it isn’t a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for “rm”, “sf” and “tt” so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault` We now support this concept directly from within L^AT_EX and for use in font packages (or the document preamble) we offer `\DeclareFontSeriesDefault`. This declaration takes three arguments:

document family interface: Can either be `rm`, `sf` or `tt`. This is optional and if not given the overall default.

document series interface: Can be `md` or `bf`.

series value: This is the value that is going to be used with the combination is requested.

For example, `\DeclareFontSeriesDefault[rm]{bf}{sb}` would use **sb** (semi-bold) when `\rmfamily \bfseries` is asked for.

If used without the optional argument, e.g., `\DeclareFontSeriesDefault{bf}{b}` then this is like redefining `\bfdefault` or `\mddefault`.

If some family specific defaults aren't given, e.g. if there are no declarations for, say, `tt` then the format defaults of `\mddefault` and `\bfdefault` are assumed. If those are later changed this is *not* reflected!³⁷

`\DeclareFontSeriesDefault` The command to declare font series defaults for the “rm”, “sf” or “tt” family.

```

36 \let\DeclareFontSeriesDefault\undefined      % for rollback
37 \newcommand\DeclareFontSeriesDefault[3][]{%

38   \expand@font@defaults
39   \maybe@update@bfseries@defaults
40   \maybe@update@mdseries@defaults

41   \def\reserved@a{#1}%

```

No optional argument: set up general default.

```

42   \ifx\reserved@a\@empty
43     \ifcsname #2series\endcsname              % supported are
44                                           % \[md/bf]default

```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```

45     \expandafter\def
46       \csname #2default\endcsname{#3\@empty}%
47     \expandafter\def
48       \csname #2default@previous\endcsname{#3\@empty}%
49   \else

50     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
51     {Mandatory first argument must be 'md' or 'bf'.}
52   \fi

```

Optional argument given, set up specific default.

```

53   \else
54     \ifcsname #2series@#1\endcsname          % supported are
55                                           % \[md/bf]series@[rm/sf/tt]

56     \expandafter\edef
57       \csname #2series@#1\endcsname{#3}%

```

If the interface is used we remove the frozen kernel default. This way, we know that something was explicitly set up (even if the setup has the same value as the default).

```

58     \expandafter\let
59       \csname #2series@#1@kernel\endcsname\@undefined
60   \else
61     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
62     {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
63       Mandatory first argument must be 'md' or 'bf'.}
64   \fi
65 \fi
66 }

```

³⁷I see no easy way to achieve this without compromising compatibility with existing packages that currently use `mweights` and directly define (some) of the `\mdseries@...` commands but not others.

(End of definition for \DeclareFontSeriesDefault.)

```
67 </2ekernel | latexrelease>
68 <latexrelease>\EndIncludeInRelease
69 <latexrelease>\IncludeInRelease{2020/02/02}%
70 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%
71 <latexrelease>
72 <latexrelease>\let\DeclareFontSeriesDefault\@undefined          % for rollback
73 <latexrelease>\newcommand\DeclareFontSeriesDefault[3] [] {%
74 <latexrelease>  \def\reserved@a{#1}%
75 <latexrelease>  \ifx\reserved@a\@empty
76 <latexrelease>    \ifcsname #2series\endcsname          % supported are
77 <latexrelease>                                     % \[md/bf]default
78 <latexrelease>    \expandafter\def
79 <latexrelease>        \csname #2default\endcsname{#3\@empty}%
80 <latexrelease>    \expandafter\def
81 <latexrelease>        \csname #2default@previous\endcsname{#3\@empty}%
82 <latexrelease>  \else
83 <latexrelease>    \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
84 <latexrelease>        {Mandatory first argument must be 'md' or 'bf'.}
85 <latexrelease>  \fi
86 <latexrelease> \else
87 <latexrelease>  \ifcsname #2series@#1\endcsname          % supported are
88 <latexrelease>                                     % \[md/bf]series@[rm/sf/tt]
89 <latexrelease>    \expandafter\edef
90 <latexrelease>        \csname #2series@#1\endcsname{#3}%
91 <latexrelease>    \expandafter\let
92 <latexrelease>        \csname #2series@#1@kernel\endcsname\@undefined
93 <latexrelease>  \else
94 <latexrelease>    \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
95 <latexrelease>        {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
96 <latexrelease>        Mandatory first argument must be 'md' or 'bf'.}
97 <latexrelease>  \fi
98 <latexrelease> \fi
99 <latexrelease>}
100 <latexrelease>
101 <latexrelease>\EndIncludeInRelease
102 <latexrelease>\IncludeInRelease{0000/00/00}%
103 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%
104 <latexrelease>
105 <latexrelease>\let\DeclareFontSeriesDefault\@undefined
106 <latexrelease>\let\bfseries@rm\@undefined
107 <latexrelease>\let\bfseries@sf\@undefined
108 <latexrelease>\let\bfseries@tt\@undefined
109 <latexrelease>\let\bfseries@rm@kernel\@undefined
110 <latexrelease>\let\bfseries@sf@kernel\@undefined
111 <latexrelease>\let\bfseries@tt@kernel\@undefined
112 <latexrelease>\let\mdseries@rm\@undefined
113 <latexrelease>\let\mdseries@sf\@undefined
114 <latexrelease>\let\mdseries@tt\@undefined
115 <latexrelease>\expandafter\let\csname ver@weights.sty\endcsname\@undefined
116 <latexrelease>
117 <latexrelease>\let\@meta@family@list\@undefined
118 <latexrelease>\let\prepare@family@series@update\@undefined
119 <latexrelease>\let@update@series@target@value\@undefined
```

120 <latexrelease>

This is always called in `\document` so don't make it undefined.

121 <latexrelease>\let\init@series@setup\relax

122 <latexrelease>

123 <latexrelease>\EndIncludeInRelease

124 <*2ekernel>

125 </2ekernel>

126 <*2ekernel | latexrelease>

127 <latexrelease>\IncludeInRelease{2020/02/02}%

128 <latexrelease>{\mdseries@rm}{Custom series}%

`\mdseries@rm` We initialize the family specific default at the end of the format generation. Later on
`\mdseries@sf` they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault`
`\mdseries@tt` (or possibly directly).

`\bfseries@rm` Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should
`\bfseries@sf` be really for nearly every font except Computer/Latin Modern.

`\bfseries@tt` To account for the fact that by default we typeset in CM or LM we set up the
`\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then
`\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed
then `\bfdefault` will get used.

129 \def\bfseries@rm{bx}

130 \def\bfseries@sf{bx}

131 \def\bfseries@tt{bx}

Frozen version of the kernel defaults so we can see if they have changed.

132 \let\bfseries@rm@kernel\bfseries@rm

133 \let\bfseries@sf@kernel\bfseries@sf

134 \let\bfseries@tt@kernel\bfseries@tt

The default for the medium series is `m` and this will be interpreted as resetting both
weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

135 \def\mdseries@rm{m}

136 \def\mdseries@sf{m}

137 \def\mdseries@tt{m}

(End of definition for `\mdseries@rm` and others.)

`\series@change@debug` For debugging, but right now none of this code is extracted. The idea is to have a separate
package with debugging code one day.

138 <*debug>

139 \let\series@change@debug\typeout

140 \let\series@change@debug@gobble

141 </debug>

(End of definition for `\series@change@debug`.)

142 </2ekernel | latexrelease>

143 <latexrelease>\EndIncludeInRelease

144 <latexrelease>\IncludeInRelease{0000/00/00}%

145 <latexrelease>{\mdseries@rm}{Custom series}%

146 <latexrelease>

147 <latexrelease>\let\bfseries@rm\@undefined

```

148 <latexrelease>\let\bfseries@sf\@undefined
149 <latexrelease>\let\bfseries@tt\@undefined
150 <latexrelease>\let\bfseries@rm@kernel\@undefined
151 <latexrelease>\let\bfseries@sf@kernel\@undefined
152 <latexrelease>\let\bfseries@tt@kernel\@undefined
153 <latexrelease>\let\mdseries@rm\@undefined
154 <latexrelease>\let\mdseries@sf\@undefined
155 <latexrelease>\let\mdseries@tt\@undefined
156 <latexrelease>\EndIncludeInRelease
157 <*2ekernel>

158 </2ekernel>
159 <*2ekernel | latexrelease>
160 <latexrelease>\IncludeInRelease{2025/11/01}%
161 <latexrelease>    {\prepare@family@series@update}{Support meta family}%

```

`\prepare@family@series@update` This is core command that prepares for the family update. The big difference to the documented code above is that the nested `\ifx` statements seem to be missing. Instead we loop through an internal list that holds the names of the three meta families. This approach allows us to extend the mechanism at a later stage to allow for additional named meta families.

Here is the current definition of that list:

```

\@meta@family@list 162 \def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}

```

This macro holds the current “meta” family. At the moment it is either `rm`, `sf`, `tt` or `??`.

`\@currentmetafamily` The latter is used if the current font family does not match any of the document meta families, defined through `\rmdefault`, `\sfdefault` or `\ttdefault`, i.e., in the case that yet another family was selected using `\fontfamily` manually.

The value gets updated by commands like `\rmfamily`, etc.

```

163 \def\@currentmetafamily{??}

164 \def\prepare@family@series@update#1#2{%

```

We start by updating the meta family.

```

165 \def\@currentmetafamily{#1}%

166 \if@forced@series
167 <+debug> \series@change@debug{No series preparation (forced \f@series)\on@line}%
168 \fontfamily#2%
169 \else
170 <+debug> \series@change@debug{Preparing for switching to #1 (#2)\on@line}%
171 \expand@font@defaults

```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```

172 \let\target@series@value\@empty

```

As the very last item in the meta family list we add `\@elt{??}` and define this pseudo meta family to be the current font family. So if none of the real meta families matched then this will match. This will cover the following case:

- `\bfseries` is called for a family using `bx` (e.g., `CMR`)
- Switch to a font family that is none of the meta families, e.g., via `\fontfamily{ptm}\allowbreak\`

- Then none of the real meta families, match but the final `\@elt{??}` will.
- Therefore if the current series is `\mddefault` or `\bfdefault` it will be detected and the corresponding target series selected.

```
173 \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

To find it we loop over the meta family list with a suitable definition of `\@elt`.

```
174 \let\@elt\update@series@target@value
175 \@meta@family@list
```

Last resort pseudo meta family. Will only be looked at if none of the real ones have matched.

```
176 \@elt{??}%
177 \let\@elt\relax
```

That will figure out the correct series value to use without updating it. Now we can change the family.

```
178 \fontfamily#2%
```

After that we update the series. That code is again like the one above.

```
179 \ifx\target@series@value\@empty
180 <+debug> \series@change@debug{Target series still empty ...}%
181 \else
182 \ifx \f@series\target@series@value
183 <+debug> \series@change@debug{Target series unchanged:
184 <+debug> \f@series \space = \target@series@value}%
185 \else
186 \maybe@load@fontshape
187 <+debug> \series@change@debug{Target series:
188 <+debug> \f@series \space -> \target@series@value}%
```

The `\target@series@value` may contain something like `cm` (coming from a default) and so we can't directly assign it to `\f@series` we have to drop any surplus `m` first.

```
189 % \let\f@series\target@series@value
190 \series@maybe@drop@one@m\target@series@value\f@series
191 \fi
192 \fi
193 \fi
194 }
```

(End of definition for `\prepare@family@series@update`, `\@meta@family@list`, and `\@currentmetafamily`.)

`\update@series@target@value` In this macro you basically find the nested `\ifxs` from the outline above. The only difference is that it is parameterized instead of being written out. And we have only for one block of tests because the code is called repeatedly when looping over the meta family list. From the `\@meta@family@list` list we get each meta family name in turn.

```
195 \def\update@series@target@value#1{%
```

There is one additional test at the beginning, because the list contains all meta families and we need to ignore the case where current one from the list and target one are identical.

```
196 \def\reserved@a{#1}%
197 \ifx\@currentmetafamily\reserved@a % rm -> rm do nothing
198 \else
```



```

199 <+debug> \series@change@debug{Trying to match #1: \csname#1def@ult\endcsname
200 <+debug> \space = \f@family\space ?}%

```

We only “do” something if the current font family matches the current meta family.

```

201 \expandafter\ifx\csname#1def@ult\endcsname\f@family

```

If that’s the case we know that this is the block that applies (only one meta family can match). So to speed things up we change \@elt so that the rest of the loop gets gobbled.

```

202 \let\@elt\@gobble

```

Then we try to find the right new value for the series (as explained above). The two macros defined first are only there because we now need to use \csname and this way the code will be a little faster.

```

203 \expandafter\let\expandafter\reserved@b
204 \csname mdseries@\@currentmetafamily\endcsname
205 \expandafter\let\expandafter\reserved@c
206 \csname bfseries@\@currentmetafamily\endcsname
207 <+debug>\series@change@debug{Targets for mdseries and bfseries:
208 <+debug> \reserved@b\space and \reserved@c}%

```

This here is now identical to the nested \ifx block from the outline, except that it there appeared twice in \rmfamily. This is now covered by looping and stopping the loop when a match was found.

We have to sanitize the default value first because it may contain something like mc and that would never match \f@series because there it would be called c with the m dropped. It would be probably better to do that differently these days, but it is hard to adjust without causing a lot of issues, so we do the dropping in various places instead.

```

209 \expandafter\series@maybe@drop@one@m
210 \csname mdseries@#1\endcsname\reserved@d
211 \ifx\reserved@d\f@series
212 <+debug> \series@change@debug{mdseries@#1 matched -> \reserved@b}%
213 \let\target@series@value\reserved@b
214 \else

```

Again do some sanitizing.

```

215 \expandafter\series@maybe@drop@one@m
216 \csname bfseries@#1\endcsname\reserved@d
217 \ifx\reserved@d\f@series
218 <+debug> \series@change@debug{bfseries@#1 matched -> \reserved@c}%
219 \let\target@series@value\reserved@c
220 \else\ifx\f@series\mddef@ult \let\target@series@value\reserved@b
221 <+debug> \series@change@debug{mddef@ult matched -> \reserved@b}%
222 \else\ifx\f@series\bfdef@ult \let\target@series@value\reserved@c
223 <+debug> \series@change@debug{bfdef@ult matched -> \reserved@c}%
224 \fi\fi\fi\fi
225 \fi
226 \fi
227 }

```

(End of definition for \update@series@target@value.)

\@restoremetafamily In some situations it might be necessary to rerun the command that sets the current meta family, e.g., rerun \rmfamily if rm is the current meta family (for example, if the hook rmfamily contains some conditional code and the condition has changed at that point). The command \@restoremetafamily offers a simply way to do this.

```

228 \def\@restoremetafamily{%
229   \csname \@currentmetafamily family\endcsname
230 }
231
232 (End of definition for \@restoremetafamily.)
233
234 \ifx\@restoremetafamily\@empty
235 \else
236 \ifx\@restoremetafamily\@empty
237 \else
238 \ifx\@restoremetafamily\@empty
239 \else
240 \ifx\@restoremetafamily\@empty
241 \else
242 \ifx\@restoremetafamily\@empty
243 \else
244 \ifx\@restoremetafamily\@empty
245 \else
246 \ifx\@restoremetafamily\@empty
247 \else
248 \ifx\@restoremetafamily\@empty
249 \else
250 \ifx\@restoremetafamily\@empty
251 \else
252 \ifx\@restoremetafamily\@empty
253 \else
254 \ifx\@restoremetafamily\@empty
255 \else
256 \ifx\@restoremetafamily\@empty
257 \else
258 \ifx\@restoremetafamily\@empty
259 \else
260 \ifx\@restoremetafamily\@empty
261 \else
262 \ifx\@restoremetafamily\@empty
263 \else
264 \ifx\@restoremetafamily\@empty
265 \else
266 \ifx\@restoremetafamily\@empty
267 \else
268 \ifx\@restoremetafamily\@empty
269 \else
270 \ifx\@restoremetafamily\@empty
271 \else
272 \ifx\@restoremetafamily\@empty
273 \else
274 \ifx\@restoremetafamily\@empty
275 \else
276 \ifx\@restoremetafamily\@empty
277 \else
278 \ifx\@restoremetafamily\@empty
279 \else

```

```

280 <latexrelease> \else\ifx\f@series\bfdef@ult \let\target@series@value\reservedc
281 <latexrelease> \fi\fi\fi\fi
282 <latexrelease> \fi
283 <latexrelease> \fi
284 <latexrelease>}

285 <latexrelease>\EndIncludeInRelease
286 <latexrelease>\IncludeInRelease{0000/00/00}%
287 <latexrelease> {\prepare@family@series@update}{Custom series}%
288 <latexrelease>
289 <latexrelease>\let\@meta@family@list\@undefined
290 <latexrelease>\let\prepare@family@series@update\@undefined
291 <latexrelease>\let@update@series@target@value\@undefined
292 <latexrelease>
293 <latexrelease>\EndIncludeInRelease
294 <*2ekernel>

```

`\init@series@setup`

```

295 </2ekernel>
296 <*2ekernel | latexrelease>
297 <latexrelease>\IncludeInRelease{2020/02/02}%
298 <latexrelease> {\init@series@setup}{Custom series}%
    This is code to be run at begin document ...
299 \def\init@series@setup{%

```

We only want `bx` in `\bfseries@rm` if the roman font is Computer Modern or Latin Modern, otherwise it should be `b`. It was set to `bx` in the kernel so that any font use with the default families in the preamble get this value. Now at the real document start we check if the fonts have been changed. If there was a `\DeclareFontSeriesDefault` declaration or `\bfseries@rm` was directly altered then it differs from `\bfseries@rm@kernel` and we do nothing. Otherwise we check if `\rmdefault` is one of the CM/LM font families and if so we keep `bx` otherwise we change it to `b`.

This approach doesn't cover one case: CM/LM got changed to a different family that supports `bx`, but the support package for that family used `\def\bfseries@rm{bx}` instead of using `\DeclareFontSeriesDefault`. In that case the code here changes it to `b`. Solution: use the `\DeclareFontSeriesDefault` interface.

```

300 \ifx\bfseries@rm@kernel\bfseries@rm
301 \expandafter\in@\expandafter{\rmdefault}%
302 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
303 \ifin@ \else \def\bfseries@rm{b}\fi\fi

```

Same approach for `\bfseries@sf` and `\bfseries@tt`:

```

304 \ifx\bfseries@sf@kernel\bfseries@sf
305 \expandafter\in@\expandafter{\sfdefault}%
306 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
307 \ifin@ \else \def\bfseries@sf{b}\fi\fi
308 \ifx\bfseries@tt@kernel\bfseries@tt
309 \expandafter\in@\expandafter{\ttdefault}%
310 {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
311 \ifin@ \else \def\bfseries@tt{b}\fi\fi

```

If the document preamble has changed the `\familydefault` or if the if the `\rmdefault` contains a new font family, we may have to adjust the series defaults accordingly, before starting typesetting.

Similarly, if the user has changed the `\mddefault` or the medium series for the family selected as document font we may also have to adjust the `\seriesdefault`.

On the other hand if the document font is still CM or LM then `\bfdefault` is wrong, because it is now saying `b` and not `bx` as it should for such fonts.

To fix all this we first run `\reset@font` (the internal kernel name for `\normalfont`). This will set up the document encoding, family, series and shape based on the current values of `\encodingdefault`, `\familydefault`, `\seriesdefault` and `\shapedefault`. However, if the family (from `\familydefault`) has special medium default we should switch to that (and not use what is current value from `\seriesdefault`). This can be achieved by afterwards calling `\mediumseries` and then changing `\seriesdefault` to the now current series value (in `\f@series`).

But what should happen if `\seriesdefault` got explicitly changed? In that case the explicit change should survive and we should not alter `\seriesdefault`. This is solved by comparing the current value of `\seriesdefault` with a kernel version saved in the format and if they differ we do not call `\mdseries` or change `\seriesdefault`.

```

312 \reset@font
313 \ifx\seriesdefault\seriesdefault@kernel
314 \mdseries
315 \let\seriesdefault\f@series
316 \fi
317 }%
```

(End of definition for `\init@series@setup`.)

As the kernel code now implements the same functionality as `mweights`, albeit internally coded slightly differently, that package shouldn't be loaded any more. We therefore pretend that it already got loaded. Thus, a font package that tries to load it and then sets `\mdseries@.`, etc. will continue to work but will now use the kernel code.

Of course, mid-term such package should probably use `\DeclareFontSeriesDefault` instead of making using low-level definitions.

```

318 \expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
319 </2kernel | latexrelease>
320 <latexrelease>\EndIncludeInRelease
321 <latexrelease>\IncludeInRelease{0000/00/00}%
322 <latexrelease> \init@series@setup}{Custom series}%
```

This is always called in `\document` so don't make it undefined.

```

323 <latexrelease>\let\init@series@setup\relax
324 <latexrelease>
325 <latexrelease>\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
326 <latexrelease>
327 <latexrelease>\EndIncludeInRelease
328 <*2kernel>
329 </2kernel>
330 <*2kernel | latexrelease>
331 <latexrelease>\IncludeInRelease{2021/11/15}%
332 <latexrelease> \bfseries}{Custom series with hooks}%
```

`\bfseries` This document command switches to the bold series.

```

333 \DeclareRobustCommand\bfseries{%
334 \not@math@alphabet\bfseries\mathbf
```

In the original NFSS definition it then called `\fontseries` with the value `\bfdefault`. In the new scheme we have more alternatives and therefore check if the current family (`\f@family`) is the current `\rmdef@ult`, `\sfdef@ult` or `\ttdef@ult` and then select the correct family default in that case.

```

335 \expand@font@defaults
336 \maybe@update@bfseries@defaults

337 \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
338 \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
339 \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt

```

If not `\bfdefault` is used.

```

340 \else \fontseries\bfdefault
341 \fi\fi\fi

```

This hook in contrast is always executed.

```

342 \UseHook{bfseries}%
343 \selectfont
344 }

```

(End of definition for \bfseries.)

`\maybe@update@bfseries@defaults` If `\bfdefault` and `\bfdefault@previous` are different then the default got changed directly through the legacy interface (i.e., via `\def` or `\renewcommand`). In that case we reset all meta family defaults so that the document behaves like it was the case before the new mechanism was introduced.

```

345 \def\maybe@update@bfseries@defaults{%
346 \ifx\bfdefault\bfdefault@previous\else

```

We add `\@empty` and then let `\bfdefault@previous` to `\bfdefault` so that we can detect any further change.

```

347 \expandafter\def\expandafter\bfdefault
348 \expandafter{\bfdefault\@empty}%
349 \let\bfdefault@previous\bfdefault

```

And we reset the meta family defaults (`\bfdef@ult` is an expanded version of `\bfdefault`).

```

350 \let\bfseries@rm\bfdef@ult
351 \let\bfseries@sf\bfdef@ult
352 \let\bfseries@tt\bfdef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here. Not that this hook is only run when resets are necessary.

```

353 \UseHook{bfseries/defaults}%
354 \fi
355 }

```

(End of definition for \maybe@update@bfseries@defaults.)

`\mdseries` This document command switches to the medium series.

```

356 \DeclareRobustCommand\mdseries{%
357 \not@math@alphabet\mdseries\relax
358 \expand@font@defaults
359 \maybe@update@mdseries@defaults
360 \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
361 \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
362 \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt

```

```

363     \else                                \fontseries\mddefault
364     \fi\fi\fi
365     \UseHook{mdseries}%
366     \selectfont
367 }

```

(End of definition for \mdseries.)

\maybeupdate@mdseries@defaults

```

368 \def\maybeupdate@mdseries@defaults{%
369   \ifx\mddefault\mddefault@previous\else
370     \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
371     \let\mddefault@previous\mddefault
372     \let\mdseries@rm\mddef@ult
373     \let\mdseries@sf\mddef@ult
374     \let\mdseries@tt\mddef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here.

```

375     \UseHook{mdseries/defaults}%
376     \fi
377 }

```

(End of definition for \maybeupdate@mdseries@defaults.)

```

378 </2ekernel | latexrelease>
379 <latexrelease>\EndIncludeInRelease
380 <latexrelease>\IncludeInRelease{2020/10/01}%
381 <latexrelease>          {\bfseries}{Custom series with hooks}%
382 <latexrelease>
383 <latexrelease>\let\maybeupdate@bfseries@defaults\@undefined
384 <latexrelease>\let\maybeupdate@mdseries@defaults\@undefined
385 <latexrelease>
386 <latexrelease>\DeclareRobustCommand\bfseries{%
387 <latexrelease>  \not@math@alphabet\bfseries\mathbf
388 <latexrelease>  \expand@font@defaults
389 <latexrelease>  \ifx\bfdefault\bfdefault@previous\else
390 <latexrelease>    \expandafter\def\expandafter\bfdefault
391 <latexrelease>      \expandafter{\bfdefault\@empty}%
392 <latexrelease>    \let\bfdefault@previous\bfdefault
393 <latexrelease>    \let\bfseries@rm\bfdef@ult
394 <latexrelease>    \let\bfseries@sf\bfdef@ult
395 <latexrelease>    \let\bfseries@tt\bfdef@ult
396 <latexrelease>    \UseHook{bfseries/defaults}%
397 <latexrelease>  \fi
398 <latexrelease>    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
399 <latexrelease>    \else\ifx\f@family\sfdef@ult  \fontseries\bfseries@sf
400 <latexrelease>    \else\ifx\f@family\ttdef@ult  \fontseries\bfseries@tt
401 <latexrelease>    \else
402 <latexrelease>      \fi\fi\fi
403 <latexrelease>    \UseHook{bfseries}%
404 <latexrelease>    \selectfont
405 <latexrelease>}
406 <latexrelease>
407 <latexrelease>\DeclareRobustCommand\mdseries{%
408 <latexrelease>  \not@math@alphabet\mdseries\relax

```

```

409 <latexrelease> \expand@font@defaults
410 <latexrelease> \ifx\mddefault\mddefault@previous\else
411 <latexrelease> \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
412 <latexrelease> \let\mddefault@previous\mddefault
413 <latexrelease> \let\mdseries@rm\mddef@ult
414 <latexrelease> \let\mdseries@sf\mddef@ult
415 <latexrelease> \let\mdseries@tt\mddef@ult
416 <latexrelease> \UseHook{mdseries/defaults}%
417 <latexrelease> \fi
418 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
419 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
420 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
421 <latexrelease> \else \fontseries\mddefault
422 <latexrelease> \fi\fi\fi
423 <latexrelease> \UseHook{mdseries}%
424 <latexrelease> \selectfont
425 <latexrelease>}
426 <latexrelease>\EndIncludeInRelease
427 <latexrelease>\IncludeInRelease{2020/02/02}%
428 <latexrelease> {\bfseries}{Custom series with hooks}%
429 <latexrelease>
430 <latexrelease>
431 <latexrelease>\DeclareRobustCommand\bfseries{%
432 <latexrelease> \not@math@alphabet\bfseries\mathbf
433 <latexrelease> \expand@font@defaults
434 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
435 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
436 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
437 <latexrelease> \else \fontseries\bfdefault
438 <latexrelease> \fi\fi\fi
439 <latexrelease> \selectfont
440 <latexrelease>}
441 <latexrelease>
442 <latexrelease>\DeclareRobustCommand\mdseries{%
443 <latexrelease> \not@math@alphabet\mdseries\relax
444 <latexrelease> \expand@font@defaults
445 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
446 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
447 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
448 <latexrelease> \else \fontseries\mddefault
449 <latexrelease> \fi\fi\fi
450 <latexrelease> \selectfont
451 <latexrelease>}
452 <latexrelease>
453 <latexrelease>
454 <latexrelease>
455 <latexrelease>\EndIncludeInRelease
456 <latexrelease>\IncludeInRelease{0000/00/00}%
457 <latexrelease> {\bfseries}{Custom series with hooks}%
458 <latexrelease>
459 <latexrelease>\DeclareRobustCommand\bfseries
460 <latexrelease> {\not@math@alphabet\bfseries\mathbf
461 <latexrelease> \fontseries\bfdefault\selectfont}
462 <latexrelease>\DeclareRobustCommand\mdseries

```

```

463 <latexrelease>          {\not@math@alphabet\mdseries\relax
464 <latexrelease>          \fontseries\mddefault\selectfont}
465 <latexrelease>
466 <latexrelease>\EndIncludeInRelease
467 <*2ekernel>
468
469
470
471
472 </2ekernel>
473 <*2ekernel | latexrelease>
474 <latexrelease> \IncludeInRelease{2020/10/01}%
475 <latexrelease>          {\expand@font@defaults}{Custom series with hooks}%

```

`\expand@font@defaults` The family specific defaults are fully expanded, i.e., they are defined via `\edef` inside `\rm@def@ult` `\DeclareFontSeriesDefault`. However, the overall defaults, e.g., `\bfdefault` may have been redefined by the user and thus may not be fully expanded. So to enable reliable comparison we make expanded versions of them. That we rerun each time. The alternative would be to only allow for changes before begin document.

```

\sf@def@ult
\tt@def@ult
\md@def@ult
\bf@def@ult
476 \def\expand@font@defaults{%
477   \edef\rmdef@ult{\rmdefault}%
478   \edef\sdef@ult{\sfdefault}%
479   \edef\ttdef@ult{\ttdefault}%

```

The series defaults may contain some surplus `m` that we need to drop here.

```

480   \series@maybe@drop@one@m\bfdefault\bfdef@ult
481   \series@maybe@drop@one@m\mddefault\mddef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add additional code here.

```

482   \UseHook{expand@font@defaults}%
483 }

```

(End of definition for `\expand@font@defaults` and others.)

`\rmfamily` Here are the document level commands for changing the main font families, or rather, here is a documented outline of the code, the actual code is then streamlined and somewhat generalized.

```

DeclareRobustCommand\rmfamily{%
  \not@math@alphabet\rmfamily\mathrm

```

If families are changed then we have to do a bit more work. In the original NFSS implementation a family change kept encoding, series shape and size unchanged but now we can't any longer simply reuse the current series value. Instead we may have to change it from one family default to the next.

```

\expand@font@defaults

```

We have to do the testing while the current family is still unchanged but we have to do the adjustment of the series after it got changed (because the new family might have different sets of shapes available and we certainly don't want to see substitution going on. So we use `\target@series@value` to hold the target series (if any).

```

\let\target@series@value\@empty

```


Thus, if the current family is the sans family

```
\ifx\f@family\sfdef@ult
```

and if we using the medium series of the sans family

```
\ifx\f@series\mdseries@sf
```

then lets switch to the medium series for the serif family

```
\let\target@series@value\mdseries@rm
```

and if we use the bold series of the sans family switch to the bold default of the serif family:

```
\else\ifx\f@series\bfseries@sf \let\target@series@value\bfseries@rm
```

However, the sans family may not have any specific defaults set, so we also compare with the overall defaults.

```
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
```

If neither test was true we leave the series alone. This way a special manual setting such as `\fontseries{lc}` is not undone if the family changes (of course there may not be any support for it in the new family but then the NFSS substitution kicks in and sorts it out).

```
\fi\fi\fi\fi
```

We need to do the same if the current family is the typewriter family:

```
\else\ifx\f@family\ttdef@ult
\ifx\f@series\mdseries@tt \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfseries@tt \let\target@series@value\bfseries@rm
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
\fi\fi\fi\fi
\fi\fi
```

With these preparations for series out of the way we can now change the font family to `\rmdefault`.

```
\fontfamily\rmdefault
```

If `\target@series@value` is still empty there is nothing more to do other than selecting the new family. However, if not then we should update the font series now as well. But there is one further subtle issue. We may not have loaded an `.fd` file for our target font family yet. In the past that was done in `\selectfont` if necessary but since we are now doing all the comparisons in `\fontseries` we need to make sure that the font family specifications are already loaded prior to calling `\fontseries`.

```
\ifx\target@series@value\@empty \else
\maybe@load@fontshape
```

Updating the series in this case means directly changing `\f@series` to the target value. We don't want to go through `\fontseries` because that would apply the mappings and then `bx + b` would keep `bx` instead of changing to `b` as desired. as

```
\let\f@series\target@series@value
\fi
\selectfont}
```

So now for the real definition: most of the code above gets delegated to a helper command `\prepare@family@series@update` so that the definition becomes again fairly short. In addition we add a hook, mainly for our Japanese friends so that the code can be extended prior to the call to `\selectfont`.

```
484 \DeclareRobustCommand\rmfamily{%
485   \not@math@alphabet\rmfamily\mathrm
```

This holds all the code discussed above, first argument is the meta family, i.e., `rm` in this case, and second argument is the default family name, e.g., `cmr` indirectly accessed via `\rmdefault`. This is calling `\fontfamily` and if necessary `\fontseries` as outline above.

```
486   \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to `\selectfont`.

```
487   \UseHook{rmfamily}%
488   \selectfont}
```

The definitions for `\sffamily` and `\ttfamily` are similar, the differences are only in what font families get checked.

```
\sffamily
\ttfamily
489 \DeclareRobustCommand\sffamily{%
490   \not@math@alphabet\sffamily\mathsf
491   \prepare@family@series@update{sf}\sfdefault
492   \UseHook{sffamily}%
493   \selectfont}

494 \DeclareRobustCommand\ttfamily{%
495   \not@math@alphabet\ttfamily\mathtt
496   \prepare@family@series@update{tt}\ttdefault
497   \UseHook{ttfamily}%
498   \selectfont}
```

(End of definition for \rmfamily, \sffamily, and \ttfamily.)

```
rmfamily  Declare the hooks used above.
sffamily  499 \NewHook{rmfamily}
ttfamily  500 \NewHook{sffamily}
normalfont 501 \NewHook{ttfamily}
expand@font@defaults 502 \NewHook{normalfont}
bfseries  503 \NewHook{expand@font@defaults}
bfseries/defaults 504 \NewHook{bfseries}
mdseries  505 \NewHook{bfseries/defaults}
mdseries/defaults 506 \NewHook{mdseries}
507 \NewHook{mdseries/defaults}
```

(End of definition for rmfamily and others.)

```

\@rmfamilyhook These four hooks have legacy versions used in 2020/02/02 so we should support them
\@sffamilyhook until they aren't any longer used.
\@ttfamilyhook By default the hooks do nothing and in new code they should not any longer be
\@defaultfamilyhook used. Instead use the standard hook system and the hook names rmfamily, sffamily,
                    ttfamily. Instead of \@defaultfamilyhook use normalfont.

508 \let\@rmfamilyhook\@empty
509 \let\@sffamilyhook\@empty
510 \let\@ttfamilyhook\@empty
511 \let\@defaultfamilyhook\@empty

(End of definition for \@rmfamilyhook and others.)

512 </2ekernel | latexrelease>
513 <latexrelease>\EndIncludeInRelease
514 <latexrelease>\IncludeInRelease{2020/02/02}%
515 <latexrelease>                {\expand@font@defaults}{Custom series with hooks}%
516 <latexrelease>
517 <latexrelease>\def\expand@font@defaults{%
518 <latexrelease>  \edef\rmdef@ult{\rmdefault}%
519 <latexrelease>  \edef\sffdef@ult{\sffdefault}%
520 <latexrelease>  \edef\ttdef@ult{\ttdefault}%
521 <latexrelease>  \edef\bfdef@ult{\bfdefault}%
522 <latexrelease>  \edef\mddef@ult{\mddefault}%
523 <latexrelease>  \edef\famdef@ult{\familydefault}%
524 <latexrelease>}%
525 <latexrelease>
526 <latexrelease>
527 <latexrelease>\DeclareRobustCommand\rmfamily{%
528 <latexrelease>  \not@math@alphabet\rmfamily\mathrm
529 <latexrelease>  \prepare@family@series@update{rm}\rmdefault
530 <latexrelease>  \@rmfamilyhook
531 <latexrelease>  \selectfont}
532 <latexrelease>\DeclareRobustCommand\sffamily{%
533 <latexrelease>  \not@math@alphabet\sffamily\mathsf
534 <latexrelease>  \prepare@family@series@update{sf}\sffdefault
535 <latexrelease>  \@sffamilyhook
536 <latexrelease>  \selectfont}
537 <latexrelease>\DeclareRobustCommand\ttfamily{%
538 <latexrelease>  \not@math@alphabet\ttfamily\mathtt
539 <latexrelease>  \prepare@family@series@update{tt}\ttdefault
540 <latexrelease>  \@ttfamilyhook
541 <latexrelease>  \selectfont}
542 <latexrelease>\let\@rmfamilyhook\@empty
543 <latexrelease>\let\@sffamilyhook\@empty
544 <latexrelease>\let\@ttfamilyhook\@empty
545 <latexrelease>
546 <latexrelease>
547 <latexrelease>\EndIncludeInRelease
548 <latexrelease>\IncludeInRelease{0000/00/00}%
549 <latexrelease>                {\expand@font@defaults}{Custom series with hooks}%
550 <latexrelease>
551 <latexrelease>\let\expand@font@defaults\@undefined
552 <latexrelease>
553 <latexrelease>\DeclareRobustCommand\bfseries

```

```

554 <latexrelease>          {\not@math@alphabet\bfseries\mathbf}
555 <latexrelease>          \fontseries\bfdefault\selectfont}
556 <latexrelease>\DeclareRobustCommand\mdseries
557 <latexrelease>          {\not@math@alphabet\mdseries\relax
558 <latexrelease>          \fontseries\mddefault\selectfont}
559 <latexrelease>\DeclareRobustCommand\rmfamily
560 <latexrelease>          {\not@math@alphabet\rmfamily\mathrm}
561 <latexrelease>          \fontfamily\rmdefault\selectfont}
562 <latexrelease>\DeclareRobustCommand\sffamily
563 <latexrelease>          {\not@math@alphabet\sffamily\mathsf}
564 <latexrelease>          \fontfamily\sfdefault\selectfont}
565 <latexrelease>\DeclareRobustCommand\ttfamily
566 <latexrelease>          {\not@math@alphabet\ttfamily\mathtt}
567 <latexrelease>          \fontfamily\ttdefault\selectfont}
568 <latexrelease>
569 <latexrelease>\let\@rmfamilyhook\@undefined
570 <latexrelease>\let\@sffamilyhook\@undefined
571 <latexrelease>\let\@ttfamilyhook\@undefined
572 <latexrelease>
573 <latexrelease>\EndIncludeInRelease
574 <*2ekernel>

```

`\IfFontSeriesContextTF` With the ability for `\bfseries` or `\mdseries` to be mapped to different NFSS axis values it becomes important to have the ability to determine the current context as we can no longer look at `\fontseries` to answer a question such as “am I currently typesetting in a bold typeface?”

This is provided by the test `\IfFontSeriesContextTF`. It takes three arguments:

- The context we try to check (either `bf` for bold or `md` for medium, i.e., the same that can go into the first mandatory argument of `\DeclareFontSeriesDefault`),
- what to do if we are in this context (true case) and
- what to do if we are not (false case).

This allows you to define commands like `\IfBold`, e.g.,

```
\NewDocumentCommand\IfBold{mm}{\IfFontSeriesContextTF{bf}{#1}{#2}}
```

and then do

```
This is \IfBold{bold}{non-bold} text.
```

and get the appropriate result.

```

575 </2ekernel>
576 <*2ekernel | latexrelease>
577 <latexrelease>\IncludeInRelease{2020/10/01}%
578 <latexrelease>          {\IfFontSeriesContextTF}{Font series context}%
579 \DeclareRobustCommand\IfFontSeriesContextTF[1]{%
580   \expand@font@defaults

```

In the beginning we haven’t found the context we are looking for.

```
581   \@font@series@contextfalse
```

We store the requested context away for use in the tests.

```
582   \def\requested@test@context{#1}%

```

The next definition is there to ensure that get a final match during testing even if the current family is non of the meta families (`rm`, `sf` or `tt`). This will then basically tests if the current font family matches the overall default.

```
583 \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

Then we run through the meta family list (currently containing just the three values) followed by the artificial meta family `??` and test each of them in turn using `\test@font@series@context` as the testing command.

```
584 \let\@elt\test@font@series@context
585 \@meta@family@list
586 \@elt{??}%
587 \let\@elt\relax
```

Following that we evaluate the status of `\if@font@series@context` to determine which of the remaining arguments (true/false case) we have to execute.

```
588 \if@font@series@context
589 \expandafter\@firstoftwo
590 \else
591 \expandafter\@secondoftwo
592 \fi
593 }
```

The T/F variants were introduced in 2025 but to simplify rollback we pretend that this happened at the same time as `\IfFontSeriesContextTF`.

```
594 \long\def\IfFontSeriesContextT#1{\IfFontSeriesContextTF{#1}{}}
595 \def\IfFontSeriesContextF{\IfFontSeriesContextTF{}}
```

(End of definition for \IfFontSeriesContextTF.)

`\test@font@series@context` This tests the context (stored in `\requested@test@context`) and updates the boolean if the right context is found.

```
596 \def\test@font@series@context#1{%
```

First task is to figure out whether the current family matches `\rmfamily`, `\sffamily`, etc. so in `\reserved@a` we store the value of `\rmdef@ult` (or whatever the given meta family is) and compare that to `\f@family`.

```
597 \edef\reserved@a{\csname #1def@ult\endcsname}%
598 \ifx\f@family\reserved@a
```

If they match we have found the right meta family so we don't need to test any of the remaining meta family and therefore change `\@elt` to `\@gobble`.

```
599 \let\@elt\@gobble
```

Now we have to test if `\f@series` matches the requested context (e.g., whether `\bfseries@rm` has that value if the current meta family is `rm` and we are looking for the `bf` context).

```
600 \expandafter\ifx
601 \csname\requested@test@context series@#1\endcsname\f@series
```

If yes we change the boolean and are done.

```
602 \@font@series@contexttrue
```

If not then maybe the reason is that nothing special was set up for that meta family so we also check now check if `\f@series` matches the overall default (e.g., `\bfdef@ult` if we are looking for the bold context). If that matches we change the boolean.

```

603     \else
604         \expandafter\ifx
605             \csname\requested@test@context def@ult\endcsname\f@series
606             \@font@series@contexttrue
607     \fi\fi\fi
608 }

```

(End of definition for \test@font@series@context.)

`\if@font@series@context` The boolean to signal if we found the requested font series context.

```

609 \newif\if@font@series@context

```

(End of definition for \if@font@series@context.)

```

610 </2ekernel | latexrelease>
611 <latexrelease>\EndIncludeInRelease
612 <latexrelease>\IncludeInRelease{0000/00/00}%
613 <latexrelease>                {\IfFontSeriesContextTF}{Font series context}%
614 <latexrelease>
615 <latexrelease>\let\IfFontSeriesContextTF\@undefined
616 <latexrelease>\let\IfFontSeriesContextT\@undefined
617 <latexrelease>\let\IfFontSeriesContextF\@undefined
618 <latexrelease>\let\test@font@series@context\@undefined
619 <latexrelease>\let\if@font@series@context\@undefined
620 <latexrelease>\let\@font@series@contexttrue\@undefined
621 <latexrelease>\let\@font@series@contextfalse\@undefined
622 <latexrelease>\EndIncludeInRelease
623 <*2ekernel>

```

3 Supporting nested emphasis

By default L^AT_EX 2_ε supports two levels of nested emphasis: if the current font has an upright shape then it switches to `\itshape` otherwise to `\emminnershape` (which defaults to `\upshape`). This means nested emphasis will oscillate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthy sequence, but instead of providing a fixed one L^AT_EX now offers a general mechanism that allows to define arbitrary sequences.

`\DeclareEmphSequence`

`\emforce` This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of emphasis. The mechanism tries to be “smart” and verifies that the declarations actually alter the font. If not it will ignore this level and tries the next one—the assumption being that there was a manual font change in the document to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list actually change the font and not, say, just the color. In such a case one has to use `\emforce` to which directs the mechanism to use the level even if the font attributes haven’t changed.

`\emreset` If the nesting is so deep, that the specified levels are exhausted then `\emreset` is

used as a final set of declarations (which by default returns back to the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\strongemph`

In addition to support for `\emph` we now also offer `\strongemph` (by default boldening the text font. Like with `\emph` there are declarations that allow you adjust used fonts.

`\DeclareStrongEmphSequence`

`\strongemforce`

This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of “strong emphasis”. It works like in a same manner as `\DeclareEmphSequence` and holds the sequence of font changes for `\strongemph`. The command `\strongemforce` forces a given entry to be used even if it doesn’t seem to alter the font.

`\strongemreset`

If the nesting is so deep, that the specified levels are exhausted then `\strongemreset` is used as a final set of declarations (which by default returns back to bold series and the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\DeclareEmphSequence`

`\DeclareEmphSequence` expects a clist of declaration. Spaces in the argument are dropped to avoid spurious spaces in the output. The declarations are additive. At the very end the shape is reset using `\emreset` and `\emforce` so that this case is never skipped.³⁸ Further nested calls restart at the beginning.

```
624 </2ekernel>
625 <*2ekernel | latexrelease>
626 <latexrelease> \IncludeInRelease{2020/02/02}%
627 <latexrelease> { \DeclareEmphSequence}{Nested emph}%
628 \def\DeclareEmphSequence#1{%
629   \protected@edef\emfontdeclare@clist{\zap@space#1, \@empty\emforce\emreset}%
630 }
```

By default the it is empty, in which case `\eminnershape` is used by L^AT_EX.

```
631 \let\emfontdeclare@clist\@empty
```

(End of definition for \DeclareEmphSequence.)

`\emrest` Reset the font to upright and upper/lower case. With the default rules using `\shapedefault` does that for us but to be on the safe side we do it like this:

```
632 \DeclareRobustCommand\emreset{\upshape\ulcshape}
```

(End of definition for \emrest.)

`\em` The new definition for `\em` (and implicitly `\emph`) is the same as before as long as `\emfontdeclare@clist` is empty.

```
633 \DeclareRobustCommand\em{%
634   \@nomath\em
635   \ifx\emfontdeclare@clist\@empty
636     \ifdim \fontdimen\@ne\font >\z@
637       \eminnershape \else \itshape \fi
638   \else
```

³⁸Maybe we should not add `\emforce` but allow that case to be skipped as well. Of course, that might result in an endless loop if somebody defines a sequence without any font change and without `\emforce` but ...

But if not we use the list to decide how to do emphasis.

We use the current font to check if the declarations have any effect, so even a size change is allowed and identified as a modification (but a color change, for example, isn't). So first we save the current status.

```
639 \edef\em@currfont{\csname\curr@fontshape/\f@size\endcsname}%
```

Then we grab the next element from the list and check if it can be used.

```
640 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
641 \fi
642 }
643 \def\emminnershape{\upshape}
```

(End of definition for \em.)

\do@emfont@update We know that the list (if not empty) has at least 2 elements separated by a comma, so we pick up the first in #1 and the rest in #2.

```
644 \def\do@emfont@update#1,#2\do@emfont@update{%
```

First action is to alter the list and move the first entry to the end

```
645 \def\emfontdeclare@clist{#2,#1}%
```

Then we execute the current declaration. Appending `\selectfont` means one can write just `\fontshape{it}` and that works then too.

```
646 % \typeout{Use: \detokenize{#1}}%
647 #1\selectfont
```

We then compare the current font with our saved version, but with a slight twist: we add `\em@force` at the end of the name. Normally this is empty so has no effect but if there was an `\emforce` as part of #1 it will append a / to the font name (making it invalid) thus this will then always fail the test.

If the test fails we are done and the declarations will be used. Otherwise we will try the next declaration in the sequence.

```
648 \expandafter\ifx\csname \curr@fontshape/\f@size\em@force
```

For the comparison with `\ifx` we have to expand `\em@currfont` once as the relevant info is inside.

```
649 \expandafter\endcsname
650 \em@currfont
651 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
```

If `\emforce` was used, we have to undo its effect:

```
652 \else
653 \let\em@force\@empty
654 \fi
655 }
```

(End of definition for \do@emfont@update.)

\emforce The definition of `\emforce` is simple: change `\em@force` to make the above test always invalid.

```
656 \protected\def\emforce{\def\em@force{/}}
657 \let\em@force\@empty
658 </2ekernel | latexrelease>
659 <latexrelease>\EndIncludeInRelease
```


(End of definition for `\emforce` and `\em@force`.)

`\em` These are the older definitions for `\em`, prior to 2020.
`\eminnershape` We also have to define the *emphasize* font change command (i.e. `\em`). This command will look is the current font is sloped (i.e. has a positive `\fontdimen1`) and will then select either `\upshape` or `\itshape`.

```

660 <latexrelease>\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%
661 <latexrelease>\let\DeclareEmphSequence@undefined
662 <latexrelease>\let\emfontdeclare@clist@undefined
663 <latexrelease>\let\emreset@undefined
664 <latexrelease>\let\do@emfont@update@undefined
665 <latexrelease>\let\emforce@undefined
666 <latexrelease>\let\em@force@undefined
667 <latexrelease>
668 <latexrelease>\DeclareRobustCommand\em
669 <latexrelease>      {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
670 <latexrelease>      \eminnershape \else \itshape \fi}%
671 <latexrelease>\EndIncludeInRelease
672 <latexrelease>
673 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
674 <latexrelease>\DeclareRobustCommand\em
675 <latexrelease>      {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
676 <latexrelease>      \upshape \else \itshape \fi}%
677 <latexrelease>\let\eminnershape@undefined
678 <latexrelease>\EndIncludeInRelease
679 <*2ekernel>

```

(End of definition for `\em` and `\eminnershape`.)

4 Supporting `\strongemph`

```

680 \def\DeclareStrongEmphSequence#1{%
681   \protected@edef\strongfontdeclare@clist{\zap@space#1, \@empty\strongemforce\strongemreset}%
682 }
683 \let\strongfontdeclare@clist@empty
684 \DeclareRobustCommand\strongem{%
685   \@nomath\strongem
686   \edef\strong@currfont{\csname\curr@fontshape/\f@size\endcsname}%
687   \expandafter\do@strongfont@update\strongfontdeclare@clist\do@strongfont@update
688 }
689
690 \def\do@strongfont@update#1,#2\do@strongfont@update{%
691   \def\strongfontdeclare@clist{#2,#1}%
692   #1\selectfont
693   \expandafter\ifx\csname \curr@fontshape/\f@size\strongem@force
694   \expandafter\endcsname
695   \strong@currfont
696   \expandafter\do@strongfont@update\strongfontdeclare@clist\do@strongfont@update
697   \else
698     \let\strongem@force@empty
699   \fi
700 }
701

```

```

702 \protected\def\strongemforce{\def\strongem@force{}}
703 \let\strongem@force\empty
704

```

The command `\strongemph` is defined in `lftntcmd.dtx`, before then some of the necessary support commands aren't defined.

```

705 %\DeclareTextFontCommand{\strongemph}{\strongem}

```

Initialization needs delaying too as some of the necessary macros are defined late. This is done in `lftfinal.dtx`.

```

706 %\DeclareRobustCommand\strongemreset{\bfseries\upshape\ulcshape}
707 %\DeclareStrongEmphSequence{\bfseries,\itshape}

```

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newfont.sty`.

```

708 \def\not@math@alphabet#1#2{%
709   \relax
710   \ifmmode
711     \@latex@error{Command \noexpand#1invalid in math mode}%
712     {%
713       Please
714       \ifx#2\relax
715         define a new math alphabet^^J%
716         if you want to use a special font in math mode%
717       \else

```

We have to a `\noexpand` below to prevent expansion of `#2`. In case of `#1` we can omit this (due to the current definition of robust commands since they do come out right there :-).

```

718         use the math alphabet \noexpand#2instead of
719         the #1command%
720       \fi
721       .
722     }%
723   \fi}

```

(End of definition for \not@math@alphabet.)

Finally we provide two abbreviations to switch to the $\text{\LaTeX}$ versions.

```

724 \DeclareRobustCommand\boldmath{\@nomath\boldmath
725   \mathversion{bold}}
726 \DeclareRobustCommand\unboldmath{\@nomath\unboldmath
727   \mathversion{normal}}

```

Here we switch to the default math version by defining the internal macro `\math@version`. We dare not to call `\mathversion` at this place because this would call `\glb@settings`.

```

728 \def\math@version{normal}

```

4.1 Legacy

We start by defining a few macros that are part of standard $\text{\LaTeX}$'s user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

`\newfont`

```

729 \def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}

```

(End of definition for \newfont.)

`\symbol`

```
730 </2ekernel>
731 <*2ekernel | latexrelease>
732 <latexrelease>\IncludeInRelease{2020/10/01}%
733 <latexrelease>{\symbol}{XeTeX change for math}%
734 \ifdefined\XeTeXversion
735 \DeclareRobustCommand\symbol[1]{\Ucharcat#1 12\relax}
736 \else
737 \DeclareRobustCommand\symbol[1]{\char#1\relax}
738 \fi
739 </2ekernel | latexrelease>
740 <latexrelease>\EndIncludeInRelease
741 <latexrelease>\IncludeInRelease{0000/00/00}%
742 <latexrelease>{\symbol}{XeTeX change for math}%
743 <latexrelease>
744 <latexrelease>\DeclareRobustCommand\symbol[1]{\char#1\relax}
745 <latexrelease>
746 <latexrelease>\EndIncludeInRelease
747 <*2ekernel>
```

(End of definition for \symbol.)

4.2 Miscellaneous

`\@setfontsize` This abbreviation is used by L^AT_EX's user level size changing commands, such as `\large`.

`\@setsize` 748 `\def\@setfontsize#1#2#3{\@nomath#1%`

For the benefit of people relying on keeping the name of the current font command saved in `\@currsize` we define it. To ensure that `\@setfontsize` keeps being robust we omit this assignment during times where `\protect` differs from `\@typeset@protect`.

```
749 \ifx\protect\@typeset@protect
750 \let\@currsize#1%
751 \fi
752 \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define `\@setsize` the 209 command

```
753 <*compat>
754 \def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
755 </compat>
```

(End of definition for \@setfontsize and \@setsize.)

`\hexnumber@` To set up L^AT_EX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple `\ifcase`.

```
756 \def\hexnumber@#1{\ifcase\number#1
757 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
758 9\or A\or B\or C\or D\or E\or F\fi}
```

(End of definition for \hexnumber@.)

`\nfss@text` In its simplest form `\nfss@text` is an `\mbox`. This will produce unbreakable text outside math and inside math you will get text with the same fonts as outside. The only drawback is that such item won't change sizes in subscripts. But this behavior can be easily changed.

With the `amstex` style option one will get a sub style called `amstext` which will redefine the `\nfss@text` macro to produce correct text in all sizes.

We have to use `\def` instead of the shorter `\let` since `\mbox` is undefined when we reach this point.

```
759 \def\nfss@text#1{\mbox{#1}}
```

(End of definition for `\nfss@text`.)

\copyright The definition of `\copyright` was changed so that it works in other type styles, and to make it robust. We leave the family untouched so that the copyright notice will come out differently if a different font family is in use. This command is commented out, since it is now defined in `ltoutenc.dtx`.

```
760 %\DeclareRobustCommand\copyright
761 %    {\ooalign{\hfil
762 %        \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
763 %        \mathhexbox20D}}}
```

(End of definition for `\copyright`.)

\normalfont The macro `\reset@font` is used in L^AT_EX to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new L^AT_EX version above but nevertheless are able to run all kind of mixtures.

The user interface name for `\reset@font` is `\normalfont`:

```
764 </2ekernel>
765 <*2ekernel|latexrelease>
766 <latexrelease>\IncludeInRelease{2025/11/01}%
767 <latexrelease>        {\normalfont}{Support meta family}%
768 \DeclareRobustCommand\normalfont{%
```

Instead of calling `\usefont`, as it was done in the past, we inline the code from `\usefont` as we want to add the hook before `\selectfont`, but after all the font attributes are set.

```
769 \fontencoding\encodingdefault
770 \edef\f@family{\familydefault}%
771 \edef\f@series{\seriesdefault}%
772 \edef\f@shape{\shapedefault}%
```

Now that `\f@family` is set we can update the meta family macro.

```
773 \set@current@meta@family
```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
774 \let\delayed@f@adjustment\@empty
775 \UseHook{normalfont}%
776 \selectfont}
777 \let\reset@font\normalfont
```

(End of definition for `\normalfont` and `\reset@font`.)

\set@current@meta@family This macros determines the current meta family by comparing the current family in `\f@family` with the various document font defaults, e.g., `\rmdefault`.

```
778 \def\set@current@meta@family {%
```

Because these defaults can be set by the user and may contain the family name indirectly, we first expand them to ensure we do not get false negatives.

```

779 \edef\rmdef@ult{\rmdefault}%
780 \ifx\f@family\rmdef@ult
781 \def\@currentmetafamily{rm}%
782 \else
783 \ifx\f@family\sfdef@ult
784 \def\@currentmetafamily{sf}%
785 \else
786 \ifx\f@family\ttdef@ult
787 \def\@currentmetafamily{tt}%
788 \else
789 \def\@currentmetafamily{??}%
790 \fi
791 \fi
792 \fi
793 }

```

(End of definition for \set@current@meta@family.)

```

794 % \changes{v3.2g}{2021/03/18}
795 % {Add missing 2020/02/02 latexrelease entry.}
796 </2ekernel | latexrelease>
797 <latexrelease>\EndIncludeInRelease
798 <latexrelease>\IncludeInRelease{2021/06/01}%
799 <latexrelease> \{\normalfont\}{Cancel delayed actions}%
800 <latexrelease>
801 <latexrelease>\DeclareRobustCommand\normalfont{%
802 <latexrelease> \fontencoding\encodingdefault
803 <latexrelease> \edef\f@family{\familydefault}%
804 <latexrelease> \edef\f@series{\seriesdefault}%
805 <latexrelease> \edef\f@shape{\shapedefault}%
806 <latexrelease> \let\delayed@f@adjustment\@empty
807 <latexrelease> \UseHook{normalfont}%
808 <latexrelease> \@defaultfamilyhook % hookname from 2020/02 will vanish
809 <latexrelease> \selectfont}
810 <latexrelease>\let\reset@font\normalfont
811 <latexrelease>
812 <latexrelease>\let\set@current@meta@family\undefined
813 <latexrelease>\EndIncludeInRelease

814 <latexrelease>\IncludeInRelease{2020/10/01}%
815 <latexrelease> \{\normalfont\}{Add hook to \normalfont}%
816 <latexrelease>
817 <latexrelease>\DeclareRobustCommand\normalfont{%
818 <latexrelease> \fontencoding\encodingdefault
819 <latexrelease> \edef\f@family{\familydefault}%
820 <latexrelease> \edef\f@series{\seriesdefault}%
821 <latexrelease> \edef\f@shape{\shapedefault}%
822 <latexrelease> \UseHook{normalfont}%
823 <latexrelease> \@defaultfamilyhook % hookname from 2020/02 will vanish
824 <latexrelease> \selectfont}
825 <latexrelease>
826 <latexrelease>\let\reset@font\normalfont
827 <latexrelease>

```

```

828 <latexrelease>\EndIncludeInRelease
829 <latexrelease>
830 <latexrelease>\IncludeInRelease{2020/02/02}%
831 <latexrelease>          {\normalfont}{Add hook to \normalfont}%
832 <latexrelease>
833 <latexrelease>\DeclareRobustCommand\normalfont{%
834 <latexrelease>  \fontencoding\encodingdefault
835 <latexrelease>  \edef\f@family{\familydefault}%
836 <latexrelease>  \edef\f@series{\seriesdefault}%
837 <latexrelease>  \edef\f@shape{\shapedefault}%
838 <latexrelease>  \@defaultfamilyhook
839 <latexrelease>  \selectfont}
840 <latexrelease>
841 <latexrelease>\let\reset@font\normalfont
842 <latexrelease>
843 <latexrelease>\let\@defaultfamilyhook\@empty
844 <latexrelease>
845 <latexrelease>\EndIncludeInRelease
846 <latexrelease>
847 <latexrelease>\IncludeInRelease{0000/00/00}%
848 <latexrelease>          {\normalfont}{Add hook to \normalfont}%
849 <latexrelease>
850 <latexrelease>\DeclareRobustCommand\normalfont
851 <latexrelease>          {\usefont\encodingdefault
852 <latexrelease>          \familydefault
853 <latexrelease>          \seriesdefault
854 <latexrelease>          \shapedefault
855 <latexrelease>          \relax}
856 <latexrelease>\let\reset@font\normalfont
857 <latexrelease>
858 <latexrelease>\let\@defaultfamilyhook\@undefined
859 <latexrelease>
860 <latexrelease>\EndIncludeInRelease
861 <*2ekernel>

```

`\fontfamily` One place where the current meta family should be set is `\fontfamily`, which is why we defined it here after the definition of `\set@current@meta@family` in this file.

```

862 </2ekernel>
863 <*2ekernel | latexrelease>
864 <latexrelease>\IncludeInRelease{2025/11/01}%
865 <latexrelease>          {\fontfamily}{Set meta family}%
866 <latexrelease>\DeclareRobustCommand\fontfamily[1]
867 <latexrelease>  {\edef\f@family{#1}\set@current@meta@family}
868 </2ekernel | latexrelease>
869 <latexrelease>\EndIncludeInRelease
870 <latexrelease>\IncludeInRelease{0000/00/00}%
871 <latexrelease>          {\fontfamily}{Set meta family}%
872 <latexrelease>
873 <latexrelease>\DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
874 <latexrelease>\EndIncludeInRelease
875 <*2ekernel>

```

(End of definition for \fontfamily.)

We left out the special L^AT_EX fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```

876 \def\not@base#1{\@latex@error
877   {Command \noexpand#1not provided in base LaTeX2e}%
878   {Load the latexsym or the amsfonts package to
879    define this symbol}}
880 \def\mho{\not@base\mho}
881 \def\Join{\not@base\Join}
882 \def\Box{\not@base\Box}
883 \def\Diamond{\not@base\Diamond}
884 \def\leadsto{\not@base\leadsto}
885 \def\squsubset{\not@base\squsubset}
886 \def\sqsupset{\not@base\sqsupset}
887 \def\lhd{\not@base\lhd}
888 \def\unlhd{\not@base\unlhd}
889 \def\rhd{\not@base\rhd}
890 \def\unrhd{\not@base\unrhd}

```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```

891 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %% don't modify this setting
892                                         %% overwrite it in fontdef.cfg
893                                         %% if necessary

```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```

894 \def\f@family{cmr}          % can't use \fontfamily at this point as
895                             % defaults aren't set up yet

```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```

896 \def\f@series{m}           % \fontseries{m}
897 \def\f@shape{n}           % \fontshape{n}
898 \fontsize{10}{10}

```

The initial `fontenc` package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```

899 \def\@fontenc@load@list{\@elt{T1,OT1}}

```

We now load the customizable parts of NFSS.

```

900 \InputIfFileExists{fonttext.cfg}
901     {\typeout{=====^^J%
902              ^^J%
903              Local config file fonttext.cfg used^^J%
904              ^^J%
905              =====}%}

```

```

906         \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
907     }
908     {\input{fonttext.ltx}}
909 \let\@addtofilelist\@gobble

Ditto for math although I don't think that we will get a lot of customization :-)
910 \InputIfFileExists{fontmath.cfg}
911     {\typeout{=====^^J%
912             ^^J%
913             Local config file fontmath.cfg used^^J%
914             ^^J%
915             =====}%
916     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
917     }
918     {\input{fontmath.ltx}}
919 \let\@addtofilelist\@gobble

```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```

920 \InputIfFileExists{preload.cfg}
921     {\typeout{=====^^J%
922             ^^J%
923             Local config file preload.cfg used^^J%
924             ^^J%
925             =====}%
926     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
927     }
928     {\input{preload.ltx}}
929 \let\@addtofilelist\@gobble

```

`\seriesdefault` After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we alter its value by appending `\@empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```

930 \expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\@empty}
931 \let\seriesdefault@kernel\seriesdefault

```

(End of definition for `\seriesdefault` and `\seriesdefault@kernel`.)

`\@acci` We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can be restored by a `minipage` inside a `tabbing` environment.

```

\@accii 932 \let\@acci\ ' \let\@accii\ ' \let\@acciii\=
\@acciii

```

(End of definition for `\@acci`, `\@accii`, and `\@acciii`.)

`\cal` Here were the two old `\alphabet identifiers`.

```

\mit
(End of definition for \cal and \mit.)

```

```

933 \endkernel

```


File 30

fontdef.dtx

1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If $\text{\LaTeX}$ 2_ε finds a file of this name it will use it, otherwise it uses the standard file which is `fontdef.ltx`.

If you don't plan to use Computer Modern much or at all, it might (!) be a good idea to make your own `fonttext.cfg`. Look at the comments below (docstrip module 'text') to see what should go into such a file.

To change the math font setup use a copy of `fontmath.ltx` under the name `fontmath.cfg` and change this copy. However, dealing with this interface is even more a job for an expert than changing the text font setup — in short, we don't encourage either.

Warning: please note that we don't support customised $\text{\LaTeX}$ versions. Thus, before sending in a bug report please try your test file with a $\text{\LaTeX}$ format which is not customised and send in the log from that version (unless the problem goes away).

Please note: the following standard encodings have to be defined in all local variants of `font....cfg` to guarantee that all $\text{\LaTeX}$ installations behave in the same way.

T1	Cork $\text{\TeX}$ text encoding
OT1	old $\text{\TeX}$ text encoding
U	unknown encoding
OML	old $\text{\TeX}$ math letters encoding
OMS	old $\text{\TeX}$ math symbols encoding
OMX	old $\text{\TeX}$ math extension symbols encoding
TU	Unicode

Notice that some of these encodings are ‘old’ in the sense that we hope that they will be superseded soon by encoding standards defined by the T_EX user community. Therefore this set of default encodings may change in the future.

The first candidate is OT1 which will soon be replaced by T1, the official T_EX text encoding.

Warning: If you add additional encodings to this file there is no guarantee any longer that files processable at your installation will also be processable at other installations. Thus, if you make use of such an encoding in your document, e.g. if you intend to typeset in Cyrillic (OT2 encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all L^AT_EX installations (provided they have suitable fonts installed).

For this reason we suggest that you define a short package file that sets up an additional encoding used at your site (rather than putting the encoding into this file) since this package can easily be shipped with your document.

3 The docstrip modules

The following modules are used to direct **docstrip** in generating external files:

driver	produce a documentation driver file
text	produce the file fonttext.ltx
math	produce the file fontmath.ltx
cfgtext	produce a dummy fonttext.cfg file
cfgmath	produce a dummy fontmath.cfg file

A typical **docstrip** command file would then have entries like:

```
generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

4 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \GetFileInfo{fontdef.dtx}
4 \begin{document}
5   \DocInput{fontdef.dtx}
6 \end{document}
7 </driver>
```

5 The fonttext.ltx file

The identification is done earlier on with a \ProvidesFile declaration.

```
8 <*text>
9 \typeout{=== Don't modify this file, use a .cfg file instead ===^~J}
```

5.1 Encodings

This file declares the standard encodings for text and math fonts. All others should be declared in packages or in the documents directly.

For every text encoding there are normally a number of encoding specific commands, e.g. accents, special characters, etc. (The definition for such a command might have to change when the encoding is changed, because the character is in a different position, or not available at all, or the accent is produced in a different way.) This is handled by a general mechanism which is described in `ltoutenc.dtx`.

By convention, text encoding specific declarations, including the `\DeclareFontEncoding` declaration, are kept in separate file of the form `<enc>enc.def`, e.g. `ot1enc.def`. This allows other applications to make use of the declarations as well.

Similar to the default encoding, the loading of the encoding files for the two major text encodings shouldn't be changed. In particular, the `inputenc` package depends on this.

```
10 \input {omlenc.def}
11 \input {omsenc.def}
```

Documents containing a lot of accented characters should really be using T1 fonts. We therefore load this after OT1 so that T1 encoding specific commands are executed as fast as possible (encoding files are no longer reloaded in `fontenc`).

```
12 \input {ot1enc.def}
13 \input {t1enc.def}
14 \input {ts1enc.def}

15 \ifx\Umathcode\@undefined
```

We then set the default text font encoding. This will hopefully change some day to T1. This setting should *not* be changed to produce a portable format.

```
16 \fontencoding{OT1}

The initial fontenc package load list if an 8-bit TEX engine is used:
17 \def\@fontenc@load@list{\@elt{T1,OT1}}
18 \def\rmsubstdefault{cmr}
19 \def\sfsubstdefault{cmss}
20 \def\ttsubstdefault{cmtt}
21 \LoadFontDefinitionFile{TS1}{cmr}

22 \else
```

Unicode.

```
23 \input {tuenc.def}
24 \fontencoding{TU}

The initial fontenc package load list if a Unicode engine is used:
25 \def\@fontenc@load@list{\@elt{TU}}
26 \DeclareFontSubstitution{TU}{lmr}{m}{n}
27 \LoadFontDefinitionFile{TU}{lmr}
28 \LoadFontDefinitionFile{TU}{lmss}
29 \LoadFontDefinitionFile{TU}{lmtt}

30 \def\rmsubstdefault{lmr}
31 \def\sfsubstdefault{lmss}
32 \def\ttsubstdefault{lmtt}
33 \LoadFontDefinitionFile{TS1}{lmr}
```

```
34 \DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
35 \fi
```

If different encodings for text fonts are in use one could put the common setup into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
36 \DeclareFontEncodingDefaults{}{}
```

The default font substitution for an encoding is defined in the corresponding `...enc.def` file so for OT1, T1, and TS1 this is already defined.

```
37 %\DeclareFontSubstitution{T1}{cmr}{m}{n}
```

```
38 %\DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

```
39 %\DeclareFontSubstitution{TS1}{cmr}{m}{n}
```

This release of L^AT_EX 2_ε assumes that the ec fonts are available. It is possible to change this to point to some other font family (e.g., Times with the appropriate encoding if it is available) without making documents non-portable. However, in such a case documents will produce different page breaks at other sites. The substitution defaults can all be changed without losing portability as long as there are font shape definitions for the selected substitutions.

For every encoding declaration, L^AT_EX 2_ε will try to verify that the given substitution information makes sense, i.e. that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. L^AT_EX 2_ε will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will check if there is a `\DeclareFontShape` declaration for this combination. We will therefore load the corresponding `.fd` files now. If we don't do this they would be loaded at verification time (i.e. at `\begin{document}`) which would delay processing unnecessarily.

Warning: Please note that this means that you have to regenerate the format whenever you change any of these `.fd` files since L^AT_EX 2_ε will not read `.fd` files if it already knows about the encoding/family combination.

The `\nfss@catcodes` ensures that white space is ignored in any definitions made in the fd files.

```
40 \begingroup
```

```
41 \nfss@catcodes
```

```
42 \input {t1cmr.fd}
```

```
43 \input {ot1cmr.fd}
```

```
44 \input {ts1cmr.fd}
```

```
45 \endgroup
```

We also load some other font definition files which are normally needed in a document. This is only done for processing speed and you can comment the next two lines out to save some memory. If necessary these files are then loaded when your document is processed. (Loading `.fd` files is a less drastic step compared to preloading fonts because the number of fonts is limited 255 at (nearly) every T_EX installation, while the amount of main memory is not a limiting factor at most installations.)

```
46 \begingroup
```

```
47 \nfss@catcodes
```

```
48 \input {t1cmss.fd}
```

```

49 \input {t1cmtt.fd}
50 \input {ot1cmss.fd}
51 \input {ot1cmtt.fd}

52 \input {ts1cmss.fd}
53 \input {ts1cmtt.fd}
54 \endgroup

```

Even though Unicode engines default to `lm` load `ts1cmr` as this may be used for fallback for TS1 encoding.

We now load it in all engines above, so the next lines are no longer necessary. We keep them here if we stop loading other `fd` files in Unicode engines.

```

55 %\ifx\Umathcode\@undefined\else
56 %\begingroup
57 %\nfss@catcodes
58 %\input {ts1cmr.fd}
59 %\endgroup
60 %\fi

```

Even with all the precautions it is still possible that NFSS will run into problems, for example, when a `.fd` file contains corrupted data. To guard against such cases NFSS has a very low-level fallback font that is installed with the following line.

```

61 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}

```

This means, “if everything else fails use Computer Modern Roman normal shape at 10pt in the old text encoding”. You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

5.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if non-standard families are used we provide nine macros which hold the name of the corresponding families, series, and so on. This makes it easy to use other font families (like Times Roman, etc.). One simply has to redefine these defaults.

All these hooks have to be defined in this file but you can change their meaning (except for `\encodingdefault`) without making documents non-portable.

```

\encodingdefault The following three definitions set up the meaning for \rmfamily, \sffamily, and
\rmdefault      \ttfamily.
\sfdefault
\ttdefault
62 \ifx\Umathcode\@undefined
63 \newcommand\encodingdefault{OT1}
64 \newcommand\rmdefault{cmr}
65 \newcommand\sfdefault{cmss}
66 \newcommand\ttdefault{cmtt}
67 \else
68 \newcommand\encodingdefault{TU}
69 \newcommand\rmdefault{lmr}
70 \fontfamily{\rmdefault}
71 \newcommand\sfdefault{lmss}
72 \newcommand\ttdefault{lmtt}
73 \fi
74 \</text>
75 \<latexrelease>\IncludeInRelease{2017/01/01}%
76 \<latexrelease>          {\encodingdefault}{TU encoding default}%

```

```

77 <latexrelease>\ifx\Umathcode\@undefined
78 <latexrelease>\renewcommand\encodingdefault{OT1}
79 <latexrelease>\fontencoding{\encodingdefault}
80 <latexrelease>\renewcommand\rmdefault{cmr}
81 <latexrelease>\fontfamily{\rmdefault}
82 <latexrelease>\renewcommand\sfddefault{cmss}
83 <latexrelease>\renewcommand\ttdefault{cmtt}
84 <latexrelease>\else
85 <latexrelease>\renewcommand\encodingdefault{TU}
86 <latexrelease>%done in everyjob\fontencoding{\encodingdefault}
87 <latexrelease>\renewcommand\rmdefault{lmr}
88 <latexrelease>\fontfamily{\rmdefault}
89 <latexrelease>\renewcommand\sfddefault{lmss}
90 <latexrelease>\renewcommand\ttdefault{lmtt}
91 <latexrelease>\fi
92 <latexrelease>\EndIncludeInRelease
93 <latexrelease>\IncludeInRelease{0000/00/00}%
94 <latexrelease>          {\encodingdefault}{TU encoding default}%
95 <latexrelease>\fontencoding{OT1}
96 <latexrelease>\renewcommand\encodingdefault{OT1}
97 <latexrelease>\fontencoding{\encodingdefault}
98 <latexrelease>\renewcommand\rmdefault{cmr}
99 <latexrelease>\fontfamily{\rmdefault}
100 <latexrelease>\renewcommand\sfddefault{cmss}
101 <latexrelease>\renewcommand\ttdefault{cmtt}
102 <latexrelease>\EndIncludeInRelease
103 <*text>

```

(End of definition for \encodingdefault and others.)

\bfdefault Series changing commands are influenced by the following hooks.
\mddefault

```

104 \newcommand\bfdefault{b} % overwritten below (for rollback)
105 \newcommand\mddefault{m} % overwritten below (for rollback)

```

(End of definition for \bfdefault and \mddefault.)

\itdefault Shape changing commands use the following hooks.
\sldefault

```

106 \newcommand\itdefault{it}
107 \newcommand\sldefault{sl}
108 \newcommand\scdefault{sc}
109 \newcommand\updefault{up} % overwritten below (for rollback)

```

(End of definition for \itdefault and others.)

```

110 </text>
111 <*text | latexrelease>
112 <latexrelease>\IncludeInRelease{2020/02/02}%
113 <latexrelease>          {\updefault}{font defaults change}%
114 \renewcommand\updefault{up}

```

We append \@empty to the series value so that we can detect if it got changed via \def or \renewcommand later.

```

115 \renewcommand\bfdefault{b\@empty}
116 \renewcommand\mddefault{m\@empty}

```

```

117 \let\bfdefault@previous\bfdefault
118 \let\mddefault@previous\mddefault
119 </text | latexrelease>
120 <latexrelease>\EndIncludeInRelease
121 <latexrelease>\IncludeInRelease{0000/00/00}%
122 <latexrelease>                {\updefault}{font defaults change}%
123 <latexrelease>
124 <latexrelease>\renewcommand\updefault{n}
125 <latexrelease>\renewcommand\bfdefault{bx}
126 <latexrelease>
127 <latexrelease>\let\bfdefault@previous\undefined
128 <latexrelease>\let\mddefault@previous\undefined
129 <latexrelease>\EndIncludeInRelease
130 <*text>

```

`\familydefault` Finally we have the hooks that describe the behaviour of the `\normalfont` command.
`\seriesdefault` To stay portable, the definition of `\encodingdefault` should *not* be changed and should
`\shapedefault` match the setting above for `\fontencoding`. All other values can be set according to your taste.

```

131 \newcommand\familydefault{\rmdefault}
132 \newcommand\seriesdefault{\mddefault}

```

In previous releases `\shapedefault` pointed to `\updefault` which resolved to `n`, but these days that is no longer the case (and `up` is wrong when you want to do a reset. So we now use `n` explicitly.

```

133 \newcommand\shapedefault{n}

```

(End of definition for `\familydefault`, `\seriesdefault`, and `\shapedefault`.)

This finishes the low-level setup in `fonttext.ltx`.

```

134 </text>

```

6 The fontmath.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```

135 <*math>
136 \typeout{=== Don't modify this file, use a .cfg file instead ===^J}

```

6.1 The font encodings used

```

137 \DeclareFontEncoding{OML}{-}{-}
138 \DeclareFontEncoding{OMS}{-}{-}
139 \DeclareFontEncoding{OMX}{-}{-}

```

Finally a declaration for `U` encoding which serves for all fonts that do not fit standard encodings. For math this sets up `\noaccents@` providing for AMS- $\text{\LaTeX}$. This macro is used therein to handle accented characters if they are not supported by the font. In other words, if fonts with `U` encoding are used in math, all accents (like from `\breve`) are obtained from some other font that has them.

```

140 \DeclareFontEncoding{U}{-}{\noaccents@}

```

The encodings for math are next:

```

141 \DeclareFontSubstitution{OML}{cmm}{m}{it}
142 \DeclareFontSubstitution{OMS}{cmsy}{m}{n}

```

```

143 \DeclareFontSubstitution{OMX}{cmex}{m}{n}
144 \DeclareFontSubstitution{U}{cmr}{m}{n}
145 \begingroup
146 \nfss@catcodes
147 \input {omlcmm.fd}
148 \input {omscmsy.fd}
149 \input {omxcmex.fd}
150 \input {ucmr.fd}
151 \endgroup

```

6.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by L^AT_EX. These four symbol fonts must be defined by this file.

It is possible to make the symbol fonts point to other external fonts without losing the ability to process documents written at other sites, as long as one defines the same symbol font names with the same encodings, e.g. `operators` with `OT1` etc. If other encodings are used documents become non-portable. Such a change should therefore be done in a package file.

```

152 \DeclareSymbolFont{operators} {OT1}{cmr}{m}{n}
153 \DeclareSymbolFont{letters} {OML}{cmm}{m}{it}
154 \DeclareSymbolFont{symbols} {OMS}{cmsy}{m}{n}
155 \DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
156 \SetSymbolFont{operators}{bold}{OT1}{cmr}{bx}{n}
157 \SetSymbolFont{letters}{bold}{OML}{cmm}{b}{it}
158 \SetSymbolFont{symbols}{bold}{OMS}{cmsy}{b}{n}

```

Below are the seven math alphabets which are defined by NFSS. Again they must be defined by this file. However, as before you can change the fonts used without losing portability, but you should be careful when changing the encoding since that may make documents come out wrong.

```

159 \DeclareSymbolFontAlphabet{\mathrm} {operators}
160 \DeclareSymbolFontAlphabet{\mathnormal}{letters}
161 \DeclareSymbolFontAlphabet{\mathcal} {symbols}
162 \DeclareMathAlphabet {\mathbf} {OT1}{cmr}{bx}{n}
163 \DeclareMathAlphabet {\mathsf} {OT1}{cmss}{m}{n}
164 \DeclareMathAlphabet {\mathit} {OT1}{cmr}{m}{it}
165 \DeclareMathAlphabet {\mathtt} {OT1}{cmtt}{m}{n}

```

Given the currently available fonts we cannot bold-en `\mathbf` and `\mathtt` but in principle one could use ‘ultra bold’ or something. The alphabets defined via `\DeclareSymbolFontAlphabet` will change automatically in a new math version if the corresponding symbol font changes.

```

166 \SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
167 \SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}

```

6.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurrences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but losing a bit of speed, so this may not stay this way).


```

168 \DeclareMathSizes{5}{5}{5}{5}
169 \DeclareMathSizes{6}{6}{5}{5}
170 \DeclareMathSizes{7}{7}{5}{5}
171 \DeclareMathSizes{8}{8}{6}{5}
172 \DeclareMathSizes{9}{9}{6}{5}
173 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
174 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
175 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
176 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
177 \DeclareMathSizes{\@xvipt}{\@xvipt}{\@xipt}{\@xpt}
178 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xipt}
179 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xvipt}

```

6.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by `IniTeX`. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

6.3.1 The letters

```

180 \DeclareMathSymbol{a}{\mathalpha}{letters}{'a}
181 \DeclareMathSymbol{b}{\mathalpha}{letters}{'b}
182 \DeclareMathSymbol{c}{\mathalpha}{letters}{'c}
183 \DeclareMathSymbol{d}{\mathalpha}{letters}{'d}
184 \DeclareMathSymbol{e}{\mathalpha}{letters}{'e}
185 \DeclareMathSymbol{f}{\mathalpha}{letters}{'f}
186 \DeclareMathSymbol{g}{\mathalpha}{letters}{'g}
187 \DeclareMathSymbol{h}{\mathalpha}{letters}{'h}
188 \DeclareMathSymbol{i}{\mathalpha}{letters}{'i}
189 \DeclareMathSymbol{j}{\mathalpha}{letters}{'j}
190 \DeclareMathSymbol{k}{\mathalpha}{letters}{'k}
191 \DeclareMathSymbol{l}{\mathalpha}{letters}{'l}
192 \DeclareMathSymbol{m}{\mathalpha}{letters}{'m}
193 \DeclareMathSymbol{n}{\mathalpha}{letters}{'n}
194 \DeclareMathSymbol{o}{\mathalpha}{letters}{'o}
195 \DeclareMathSymbol{p}{\mathalpha}{letters}{'p}
196 \DeclareMathSymbol{q}{\mathalpha}{letters}{'q}
197 \DeclareMathSymbol{r}{\mathalpha}{letters}{'r}
198 \DeclareMathSymbol{s}{\mathalpha}{letters}{'s}
199 \DeclareMathSymbol{t}{\mathalpha}{letters}{'t}
200 \DeclareMathSymbol{u}{\mathalpha}{letters}{'u}
201 \DeclareMathSymbol{v}{\mathalpha}{letters}{'v}
202 \DeclareMathSymbol{w}{\mathalpha}{letters}{'w}
203 \DeclareMathSymbol{x}{\mathalpha}{letters}{'x}
204 \DeclareMathSymbol{y}{\mathalpha}{letters}{'y}
205 \DeclareMathSymbol{z}{\mathalpha}{letters}{'z}

206 \DeclareMathSymbol{A}{\mathalpha}{letters}{'A}
207 \DeclareMathSymbol{B}{\mathalpha}{letters}{'B}
208 \DeclareMathSymbol{C}{\mathalpha}{letters}{'C}
209 \DeclareMathSymbol{D}{\mathalpha}{letters}{'D}
210 \DeclareMathSymbol{E}{\mathalpha}{letters}{'E}

```

```

211 \DeclareMathSymbol{F}{\mathalpha}{letters}{'F}
212 \DeclareMathSymbol{G}{\mathalpha}{letters}{'G}
213 \DeclareMathSymbol{H}{\mathalpha}{letters}{'H}
214 \DeclareMathSymbol{I}{\mathalpha}{letters}{'I}
215 \DeclareMathSymbol{J}{\mathalpha}{letters}{'J}
216 \DeclareMathSymbol{K}{\mathalpha}{letters}{'K}
217 \DeclareMathSymbol{L}{\mathalpha}{letters}{'L}
218 \DeclareMathSymbol{M}{\mathalpha}{letters}{'M}
219 \DeclareMathSymbol{N}{\mathalpha}{letters}{'N}
220 \DeclareMathSymbol{O}{\mathalpha}{letters}{'O}
221 \DeclareMathSymbol{P}{\mathalpha}{letters}{'P}
222 \DeclareMathSymbol{Q}{\mathalpha}{letters}{'Q}
223 \DeclareMathSymbol{R}{\mathalpha}{letters}{'R}
224 \DeclareMathSymbol{S}{\mathalpha}{letters}{'S}
225 \DeclareMathSymbol{T}{\mathalpha}{letters}{'T}
226 \DeclareMathSymbol{U}{\mathalpha}{letters}{'U}
227 \DeclareMathSymbol{V}{\mathalpha}{letters}{'V}
228 \DeclareMathSymbol{W}{\mathalpha}{letters}{'W}
229 \DeclareMathSymbol{X}{\mathalpha}{letters}{'X}
230 \DeclareMathSymbol{Y}{\mathalpha}{letters}{'Y}
231 \DeclareMathSymbol{Z}{\mathalpha}{letters}{'Z}

```

6.3.2 The digits

```

232 \DeclareMathSymbol{0}{\mathalpha}{operators}{'0}
233 \DeclareMathSymbol{1}{\mathalpha}{operators}{'1}
234 \DeclareMathSymbol{2}{\mathalpha}{operators}{'2}
235 \DeclareMathSymbol{3}{\mathalpha}{operators}{'3}
236 \DeclareMathSymbol{4}{\mathalpha}{operators}{'4}
237 \DeclareMathSymbol{5}{\mathalpha}{operators}{'5}
238 \DeclareMathSymbol{6}{\mathalpha}{operators}{'6}
239 \DeclareMathSymbol{7}{\mathalpha}{operators}{'7}
240 \DeclareMathSymbol{8}{\mathalpha}{operators}{'8}
241 \DeclareMathSymbol{9}{\mathalpha}{operators}{'9}

```

6.3.3 Punctuation, brace, etc. keys

```

242 \DeclareMathSymbol{!}{\mathclose}{operators}{"21}
243 \DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
244 \DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
245 \DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
246 \DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
247 \DeclareMathSymbol{.}{\mathord}{letters}{"3A}
248 \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
249 \DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
250 \DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
251 \DeclareMathSymbol{?}{\mathclose}{operators}{"3F}

```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```

252 %\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
253 %\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
254 %\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
255 %\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
256 %\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
257 %\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}

```

```

258 %\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
259 %\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

Should all of the following being activated by default? Probably not.

```

260 %\DeclareMathSymbol{\{}{\mathopen}{symbols}{"66}
261 %\DeclareMathSymbol{\}}{\mathclose}{symbols}{"67}
262 %\DeclareMathSymbol{\}\}{\mathord}{symbols}{"6E} % \backslash
263 \mathcode'\ =8000 % \space
264 \mathcode\' =8000 % ^\prime
265 \mathcode\'_ =8000 % \_

```

6.3.4 Delimitercodes for characters

[to be completed]

Finally, $\text{Init}\text{\TeX}$ sets all $\backslash\text{delcode}$ values to -1, except $\backslash\text{delcode}' = 0$

```

266 \DeclareMathDelimiter{()}{\mathopen}{operators}{"28}{largesymbols}{"00}
267 \DeclareMathDelimiter{)}{\mathclose}{operators}{"29}{largesymbols}{"01}
268 \DeclareMathDelimiter{[]}{\mathopen}{operators}{"5B}{largesymbols}{"02}
269 \DeclareMathDelimiter{[]}{\mathclose}{operators}{"5D}{largesymbols}{"03}

```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain $\text{\TeX}$. Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```

270 \DeclareMathDelimiter{<}{\mathopen}{symbols}{"68}{largesymbols}{"0A}
271 \DeclareMathDelimiter{>}{\mathclose}{symbols}{"69}{largesymbols}{"0B}
272 \DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
273 \DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```

274 \DeclareMathDelimiter{/}{\mathord}{operators}{"2F}{largesymbols}{"0E}
275 \DeclareMathSymbol{/}{\mathord}{letters}{"3D}

276 \DeclareMathDelimiter{|}{\mathord}{symbols}{"6A}{largesymbols}{"0C}
277 \expandafter\DeclareMathDelimiter\@backslashchar
278 \mathord}{symbols}{"6E}{largesymbols}{"0F}

```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

6.4 Symbols accessed via control sequences

6.4.1 Greek letters

```

279 \DeclareMathSymbol{\alpha}{\mathord}{letters}{"0B}
280 \DeclareMathSymbol{\beta}{\mathord}{letters}{"0C}
281 \DeclareMathSymbol{\gamma}{\mathord}{letters}{"0D}
282 \DeclareMathSymbol{\delta}{\mathord}{letters}{"0E}
283 \DeclareMathSymbol{\epsilon}{\mathord}{letters}{"0F}
284 \DeclareMathSymbol{\zeta}{\mathord}{letters}{"10}
285 \DeclareMathSymbol{\eta}{\mathord}{letters}{"11}
286 \DeclareMathSymbol{\theta}{\mathord}{letters}{"12}
287 \DeclareMathSymbol{\iota}{\mathord}{letters}{"13}
288 \DeclareMathSymbol{\kappa}{\mathord}{letters}{"14}
289 \DeclareMathSymbol{\lambda}{\mathord}{letters}{"15}
290 \DeclareMathSymbol{\mu}{\mathord}{letters}{"16}
291 \DeclareMathSymbol{\nu}{\mathord}{letters}{"17}

```

```

292 \DeclareMathSymbol{\xi}{\mathord}{letters}{18}
293 \DeclareMathSymbol{\pi}{\mathord}{letters}{19}
294 \DeclareMathSymbol{\rho}{\mathord}{letters}{1A}
295 \DeclareMathSymbol{\sigma}{\mathord}{letters}{1B}
296 \DeclareMathSymbol{\tau}{\mathord}{letters}{1C}
297 \DeclareMathSymbol{\upsilon}{\mathord}{letters}{1D}
298 \DeclareMathSymbol{\phi}{\mathord}{letters}{1E}
299 \DeclareMathSymbol{\chi}{\mathord}{letters}{1F}
300 \DeclareMathSymbol{\psi}{\mathord}{letters}{20}
301 \DeclareMathSymbol{\omega}{\mathord}{letters}{21}
302 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{22}
303 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{23}
304 \DeclareMathSymbol{\varpi}{\mathord}{letters}{24}
305 \DeclareMathSymbol{\varrho}{\mathord}{letters}{25}
306 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{26}
307 \DeclareMathSymbol{\varphi}{\mathord}{letters}{27}
308 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{00}
309 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{01}
310 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{02}
311 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{03}
312 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{04}
313 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{05}
314 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{06}
315 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{07}
316 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{08}
317 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{09}
318 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{0A}

```

6.4.2 Ordinary symbols

```

319 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{40}
320 \DeclareMathSymbol{\imath}{\mathord}{letters}{7B}
321 \DeclareMathSymbol{\jmath}{\mathord}{letters}{7C}
322 \DeclareMathSymbol{\ell}{\mathord}{letters}{60}
323 \DeclareMathSymbol{\wp}{\mathord}{letters}{7D}
324 \DeclareMathSymbol{\Re}{\mathord}{symbols}{3C}
325 \DeclareMathSymbol{\Im}{\mathord}{symbols}{3D}
326 \DeclareMathSymbol{\partial}{\mathord}{letters}{40}
327 \DeclareMathSymbol{\infty}{\mathord}{symbols}{31}
328 \DeclareMathSymbol{\prime}{\mathord}{symbols}{30}
329 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{3B}
330 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{72}
331 \DeclareMathSymbol{\top}{\mathord}{symbols}{3E}
332 \DeclareMathSymbol{\bot}{\mathord}{symbols}{3F}
333 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{34}
334 \DeclareMathSymbol{\forall}{\mathord}{symbols}{38}
335 \DeclareMathSymbol{\exists}{\mathord}{symbols}{39}
336 \DeclareMathSymbol{\neg}{\mathord}{symbols}{3A}

```

Alias:

```

337 % \let\not=\neg
338 \DeclareMathSymbol{\not}{\mathord}{symbols}{3A}
339 \DeclareMathSymbol{\flat}{\mathord}{letters}{5B}
340 \DeclareMathSymbol{\natural}{\mathord}{letters}{5C}
341 \DeclareMathSymbol{\sharp}{\mathord}{letters}{5D}

```

```

342 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
343 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
344 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
345 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}

346 \DeclareRobustCommand\hbar{{\mathchar'26\mkern-9mu}}
347 \DeclareRobustCommand\surd{{\mathchar"1270}}
348 \DeclareRobustCommand\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crrc
349     \not\mathrel{\mkern14mu}\crrc
350     \noalign{\nointerlineskip}
351     \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crrc}}}}

```

6.4.3 Large Operators

```

352 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
353 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
354 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
355 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
356 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
357 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
358 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
359 \DeclareRobustCommand\int{\intop\nolimits}
360 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
361 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
362 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
363 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
364 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
365 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
366 \DeclareRobustCommand\oint{\ointop\nolimits}
367 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
368 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}

```

6.4.4 Binary symbols

```

369 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
370 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
371 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
372 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}

```

Alias:

```

373 % \let \varbigtriangledown \bigtriangledown
374 % \let \varbigtriangleup \bigtriangleup
375 \DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{"34}
376 \DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{"35}

```

These last two synonyms are needed because the `stmaryrd` package redefines them as Operators.

```

377 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
378 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}

```

Alias:

```

379 % \let \land=\wedge
380 % \let \lor=\vee
381 \DeclareMathSymbol{\land}{\mathbin}{symbols}{"5E}
382 \DeclareMathSymbol{\lor}{\mathbin}{symbols}{"5F}
383 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
384 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
385 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}

```

```

386 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
387 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
388 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
389 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
390 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
391 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
392 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
393 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
394 \DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
395 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
396 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
397 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
398 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
399 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
400 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
401 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
402 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
403 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
404 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
405 \DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
406 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
407 \DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
408 \DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}

```

6.4.5 Relations

```

409 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
410 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
411 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
412 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
413 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
414 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
415 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
416 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
417 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
418 \DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
419 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
420 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
421 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
422 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
423 \DeclareRobustCommand\neq{\not=}

```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```

424 \DeclareRobustCommand\ne{\not=}

```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```

425 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
426 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}

```

Alias:

```

427 % \let\le=\leq
428 % \let\ge=\geq
429 \DeclareMathSymbol{\le}{\mathrel}{symbols}{"14}
430 \DeclareMathSymbol{\ge}{\mathrel}{symbols}{"15}

```

```

431 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
432 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
433 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
434 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
435 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
436 \DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
437 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
438 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
439 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
440 \DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
441 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}

```

Alias:

```

442 % \let\owns=\ni
443 \DeclareMathSymbol{\owns}{\mathrel}{symbols}{"33}
444 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
445 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
446 \DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
447 \DeclareMathSymbol{\leftrightharpoonup}{\mathrel}{symbols}{"24}
448 \DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
449 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}

```

Alias:

```

450 % \let\gets=\leftarrow
451 % \let\to=\rightarrow
452 \DeclareMathSymbol{\gets}{\mathrel}{symbols}{"20}
453 \DeclareMathSymbol{\to}{\mathrel}{symbols}{"21}
454 \DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
455 \DeclareRobustCommand\mapsto{\mapstochar\rightarrow}
456 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
457 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
458 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
459 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
460 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
461 \DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
462 \DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
463 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
464 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
465 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
466 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}

```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```

467 \DeclareRobustCommand
468 \cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
469 \def\@vereq#1#2{\lower.5p\p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
470 \ialign{\$m@th#1\hfil##\hfil$\crrc#2\crrc=\crrc}}
471 \DeclareRobustCommand
472 \notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
473 \def\c@ncel#1#2{\m@th\oalign{\$hfil#1\mkern1mu\hfil$\crrc$#1#2$}}
474 \DeclareRobustCommand
475 \rightleftharpoons{\mathrel{\mathpalette\rightharpoonup}}

```

```

476 \def\rhlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
477 \hbox{$#1\rightarrow$}\cr
478 \hbox{$#1\leftarrow$}\cr}}}
479 \DeclareRobustCommand
480 \doteq{\buildrel\textstyle.\over=}

```

6.4.6 Arrows

```

481 \DeclareRobustCommand
482 \joinrel{\mathrel{\mkern-3mu}}
483 \DeclareRobustCommand
484 \relbar{\mathrel{\smash-}} % \smash, because -
485 % has the same height as +

```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being “math active” expanding to `\futurelet...`. This might be the case when packages are implementing shorthands for math, e.g. `=>` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can’t do this now as it would break other math layouts where characters are in different places (since those wouldn’t know about the need for a new command name).

```

486 \DeclareRobustCommand
487 \Relbar{\mathrel{=}}
488 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
489 \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
490 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
491 \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
492 \DeclareRobustCommand
493 \bowtie{\mathrel{\triangleright\joinrel\mathrel{\triangleleft}}}
494 \DeclareRobustCommand
495 \models{\mathrel{|\joinrel\Relbar}}
496 \DeclareRobustCommand
497 \Longrightarrow{\Relbar\joinrel\rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make them robust.

```

498 \DeclareRobustCommand\longrightarrow
499 {\relbar\joinrel\rightarrow}
500 \DeclareRobustCommand\longleftarrow
501 {\leftarrow\joinrel\relbar}
502 \DeclareRobustCommand
503 \Longleftarrow{\Leftarrow\joinrel\Relbar}
504 \DeclareRobustCommand
505 \longmapsto{\mapstochar\longrightarrow}
506 \DeclareRobustCommand
507 \longlefttrightarrow{\leftarrow\joinrel\rightarrow}
508 \DeclareRobustCommand
509 \Longlefttrightarrow{\Leftarrow\joinrel\rightarrow}
510 \DeclareRobustCommand
511 \iff{\;\Longlefttrightarrow\;}

```

6.4.7 Punctuation symbols

```

512 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
513 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
514 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}

```


This is commented out, since `\ldots` is now defined in `ltoutenc.dtx`.

```

515 %\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
516 %\DeclareRobustCommand\ldots
517 %      {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots$,}\fi}
518 \DeclareRobustCommand
519   \cdots{\mathinner{\cdotp\cdotp\cdotp}}
520 \DeclareRobustCommand
521   \vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
522     \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
523 \DeclareRobustCommand
524   \ddots{\mathinner{\mkern1mu\raise7\p@
525     \vbox{\kern7\p@\hbox{.}}\mkern2mu
526     \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}

```

6.4.8 Math accents

```

527 \DeclareMathAccent{\acute}{\mathalpha}{operators}{13}
528 \DeclareMathAccent{\grave}{\mathalpha}{operators}{12}
529 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{7F}
530 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{7E}
531 \DeclareMathAccent{\bar}{\mathalpha}{operators}{16}
532 \DeclareMathAccent{\breve}{\mathalpha}{operators}{15}
533 \DeclareMathAccent{\check}{\mathalpha}{operators}{14}
534 \DeclareMathAccent{\hat}{\mathalpha}{operators}{5E}
535 \DeclareMathAccent{\vec}{\mathord}{letters}{7E}
536 \DeclareMathAccent{\dot}{\mathalpha}{operators}{5F}
537 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{65}
538 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{62}

```

For some reason plain T_EX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```

539 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{17}

```

6.4.9 Radicals

```

540 \DeclareMathRadical{\sqrtsign}{symbols}{70}{largesymbols}{70}

```

6.4.10 Over and under something, etc

```

541 \DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crrc
542   \rightarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
543   $\hfil\displaystyle{#1}\hfil$\crrc}}}
544 \DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crrc
545   \leftarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}%
546   $\hfil\displaystyle{#1}\hfil$\crrc}}}
547 \DeclareRobustCommand\overbrace[1]
548   {\mathop{\vbox{\m@th\ialign{##\crrc\noalign{\kern3\p@}%
549     \downbracefill\crrc\noalign{\kern3\p@\nointerlineskip}%
550     $\hfil\displaystyle{#1}\hfil$\crrc}}}\limits}
551 \DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crrc
552   $\hfil\displaystyle{#1}\hfil$\crrc
553   \noalign{\kern3\p@\nointerlineskip}%
554   \upbracefill\crrc\noalign{\kern3\p@}}}\limits}

```

(quite a waste of tokens, IMHO — Frank)

```

555 \DeclareRobustCommand\skew[3]
556   {\{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
557   #2{\mkern-\muskip\z@#3}\mkern\muskip\z@}\mkern-\muskip\z@}\}

```

```

558 \DeclareRobustCommand\rightarrowfill{\smash-\mkern-7mu%
559 \cleaders\hbox{\mkern-2mu\smash-\mkern-2mu$\hfill
560 \mkern-7mu\mathord\rightarrow$}
561 \DeclareRobustCommand\leftarrowfill{\smash-\mkern-7mu%
562 \cleaders\hbox{\mkern-2mu\smash-\mkern-2mu$\hfill
563 \mkern-7mu\smash-$}
564 \DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{7A}
565 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{7B}
566 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{7C}
567 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{7D}
568 \DeclareRobustCommand\downbracefill{\smash-\mkern-7mu%
569 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
570 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
571 \DeclareRobustCommand\upbracefill{\smash-\mkern-7mu%
572 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
573 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}

```

6.4.11 Delimiters

```

574 \DeclareMathDelimiter{\lmoustache} % top from (, bottom from )
575 {\mathopen}{largesymbols}{7A}{largesymbols}{40}
576 \DeclareMathDelimiter{\rmoustache} % top from ), bottom from (
577 {\mathclose}{largesymbols}{7B}{largesymbols}{41}
578 \DeclareMathDelimiter{\arrowvert} % arrow without arrowheads
579 {\mathord}{symbols}{6A}{largesymbols}{3C}
580 \DeclareMathDelimiter{\Arrowvert} % double arrow without arrowheads
581 {\mathord}{symbols}{6B}{largesymbols}{3D}
582 \DeclareMathDelimiter{\Vert}
583 {\mathord}{symbols}{6B}{largesymbols}{0D}

```

`\DeclareMathDelimiter` produces a command that is robust (with an internal macro containing the payload) so we should not use `\let` for making an alias

```

584 %\let\l=\Vert
585 \DeclareMathDelimiter{\l}
586 {\mathord}{symbols}{6B}{largesymbols}{0D}
587 \DeclareMathDelimiter{\v}
588 {\mathord}{symbols}{6A}{largesymbols}{0C}
589 \DeclareMathDelimiter{\uparrow}
590 {\mathrel}{symbols}{22}{largesymbols}{78}
591 \DeclareMathDelimiter{\downarrow}
592 {\mathrel}{symbols}{23}{largesymbols}{79}
593 \DeclareMathDelimiter{\updownarrow}
594 {\mathrel}{symbols}{6C}{largesymbols}{3F}
595 \DeclareMathDelimiter{\Uparrow}
596 {\mathrel}{symbols}{2A}{largesymbols}{7E}
597 \DeclareMathDelimiter{\Downarrow}
598 {\mathrel}{symbols}{2B}{largesymbols}{7F}
599 \DeclareMathDelimiter{\Updownarrow}
600 {\mathrel}{symbols}{6D}{largesymbols}{77}
601 \DeclareMathDelimiter{\backslash} % for double coset G\backslash H
602 {\mathord}{symbols}{6E}{largesymbols}{0F}
603 \DeclareMathDelimiter{\rangle}
604 {\mathclose}{symbols}{69}{largesymbols}{0B}
605 \DeclareMathDelimiter{\langle}
606 {\mathopen}{symbols}{68}{largesymbols}{0A}

```

```

607 \DeclareMathDelimiter{\rbrace
608     {\mathclose}{symbols}{"67}{largesymbols}{"09}
609 \DeclareMathDelimiter{\lbrace
610     {\mathopen}{symbols}{"66}{largesymbols}{"08}
611 \DeclareMathDelimiter{\rceil}
612     {\mathclose}{symbols}{"65}{largesymbols}{"07}
613 \DeclareMathDelimiter{\lceil}
614     {\mathopen}{symbols}{"64}{largesymbols}{"06}
615 \DeclareMathDelimiter{\rfloor}
616     {\mathclose}{symbols}{"63}{largesymbols}{"05}
617 \DeclareMathDelimiter{\lfloor}
618     {\mathopen}{symbols}{"62}{largesymbols}{"04}

```

`\lgroup` `\rgroup` `\bracevert` There are three plain TeX delimiters which are not fully supported by NFSS, since they partly point into a bold cmr font. Allocating a full symbol font, just to have three delimiters seems a bit too much given the limited space available. For this reason only the extensible sizes are supported. If this is not desired one can use, without losing portability, define `\mathbf` and `\mathtt` as font symbol alphabet (setting up `cmr/bx/n` and `cmtt/m/n` as symbol fonts first) and modify the delimiter declarations to point with their small variant to those symbol fonts. (This is done in `oldlfont.dtx` so look there for examples.)

```

619 \DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
620     {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
621 \DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
622     {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
623 \DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
624     {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}

```

(End of definition for \lgroup, \rgroup, and \bracevert.)

6.5 Math versions of text commands

The `\mathunderscore` here is really a text definition, so it has been put back into `loutenc.dtx` (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as `\P`, `\$`, etc.

```

\mathparagraph These math symbols are not in plain TeX.
\mathsection 625 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\mathdollar 626 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\mathsterling 627 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
\mathunderscore 628 \DeclareRobustCommand\mathsterling{\mathit{\mathchar"7024}}
629 \DeclareRobustCommand\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}

```

(End of definition for \mathparagraph and others.)

`\mathellipsis` This is plain TeX's `\ldots`.

```

630 \DeclareRobustCommand\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%

```

(End of definition for \mathellipsis.)

6.6 Other special functions and parameters

6.6.1 Biggggg

```

631 </math>
632 <*math | latexrelease>
633 <latexrelease>\IncludeInRelease{2018/12/01}%
634 <latexrelease>          {\Big}{Start LR-mode}%
635 \DeclareRobustCommand\big[1]{\leavevmode@ifvmode
636   {\hbox{$\left#1\ vbox to8.5\p@{} \right.\n@space$}}}
637 \DeclareRobustCommand\Big[1]{\leavevmode@ifvmode
638   {\hbox{$\left#1\ vbox to11.5\p@{} \right.\n@space$}}}
639 \DeclareRobustCommand\bigg[1]{\leavevmode@ifvmode
640   {\hbox{$\left#1\ vbox to14.5\p@{} \right.\n@space$}}}
641 \DeclareRobustCommand\Bigg[1]{\leavevmode@ifvmode
642   {\hbox{$\left#1\ vbox to17.5\p@{} \right.\n@space$}}}
643 </math | latexrelease>
644 <latexrelease>\EndIncludeInRelease
645 <latexrelease>\IncludeInRelease{0000/00/00}%
646 <latexrelease>          {\Big}{Start LR-mode}%
647 <latexrelease>\def\big#1{{\hbox{$\left#1\ vbox to8.5\p@{} \right.\n@space$}}}
648 <latexrelease>\def\Big#1{{\hbox{$\left#1\ vbox to11.5\p@{} \right.\n@space$}}}
649 <latexrelease>\def\bigg#1{{\hbox{$\left#1\ vbox to14.5\p@{} \right.\n@space$}}}
650 <latexrelease>\def\Bigg#1{{\hbox{$\left#1\ vbox to17.5\p@{} \right.\n@space$}}}
651 <latexrelease>\EndIncludeInRelease
652 <*math>

653 \def\n@space{\null\delimiterspace\z@ \m@th}

```

6.6.2 The log-like functions

`\operator@font` The `\operator@font` determines the symbol font used for log-like functions.

```

654 \def\operator@font{\mathgroup\symoperators}

```

(End of definition for \operator@font.)

6.6.3 Parameters

```

655 \thinmuskip=3mu
656 \medmuskip=4mu plus 2mu minus 4mu
657 \thickmuskip=5mu plus 5mu

```

This finishes the low-level setup in `fontmath.ltx`.

```

658 </math>

```

7 Default cfg files

We provide default `cfg` files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```

659 <*cfgtext | cfgmath | cfgprel>
660 %%
661 %%
662 %%
663 %% Load the standard setup:
664 %%
665 <+cfgtext>\input{fonttext.ltx}

```

```

666 <+cfgmath>\input{fontmath.ltx}
667 <+cfgprel>\input{preload.ltx}
668 %%
669 %% Small changes could go here; see documentation in cfgguide.tex for
670 %% allowed modifications.
671 %%
672 %% In particular it is not allowed to misuse this configuration file
673 %% to modify internal LaTeX commands!
674 %%
675 %% If you use this file as the basis for configuration please change
676 %% the \ProvidesFile lines to clearly identify your modification, e.g.,
677 %%
678 <+cfgtext>%% \ProvidesFile{fonttext.cfg}[2001/06/01
679 <+cfgmath>%% \ProvidesFile{fonttext.cfg}[2001/06/01
680 <+cfgprel>%% \ProvidesFile{preload.cfg}[2001/06/01
681 %% Customised local font setup]
682 %%
683 %%
684 </cfgtext | cfgmath | cfgprel>

```

File 31

preload.dtx

1 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by L^AT_EX 2_ε). It will be used to generate the following files:

preload.min	minimal subset of fonts necessary to run NFSS2
preload.ori	preload of CM fonts similar to the old <code>lfonts.tex</code>
preload.ltx	The standard selection of preloads
cmpreload.xpt	preload of CM fonts for 10pt document size
cmpreload.xip	preload of CM fonts for 11pt document size
cmpreload.xii	preload of CM fonts for 12pt document size
dcpreload.xpt	preload of DC fonts for 10pt size
dcpreload.xip	preload of DC fonts for 11pt size
dcpreload.xii	preload of DC fonts for 12pt size

These files are for installations that make use of Computer Modern fonts either old encoding (OT1) or Cork encoding (T1). The Computer Modern fonts with Cork encoding are known as DC-fonts.

Most important is `preload.ltx` which is used during format generation. You are *not* allowed to change this file.

2 Customization

You can customize the preloaded fonts in your L^AT_EX 2_ε system by installing a file with the name `preload.cfg`. If this file exists it will be used in place of the system file `preload.ltx`. You can, for example, copy one of the files mentioned above (that can be generated from this source) to `preload.cfg`.

Or you can define completely other preloads. In that case start from `preload.min` since that contains the fonts that have to be preloaded by **all** L^AT_EX 2_ε systems.

Avoid using `preload.ori`, it will load so many fonts that on most installations it is nearly impossible to load other font families afterwards. This file is only generated to show what fonts have been preloaded by L^AT_EX 2.09.

If you normally use other fonts than Computer Modern `preload.min` might be best.

Warning: If you preload fonts with encodings other than the normally supported encodings you have to declare that encoding in a `fontdef.cfg` configuration file (see the documentation in the file `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

3 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

driver	produce a documentation driver file
preload	produce a preload... file
cm	for OT1 encoded Computer Modern
dc	for T1 encoded Computer Modern
min	produce minimal subset
xpt	produce 10pt preloads
xipt	produce 11pt preloads
xipt	produce 12pt preloads
ori	produce preloads similar to old <code>lfonts.tex</code>
tex	produce <code>preload.ltx</code>

A typical DOCSTRIP command file would then have entries like:

```
generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

4 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```

1 <*driver>
2 \documentclass{ltxdoc}
3 %\OnlyDescription % comment out for implementation details
4 \begin{document}
5   \DocInput{preload.dtx}
6 \end{document}
7 </driver>
```

5 The code

We begin by loading the math extension font (cmex10) and the L^AT_EX line and circle fonts. It is necessary to do this explicitly since these are used by the L^AT_EX format. Since the internal font name contains / characters and digits we construct the name via `\csname`. These are the only fonts (!) that must be loaded in this file.

All `\DeclarePreloadSizes` can be removed or others can be added, they only influence the processing speed.

```

8 \expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
9 \font\tenln =line10 \font\tenlnw =linew10\relax
10 \font\tencirc=lcircle10 \font\tencircw=lcirclew10\relax
```

The above fonts should not be touched but anything below this point here in the preload suggestions can be modified without any problems.

```

11 <-tex>%*****
12 <-tex>% Start any modification below this point **
13 <-tex>%*****
14 <-tex>
15 %%
16 %% Computer Modern Roman:
17 %%-----
```

```

18 <*ori>
19 \DeclarePreloadSizes{OT1}{cmr}{m}{n}
20     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
21 \DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
22 \DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
23 \DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
24 </ori>
25 <+xpt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
26 <+xpt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
27 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
28 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
29 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
30 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
31 %%
32 %% Computer Modern Sans:
33 %%-----
34 <+ori> \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
35 %%
36 %% Computer Modern Typewriter:
37 %%-----
38 <+ori> \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
39 %%
40 %% Computer Modern Math:
41 %%-----
42 <*ori>
43 \DeclarePreloadSizes{OML}{cmm}{m}{it}
44     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
45 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}
46     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
47 </ori>

```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```

48 <*xpt>
49 \DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
50 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
51 </xpt>
52 <*xipt>
53 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
54 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
55 </xipt>
56 <*xipt>
57 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
58 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
59 </xipt>
60 %%
61 %% LaTeX symbol fonts:
62 %%-----
63 <*ori>
64 \DeclarePreloadSizes{U}{lasy}{m}{n}
65     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
66 </ori>
67 </preload>

```


File 32

lftntcmd.dtx

Abstract

The commands defined in this file `lftntcmd` are part of the kernel code for $\text{\LaTeX} 2_{\epsilon}/\text{NFSS2}$.

It is also meant to serve as documentation for package writers since it demonstrates how to define high-level font changing commands using a small number of creator functions.

1 Introduction

Font changes such as `\bfseries`, `\sffamily`, etc. are declarations; this means that their scope is delimited by the grouping structure, either by the next `\end` of some environment or by explicitly using a group, e.g., writing something like `{\bfseries...}` in the source. If you make the mistake of writing `\bfseries{...}` (thinking of `\bfseries` as a command with one argument) then the result is rather striking.

Font declarations are an artifact of the $\text{\TeX}$ system and for several reasons it is better to avoid them on the user level whenever possible. In $\text{\LaTeX}3$ they will probably all be replaced by environments and by font commands taking one argument.

This file defines a creator function for such declarative font switches. This function creates commands which can be used in both math and text.

This file also defines a number of high-level commands (all starting with `\text..`) that have one argument and typeset this argument in the requested way. Thus these commands are for typesetting short pieces of text in a specific family, series or shape. These are all produced as examples of the use of a creator function which is itself also defined in this file.

Table 3 shows all these high-level commands in action. A further advantage of using these commands is that they automatically take care of any necessary italic correction on either side of their argument.

Thus, when using such commands, one does not have to worry about forgetting the italic correction when changing fonts. Only in very few situations is this additional space wrong but, for example, most typographers recommend omitting the italic correction if a small punctuation character, like a comma, directly follows the font change. Since the amount of correction required is partly a matter of taste, you can define in what situations the italic correction should be suppressed. This is done by putting the characters that should cancel a preceding italic correction in the list `\nocorrlist`.³⁹ The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,  
the \emph{proper} use of italic corrections is  
automatically taken care of}. Only
```

³⁹Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

<i>Command</i>	<i>Corresponds to</i>	<i>Action</i>
<code>\textnormal{..}</code>	<code>\normalfont</code>	Typeset argument in normal family
<code>\textrm{..}</code>	<code>\rmfamily</code>	Typeset argument in roman family
<code>\textsf{..}</code>	<code>\sffamily</code>	Typeset argument in sans serif family
<code>\texttt{..}</code>	<code>\ttfamily</code>	Typeset argument in typewriter family
<code>\textmd{..}</code>	<code>\mdseries</code>	Typeset argument in medium series
<code>\textbf{..}</code>	<code>\bfseries</code>	Typeset argument in bold series
<code>\textup{..}</code>	<code>\upshape</code>	Typeset argument in normal shape
<code>\textit{..}</code>	<code>\itshape</code>	Typeset argument in <i>italic</i> shape
<code>\textsl{..}</code>	<code>\slshape</code>	Typeset argument in <i>slanted</i> shape
<code>\textsc{..}</code>	<code>\scshape</code>	Typeset argument in SMALL CAPS shape
<code>\emph{..}</code>	<code>\em</code>	Typeset argument <i>emphasized</i>

Table 3: Font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable. They automatically take care of any necessary italic correction on either side of the argument.

`\emph{sometimes}` one has to help `\LaTeX{}` by adding a `\verb=\nocorr=` command.

which results in:

When using the NFSS high-level commands, the proper use of italic corrections is automatically taken care of. Only sometimes one has to help L^AT_EX by adding a `\nocorr` command.

In contrast, the use of the declaration forms is often more appropriate when you define your own commands or environments.

```
\newenvironment{bfitemize}{\begin{itemize}\normalfont\bfseries}
                          {\end{itemize}}
\begin{bfitemize}
\item This environment produces boldface items.
\item It is defined in terms of \LaTeX's
      \texttt{itemize} environment and NFSS
      declarations.
\end{bfitemize}
```

This gives:

- **This environment produces boldface items.**
- **It is defined in terms of L^AT_EX's `itemize` environment and NFSS declarations.**

In addition to global customization of when to insert the italic correction, it is of course sometimes necessary to explicitly insert one with `\/`.

It is also possible to suppress the italic correction in individual instances. For this, the command `\nocorr` is provided.

The `\nocorr` must appear as the first or last token inside the braces of the argument of the `\text...` commands, at that end of the text where you wish to suppress the italic correction.

It is worth pointing out here that inserting a `\/` in places where it can have no function (i.e. anywhere except immediately after a slanted letter) is not an error—it will just be silently ignored. Unfortunately this is not true if the redefinition of `\/` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\/`.

2 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

The algorithm for deciding whether to put in an italic correction is not very subtle: one is added whenever the newly current font is not itself positively sloped, unless the next token is a character in the ‘nocorr’ list. At the end of the text this is done after closing the group so as to check the ‘outer font’. Note that this will often result in adding an italic correction token after a character in an unsloped font; we believe (in early 2003) that this is perhaps inefficient but not dangerous.

It also now checks for empty contents of the text command and optimizes this case. Some care is also taken to check that doing dangerous things in vertical mode is avoided.

The italic correction token is added to the horizontal list before (in the list) an immediately preceding non-zero glob of glue (skip) and any non-zero penalty preceding that since, in the typical case, this puts it immediately after the last character in the preceding word.

Note that it is necessary to put in the `\aftergroup\maybe@ic` at the end of the group so that it comes after any other aftergroup tokens and immediately before the following tokens. It is also necessary to remove the `\fi` from the token list before the group ends; this is done by adding an `\expandafter` just before the closing brace.

```

1  (*2kernel)
2  \def \DeclareTextFontCommand #1#2{%
3    \DeclareRobustCommand#1[1]{%
4      \ifmmode
5        \nfss@text{#2##1}%
6      \else
7        \hmode@bgroup
8        \text@command{##1}%
9        #2\check@icl ##1\check@icr
10       \expandafter
11       \egroup
12     \fi
13           }%
14 }

```

(End of definition for `\DeclareTextFontCommand`.)

```

\textrm Now we define the \text<family> commands in terms of the above; \textttt does not
\textsf look very nice!
\texttt 15 \DeclareTextFontCommand{\textrm}{\rmfamily}
\textnormal 16 \DeclareTextFontCommand{\textsf}{\sffamily}
17 \DeclareTextFontCommand{\texttt}{\ttfamily}
18 \DeclareTextFontCommand{\textnormal}{\normalfont}

```

(End of definition for `\textrm` and others.)

```

\textbf For the series attribute:
\textmd 19 \DeclareTextFontCommand{\textbf}{\bfseries}
20 \DeclareTextFontCommand{\textmd}{\mdseries}

```

(End of definition for `\textbf` and `\textmd`.)

```

\textit And for the shapes:
\textsl 21 \DeclareTextFontCommand{\textit}{\itshape}
\textsc 22 \DeclareTextFontCommand{\textsl}{\slshape}
\textup 23 \DeclareTextFontCommand{\textsc}{\scshape}
24 \DeclareTextFontCommand{\textup}{\upshape}

```

(End of definition for `\textit` and others.)

```

\textulc
\textsw 25 </2ekernel>
\textssc 26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/02/02}%
28 <latexrelease> \textulc{Additional text commands}%
29 \DeclareTextFontCommand{\textulc}{\ulcshape}
30 \DeclareTextFontCommand{\textsw}{\swshape}
31 \DeclareTextFontCommand{\textssc}{\sscshape}
32 </2ekernel | latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <latexrelease>\IncludeInRelease{0000/00/00}%
35 <latexrelease> \textulc{Additional text commands}%
36 <latexrelease>
37 <latexrelease>\let\textulc\@undefined
38 <latexrelease>\let\textsw\@undefined
39 <latexrelease>\let\textssc\@undefined
40 <latexrelease>\EndIncludeInRelease
41 <*2ekernel>

```

(End of definition for `\textulc`, `\textsw`, and `\textssc`.)

`\emph` Finally we have the `\em` font change declaration of L^AT_EX. The corresponding definition with argument is

```
42 \DeclareTextFontCommand{\emph}{\em}
```

(End of definition for `\emph`.)

```

\strongemph
43 \DeclareTextFontCommand{\strongemph}{\strongem}

```

(End of definition for `\strongemph`.)

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

```
44 \let \nocorr \relax
```

(End of definition for `\nocorr`.)

`\check@ic1` We define these defaults in case some error causes them to be expanded at the wrong
`\check@icr` time.

```
45 \let \check@ic1 \@empty
```

```
46 \let \check@icr \@empty
```

(End of definition for `\check@ic1` and `\check@icr`.)

`\text@command` This checks for a `\nocorr` as the first token in its argument and also for one in any other
`\check@nocorr@` position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the ‘empty’ case? It is efficient but typographically it is, strictly, incorrect!

```
47 \def \text@command #1{%
48   \edef \reserved@a {\unexpanded{#1}}%
49   \ifx \reserved@a \@empty
50     \let \check@ic1 \@empty
51     \let \check@icr \@empty
52   \else
```

`\space` is a reserved word in L^AT_EX or actually already in plain T_EX. If somebody really redefines it so many things will break that I don’t see any reason to make this routine here slower than necessary.

```
53 %   \def \reserved@b { }%
54 %   \ifx \reserved@a \reserved@b
55   \ifx \reserved@a \space
56     \let \check@ic1 \@empty
57     \let \check@icr \@empty
58   \else
59     \check@nocorr@ #1\nocorr\@nil
60   \fi
61 \fi
62 }
63 \def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialized here to their values in the normal case.

```
64 \let \check@ic1 \maybe@ic
65 \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
66 \def \reserved@a {\nocorr}%
67 \def \reserved@b {#1}%
68 \def \reserved@c {#3}%
69 \ifx \reserved@a \reserved@b
70   \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@ic1` should be empty.

```
71   \let \check@ic1 \@empty
72 \else
```

Otherwise there is a `\nocorr` both at the start and elsewhere, so no italic corrections should be added.

```

73      \let \check@icl \@empty
74      \let \check@icr \@empty
75      \fi
76      \else
77      \ifx \reserved@c \@empty

```

In this case there is no `\nocorr` anywhere, so we need to check for an italic correction at both the beginning and the end. This has been set up as the default so no code is needed here.

```

78      \else

```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```

79      \let \check@icr \@empty
80      \fi
81      \fi
82  }

```

(End of definition for \text@command and \check@nocorr@.)

`\ifmaybe@ic` Switch used solely within `\maybe@ic` not interfering with other switches.

```

83  \newif\ifmaybe@ic

```

(End of definition for \ifmaybe@ic.)

`\maybe@ic` These macros implement the italic correction.

```

\maybe@ic@
84  \def \maybe@ic {\futurelet\@let@token\maybe@ic@}
85  \def \maybe@ic@ {%

```

We first check to see if the current font is positively sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope! Or is this an urban myth?) It has been suggested that this should test against a small positive value, but what?

```

86  \ifdim \fontdimen\@ne\font>\z@
87  \else
88  \maybe@ictrue

```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```

89      \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
90      \nocorrlist

```

We have to hide the `\@let@token` in the macro `\t@st@ic` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```

91      \do \t@st@ic

```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of `\nolinebreak`.

```

92      \ifmaybe@ic \sw@slant \fi
93      \fi
94  }

```

(End of definition for \maybeic and \maybeic@.)

`\t@st@ic` The next token in the input stream is stored in `\@let@token` via a `\let`, the current token from `\nocorrlist` is stored via `\def` in `\reserved@a`. To compare them we have to fiddle around a bit.

If the only things to check were characters then this could be done via an `\if` thus their catcodes would not matter; but this will not work whilst `\futurelet` is used above.

```

95 \def \t@st@ic {%
96   \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
97   \ifx\reserved@b\@let@token

```

If they are the same we record the fact and jump out of the loop.

```

98     \maybeicfalse
99     \@break@tfor
100   \fi
101 }

```

(End of definition for `\t@st@ic`.)

`\sw@slant` The definition of the mysterious `\sw@slant` command is as follows.
`\fix@penalty`

```

102 \def \sw@slant {%

```

It is surely correct to put in an italic correction when there is no skip. If the last thing on the list is actually a zero skip (including things whose dimension part is zero, such as `\hfill`), or anything other than a character, then the italic correction will have no effect.

In order to work correctly with unbreakable spaces from `~` (and other common forms of line-breaking control) we also move back across a penalty before the glue.

```

103   \ifdim \lastskip=\z@
104     \fix@penalty
105   \else
106     \skip@ \lastskip
107     \unskip
108     \fix@penalty
109     \hskip \skip@
110   \fi
111 }

```

The above code means: “If there is a non-zero space just before the current position (`\ifdim...`) save the amount of that space (`\skip@\lastskip`), remove it (`\unskip`), then do a similar thing if there is a penalty just before the skip, and finally put the space back in.”

Since zero glue cannot be distinguished in this context from no glue, we dare not put in an `\hskip` in this case as this may produce an unwanted breakpoint. This is not satisfactory.

The penalty before the glue is handled similarly, with the same caveats concerning the zero case. Is this the first recorded use of `\unpenalty` in standard L^AT_EX code?

```

112 \def \fix@penalty {%
113   \ifnum \lastpenalty=\z@
114     @@italiccorr
115   \else
116     \count@ \lastpenalty
117     \unpenalty

```

```

118     \@@italiccorr
119     \penalty \count@
120     \fi
121 }

```

(End of definition for \sw@slant and \fix@penalty.)

\nocorrlist This holds the list of characters that should prevent italic correction. They should be ordered by decreasing frequency of use. If any such character is made active later on one needs to redefine the list so that the active character becomes part of it.

```

122 \def \nocorrlist {,.}

```

(End of definition for \nocorrlist.)

\nfss@text This command will by default behave like a L^AT_EX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```

123 \ifx \nfss@text\undefined
124   \def \nfss@text {\leavevmode\hbox}
125 \fi

```

(End of definition for \nfss@text.)

\DeclareOldFontCommand This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{<font-change decls>} <math-alphabet>`

Here `\fn` is the font-declaration command being defined, `<font-change decls>` is the declaration it will expand to in text-mode, and `<math-alphabet>` is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```

\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}

```

```

126 \def \DeclareOldFontCommand #1#2#3{%
127   \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
128 }

```

(End of definition for \DeclareOldFontCommand.)

\@fontswitch **\@math@egroup** These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```

\@math@egroup
\@math@egroup
129 \def \@fontswitch #1#2{%
130   \ifmmode
131     \let \math@bgroup \relax
132     \def \math@egroup {\let \math@bgroup \@math@bgroup
133                       \let \math@egroup \@math@egroup}%

```


We need to have a `\relax` in the following line in case the `#2` is something like `\mathsf` grabbing the next token as an argument. For this reason the code also uses explicit arguments again (see pr/1275).

```

134     #2\relax
135   \else
136     #1%
137   \fi
138 }
139 \let \@@math@bgroup \math@bgroup
140 \let \@@math@egroup \math@egroup

(End of definition for \@fontswitch, \@@math@egroup, and \@@math@egroup.)
These commands are available only in the preamble.
141 \@onlypreamble \DeclareTextFontCommand
142 \@onlypreamble \DeclareOldFontCommand

```

3 Initialization

```

\normalsize This is defined to produce an error.
143 \def\normalsize{%
144   \@latex@error {The font size command \protect\normalsize\space
145                 is not defined:\MessageBreak
146                 there is probably something wrong with
147                 the class file}\@eha
148 }
149 \</2ekernel>

(End of definition for \normalsize.)

```

File 33

ltxtextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the `textcomp` package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
1 <*2ekernel | latexrelease>
2 <latexrelease>\NewModuleRelease{2020/02/02}{ltxtextcomp}
3 <latexrelease>                                {Text Companion symbols}
```

`\oldstylenums`
`\legacyoldstylenums`

Preserve the old definition of `\oldstylenums` under a different name.

This macro implements old style numerals but only works if we assume that the standard math fonts are used. Thus it needs changing in case other math encodings are used.

```
4 \DeclareRobustCommand\legacyoldstylenums[1]{%
5   \begingroup
```

Provide spacing using the interword space of the current font.

```
6   \spaceskip\fontdimen\tw@\font
```

Then switch to the math italic font. We don't change the current value of `\f@series` which means that you can use bold numerals if `\bfseries` is in force. As family we use `\rmdefault` which means that this only works if there exist an OML encoded version of that font or rather a corresponding .fd file (which is the case for standard L^AT_EX fonts even though they only contain substitutions).

```
7   \usefont{OML}{\rmdefault}{\f@series}{it}%
8   \mathgroup\symletters #1%
9   \endgroup
10 }
```

And here is the improved one that adjusts depending on surroundings.

```
11 \DeclareRobustCommand\oldstylenums[1]{%
12   \begingroup
13   \ifmmode
14     \mathgroup\symletters #1%
15   \else
```

The `\CheckEncodingSubset` is discussed below.

```
16   \CheckEncodingSubset\@use@text@encoding{TS1}{\tc@oldstylesubst2{{#1}}}%
17   \fi
18   \endgroup
19 }
```

The helper to select the substitution if needed.

```
20 \def\tc@oldstylesubst#1{%
21   \tc@errorwarn
22     {Oldstyle digits unavailable for
23     family \f@family.\MessageBreak
24     Default oldstyle digits used instead}\@eha
25   \bgroup
26   \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```

27     \ifx\f@family\rmdef@ult
28         \fontfamily\rmsubstdefault
29     \else\ifx\f@family\sfddef@ult
30         \fontfamily\sfsbstdefault
31     \else\ifx\f@family\ttdef@ult
32         \fontfamily\ttsbstdefault
33     \else
34         \fontfamily\textcompsubstdefault
35     \fi\fi\fi
36     \fontencoding{TS1}\selectfont#1%
37 \egroup
38 }

```

(End of definition for \oldstylenums and \legacyoldstylenums.)

`\textcompsubstdefault` Here is the default for the “unknown” case:

```

39 \def\textcompsubstdefault{\rmsubstdefault}

```

(End of definition for \textcompsubstdefault.)

To set up the glyphs for the subsets we need a number helpers.

`\tc@errorwarn` To we produce errors, warnings, or only info in the transcripts if glyphs require substitutions? By default it is “info” only. With the `textcomp` package that can be changed.

```

40 \def\tc@errorwarn#1#2{\@latex@info{#1}}

```

(End of definition for \tc@errorwarn.)

`\tc@subst`

```

41 \def\tc@subst#1{%
42     \tc@errorwarn
43     {Symbol \string#1 not provided by\MessageBreak
44     font family \f@family\space
45     in TS1 encoding.\MessageBreak Default family used instead}\@eha
46 \bgroup
47     \expand@font@defaults
48     \ifx\f@family\rmdef@ult
49         \fontfamily\rmsubstdefault
50     \else\ifx\f@family\sfddef@ult
51         \fontfamily\sfsbstdefault
52     \else\ifx\f@family\ttdef@ult
53         \fontfamily\ttsbstdefault
54     \else
55         \fontfamily\textcompsubstdefault
56     \fi\fi\fi

```

Whatever default was chosen, we claim now (locally hopefully) that it can handle all slots (even if not true) to avoid looping in certain situations, e.g., when something was set up incorrectly.

```

57     \@namedef{TS1:\f@family}{0}%
58     \selectfont#1%
59 \egroup
60 }

```

(End of definition for \tc@subst.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of the command `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce a poor man’s Euro symbol by combining a “C” with a “=”.

```

61 \def\tc@fake@euro#1{%
62   \leavevmode
63   \@font@info{Faking \noexpand#1for font family
64               \f@family\MessageBreak in TS1 encoding}%
65   \valign{##\cr
66     \vfil\hbox to 0.07em{\dimen@\f@size\p@
67                         \math@fontsfalse
68                         \fontsize{.7\dimen@}\z@\selectfont=\hss}%
69     \vfil\cr%
70     \hbox{C}\crrcr
71   }%
72 }
```

(End of definition for \tc@fake@euro.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```
73 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents have been made an error in the `textcomp` package when not available. Now that we provide the functionality in the kernel we avoid the error by swapping in a T1 accent if the TS1 accent is not available.

```

74 %\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
75 \def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent
76                         {TS1}{\tc@swap@accent#1}}
77 \def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}
```

(End of definition for \tc@check@symbol and \tc@check@accent.)

1 Sub-encodings

Here are the default definitions for the TS1 symbols. First those that we assume are always available if a font implements TS1.

```

78 \DeclareTextSymbolDefault{\textdollar}{TS1}
79 \UndeclareTextCommand{\textdollar}{OT1} % don't use the OT1 def any longer
80 \DeclareTextSymbolDefault{\textsterling}{TS1}
81 \UndeclareTextCommand{\textsterling}{OT1}% don't use the OT1 def any longer
82 \DeclareTextSymbolDefault{\textperthousand}{TS1}
83 \UndeclareTextCommand{\textperthousand}{T1} % don't use the T1 def
```

Using `\UndeclareTextCommand` above is enough only if the encoding definition files are not reloaded afterwards. In the past that happened if `fontenc` was used in the document preamble (not any longer). So in some sense it is better to fully remove them from the encoding files, but for rollbacks it is easier to keep them in for now.

These are the standard `itemize` and footnote symbols originally taken from OMS and now from TS1:

```
84 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
85 \DeclareTextSymbolDefault{\textbullet}{TS1}
86 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
87 \DeclareTextSymbolDefault{\textdagger}{TS1}
88 \DeclareTextSymbolDefault{\textparagraph}{TS1}
89 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
90 \DeclareTextSymbolDefault{\textsection}{TS1}
```

And here are the other TS1 glyphs that are implemented by every font (or nearly every)—a few are commented out and moved to sub-encoding 9, because they aren't around in some fonts.

```
91 %%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
92 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
93 %%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
94 \DeclareTextSymbolDefault{\textcent}{TS1}
95 \DeclareTextSymbolDefault{\textcopyright}{TS1}
96 \DeclareTextSymbolDefault{\textdegree}{TS1}
97 \DeclareTextSymbolDefault{\textdiv}{TS1}
98 \DeclareTextSymbolDefault{\textlnot}{TS1}
99 \DeclareTextSymbolDefault{\textonehalf}{TS1}
100 \DeclareTextSymbolDefault{\textonequarter}{TS1}
101 %%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
102 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
103 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
104 \DeclareTextSymbolDefault{\textpm}{TS1}
105 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
106 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
107 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
108 \DeclareTextSymbolDefault{\textregistered}{TS1}
109 %%\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1} % subst in sub-enc 9 above
110 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
111 %%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
112 \DeclareTextSymbolDefault{\texttimes}{TS1}
113 \DeclareTextSymbolDefault{\texttrademark}{TS1}
114 %%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
115 %%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
116 \DeclareTextSymbolDefault{\textyen}{TS1}

117 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
118 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become **unavailable**, i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.

Thus the symbols that are available in sub-encoding x are the symbols above (always available) and the symbols listed as becoming unavailable in sub-encodings $x + 1$ and higher.

1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the OMS glyph.

```
119 \DeclareTextCommandDefault{\textcircled}
120 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
```

1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autost) and higher

```
121 \DeclareTextCommandDefault{\t}
122 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}
```

Capital accents are really only very seldom implemented, so from sub-encoding 2 onwards we use the normal T1 accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```
123 \ifx\Umathcode\@undefined
124 \DeclareTextCommandDefault{\capitalacute}
125 {\tc@check@accent{\'}2\capitalacute}
126 \DeclareTextCommandDefault{\capitalbreve}
127 {\tc@check@accent{\u}2\capitalbreve}
128 \DeclareTextCommandDefault{\capitalcaron}
129 {\tc@check@accent{\v}2\capitalcaron}
130 \DeclareTextCommandDefault{\capitalcedilla}
131 {\tc@check@accent{\c}2\capitalcedilla}
132 \DeclareTextCommandDefault{\capitalcircumflex}
133 {\tc@check@accent{\^}2\capitalcircumflex}
134 \DeclareTextCommandDefault{\capitaldieresis}
135 {\tc@check@accent{\"}2\capitaldieresis}
136 \DeclareTextCommandDefault{\capitaldotaccent}
137 {\tc@check@accent{\.}2\capitaldotaccent}
138 \DeclareTextCommandDefault{\capitalgrave}
139 {\tc@check@accent{\`}2\capitalgrave}
140 \DeclareTextCommandDefault{\capitalhungarumlaut}
141 {\tc@check@accent{\H}2\capitalhungarumlaut}
142 \DeclareTextCommandDefault{\capitalmacron}
143 {\tc@check@accent{\=}2\capitalmacron}
144 \DeclareTextCommandDefault{\capitalogonek}
145 {\tc@check@accent{\k}2\capitalogonek}
146 \DeclareTextCommandDefault{\capitalring}
147 {\tc@check@accent{\r}2\capitalring}
148 \DeclareTextCommandDefault{\capitaltie}
149 {\tc@check@accent{\t}2\capitaltie}
150 \DeclareTextCommandDefault{\capitaltilde}
151 {\tc@check@accent{\~}2\capitaltilde}
```

For `\newtie` and `\capitalnewtie` this is actually wrong, they should pick up the accent from the substitution font (not done yet).

```
152 \DeclareTextCommandDefault{\newtie}
153 {\tc@check@accent{\t}2\newtie}
154 \DeclareTextCommandDefault{\capitalnewtie}
```

```
155 \tc@check@accent{\t}2\capitalnewtie}
```

In Unicode engines we just execute the simple accents:

```
156 \else
157 \DeclareTextCommandDefault\capitalacute{\@tabacckludge'}
158 \DeclareTextCommandDefault\capitalbreve{\u}
159 \DeclareTextCommandDefault\capitalcaron{\v}
160 \DeclareTextCommandDefault\capitalcedilla{\c}
161 \DeclareTextCommandDefault\capitalcircumflex{\^}
162 \DeclareTextCommandDefault\capitaldieresis{"}
163 \DeclareTextCommandDefault\capitaldotaccent{.}
164 \DeclareTextCommandDefault\capitalgrave{\@tabacckludge`}
165 \DeclareTextCommandDefault\capitalhungarumlaut{\H}
166 \DeclareTextCommandDefault\capitalmacron{\@tabacckludge=}
167 \DeclareTextCommandDefault\capitalnewtie{\t}
168 \DeclareTextCommandDefault\capitalogonek{\k}
169 \DeclareTextCommandDefault\capitalring{\r}
170 \DeclareTextCommandDefault\capitaltie{\t}
171 \DeclareTextCommandDefault\capitaltilde{\~}
172 \DeclareTextCommandDefault\newtie{\t}
173 \fi
```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```
174 \DeclareTextCommandDefault{\textlbrackdbl}
175 {\tc@check@symbol2\textlbrackdbl}
176 \DeclareTextCommandDefault{\textrbrackdbl}
177 {\tc@check@symbol2\textrbrackdbl}
```

Old style numerals are again in some fonts but using -OsF, etc. is the better approach to get them, so we claim they aren't in sub-encoding 2 as that's true for most fonts.

```
178 \DeclareTextCommandDefault{\texteightoldstyle}
179 {\tc@check@symbol2\texteightoldstyle}
180 \DeclareTextCommandDefault{\textfiveoldstyle}
181 {\tc@check@symbol2\textfiveoldstyle}
182 \DeclareTextCommandDefault{\textfouroldstyle}
183 {\tc@check@symbol2\textfouroldstyle}
184 \DeclareTextCommandDefault{\textnineoldstyle}
185 {\tc@check@symbol2\textnineoldstyle}
186 \DeclareTextCommandDefault{\textoneoldstyle}
187 {\tc@check@symbol2\textoneoldstyle}
188 \DeclareTextCommandDefault{\textsevenoldstyle}
189 {\tc@check@symbol2\textsevenoldstyle}
190 \DeclareTextCommandDefault{\textsixoldstyle}
191 {\tc@check@symbol2\textsixoldstyle}
192 \DeclareTextCommandDefault{\textthreeoldstyle}
193 {\tc@check@symbol2\textthreeoldstyle}
194 \DeclareTextCommandDefault{\texttwooldstyle}
195 {\tc@check@symbol2\texttwooldstyle}
196 \DeclareTextCommandDefault{\textzerooldstyle}
197 {\tc@check@symbol2\textzerooldstyle}
```

The next set of glyphs is special to T_EX fonts (and available with a few older PS fonts supported through virtual fonts), but not any longer in the majority of fonts provided through autost, so we pretend there aren't available in sub-encoding 2 and below.

```
198 \DeclareTextCommandDefault{\textacutedbl}
```

```

199             {\tc@check@symbol2\textacutedbl}
200 \DeclareTextCommandDefault{\textasciiacute}
201             {\tc@check@symbol2\textasciiacute}
202 \DeclareTextCommandDefault{\textasciibreve}
203             {\tc@check@symbol2\textasciibreve}
204 \DeclareTextCommandDefault{\textasciicaron}
205             {\tc@check@symbol2\textasciicaron}
206 \DeclareTextCommandDefault{\textasciidieresis}
207             {\tc@check@symbol2\textasciidieresis}
208 \DeclareTextCommandDefault{\textasciigrave}
209             {\tc@check@symbol2\textasciigrave}
210 \DeclareTextCommandDefault{\textasciimacron}
211             {\tc@check@symbol2\textasciimacron}
212 \DeclareTextCommandDefault{\textgravedbl}
213             {\tc@check@symbol2\textgravedbl}
214 \DeclareTextCommandDefault{\texttildelow}
215             {\tc@check@symbol2\texttildelow}

```

Finally those below are only available in CM-based fonts but in no font that has its origin outside of the $\text{\TeX}$ world.

```

216 \DeclareTextCommandDefault{\textbaht}
217             {\tc@check@symbol2\textbaht}
218 \DeclareTextCommandDefault{\textbigcircle}
219             {\tc@check@symbol2\textbigcircle}
220 \DeclareTextCommandDefault{\textborn}
221             {\tc@check@symbol2\textborn}
222 \DeclareTextCommandDefault{\textcentoldstyle}
223             {\tc@check@symbol2\textcentoldstyle}
224 \DeclareTextCommandDefault{\textcircledP}
225             {\tc@check@symbol2\textcircledP}
226 \DeclareTextCommandDefault{\textcopleft}
227             {\tc@check@symbol2\textcopleft}
228 \DeclareTextCommandDefault{\textdblhyphenchar}
229             {\tc@check@symbol2\textdblhyphenchar}
230 \DeclareTextCommandDefault{\textdblhyphen}
231             {\tc@check@symbol2\textdblhyphen}
232 \DeclareTextCommandDefault{\textdied}
233             {\tc@check@symbol2\textdied}
234 \DeclareTextCommandDefault{\textdiscount}
235             {\tc@check@symbol2\textdiscount}
236 \DeclareTextCommandDefault{\textdivorced}
237             {\tc@check@symbol2\textdivorced}
238 \DeclareTextCommandDefault{\textdollaroldstyle}
239             {\tc@check@symbol2\textdollaroldstyle}
240 \DeclareTextCommandDefault{\textguarani}
241             {\tc@check@symbol2\textguarani}
242 \DeclareTextCommandDefault{\textleaf}
243             {\tc@check@symbol2\textleaf}
244 \DeclareTextCommandDefault{\textlquill}
245             {\tc@check@symbol2\textlquill}
246 \DeclareTextCommandDefault{\textmarried}
247             {\tc@check@symbol2\textmarried}
248 \DeclareTextCommandDefault{\textmho}
249             {\tc@check@symbol2\textmho}
250 \DeclareTextCommandDefault{\textmusicalnote}

```



```

251             {\tc@check@symbol2\textmusicalnote}
252 \DeclareTextCommandDefault{\textnaira}
253             {\tc@check@symbol2\textnaira}
254 \DeclareTextCommandDefault{\textopenbullet}
255             {\tc@check@symbol2\textopenbullet}
256 \DeclareTextCommandDefault{\textpeso}
257             {\tc@check@symbol2\textpeso}
258 \DeclareTextCommandDefault{\textpildcrow}
259             {\tc@check@symbol2\textpildcrow}
260 \DeclareTextCommandDefault{\textrecipe}
261             {\tc@check@symbol2\textrecipe}
262 \DeclareTextCommandDefault{\textreferencemark}
263             {\tc@check@symbol2\textreferencemark}
264 \DeclareTextCommandDefault{\textrquill}
265             {\tc@check@symbol2\textrquill}
266 \DeclareTextCommandDefault{\textservicemark}
267             {\tc@check@symbol2\textservicemark}
268 \DeclareTextCommandDefault{\textsurd}
269             {\tc@check@symbol2\textsurd}

```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is a definition for `\textpertenthousand` making the symbol up from % and `\char 24` (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```

270 \DeclareTextCommandDefault{\textpertenthousand}
271             {\tc@check@symbol2\textpertenthousand}
272 \UndeclareTextCommand{\textpertenthousand}{T1}

```

1.3 Unavailable in sub-encoding 3 and higher

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```

273 \DeclareTextCommandDefault{\textlangle}
274             {\tc@check@symbol3\textlangle}
275 \DeclareTextCommandDefault{\textrangle}
276             {\tc@check@symbol3\textrangle}

```

1.4 Unavailable in sub-encoding 4 and higher

```

277 \DeclareTextCommandDefault{\textcolonmonetary}
278             {\tc@check@symbol4\textcolonmonetary}
279 \DeclareTextCommandDefault{\textdong}
280             {\tc@check@symbol4\textdong}
281 \DeclareTextCommandDefault{\textdownarrow}
282             {\tc@check@symbol4\textdownarrow}
283 \DeclareTextCommandDefault{\textleftarrow}
284             {\tc@check@symbol4\textleftarrow}
285 \DeclareTextCommandDefault{\textlira}
286             {\tc@check@symbol4\textlira}
287 \DeclareTextCommandDefault{\textrightarrow}

```

```

288             {\tc@check@symbol4\textrightarrow}
289 \DeclareTextCommandDefault{\textuparrow}
290             {\tc@check@symbol4\textuparrow}
291 \DeclareTextCommandDefault{\textwon}
292             {\tc@check@symbol4\textwon}

```

1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher

Most older PS fonts (supported in T_EX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce “tofu” if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```

293 \DeclareTextCommandDefault{\textestimated}
294             {\tc@check@symbol5\textestimated}
295 \DeclareTextCommandDefault{\textnumero}
296             {\tc@check@symbol5\textnumero}

```

1.6 Unavailable in sub-encoding 6 and higher

```

297 \DeclareTextCommandDefault{\textflorin}
298             {\tc@check@symbol6\textflorin}
299 \DeclareTextCommandDefault{\textcurrency}
300             {\tc@check@symbol6\textcurrency}

```

1.7 Unavailable in sub-encoding 7 and higher

```

301 \DeclareTextCommandDefault{\textfractionsolidus}
302             {\tc@check@symbol7\textfractionsolidus}
303 \DeclareTextCommandDefault{\textohm}
304             {\tc@check@symbol7\textohm}
305 \DeclareTextCommandDefault{\textmu}
306             {\tc@check@symbol7\textmu}
307 \DeclareTextCommandDefault{\textminus}
308             {\tc@check@symbol7\textminus}

```

1.8 Unavailable in sub-encoding 8 and higher

```

309 \DeclareTextCommandDefault{\textblank}
310             {\tc@check@symbol{8}\textblank}
311 \DeclareTextCommandDefault{\textinterrobangdown}
312             {\tc@check@symbol{8}\textinterrobangdown}
313 \DeclareTextCommandDefault{\textinterrobang}
314             {\tc@check@symbol{8}\textinterrobang}

```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```

315 \DeclareTextCommandDefault{\texteuro}
316     {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}

```

1.9 Unavailable in Sub-encoding 9 (most missing)

```
317 \DeclareTextCommandDefault{\textcelsius}
318         {\tc@check@symbol{9}\textcelsius}
319 \DeclareTextCommandDefault{\textonesuperior}
320         {\tc@check@symbol{9}\textonesuperior}
321 \DeclareTextCommandDefault{\textthreequartersemdash}
322         {\tc@check@symbol{9}\textthreequartersemdash}
323 \DeclareTextCommandDefault{\textthreesuperior}
324         {\tc@check@symbol{9}\textthreesuperior}
325 \DeclareTextCommandDefault{\texttwelveudash}
326         {\tc@check@symbol{9}\texttwelveudash}
327 \DeclareTextCommandDefault{\texttwosuperior}
328         {\tc@check@symbol{9}\texttwosuperior}
329 \DeclareTextCommandDefault{\textbardbl}
330         {\tc@check@symbol{9}\textbardbl}
```

2 Unicode engine specials

If we are using a unicode engine we handle some glyphs differently, so this here are the definitions for the Unicode encoding (overwriting the defaults above).

```
331 \ifx \Umathcode\@undefined \else
```

This set should be taken from TS1 encoding even if it means you get it from the default font for that encoding.

```
332 %\DeclareTextSymbol{\textcopyright}{TS1}{171}
333 %\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
334 %\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
335 %\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
336 %\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
337 %\DeclareTextSymbol{\textleaf}{TS1}{108}
338 %\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
339 %\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

If oldstyle numerals are asked for we just use \oldstylenums.

```
340 \DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
341 \DeclareTextCommand{\textoneoldstyle} \UnicodeEncodingName{\oldstylenums{1}}
342 \DeclareTextCommand{\texttwooldstyle} \UnicodeEncodingName{\oldstylenums{2}}
343 \DeclareTextCommand{\textthreeoldstyle} \UnicodeEncodingName{\oldstylenums{3}}
344 \DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
345 \DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
346 \DeclareTextCommand{\textsixoldstyle} \UnicodeEncodingName{\oldstylenums{6}}
347 \DeclareTextCommand{\textsevenoldstyle} \UnicodeEncodingName{\oldstylenums{7}}
348 \DeclareTextCommand{\texteightoldstyle} \UnicodeEncodingName{\oldstylenums{8}}
349 \DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}
```

These have Unicode slots so this should be integrated into TU explicitly

```
350 \DeclareTextSymbol{\textpilcrow} \UnicodeEncodingName{"00B6}
351 \DeclareTextSymbol{\textborn} \UnicodeEncodingName{"002A}
352 \DeclareTextSymbol{\textdied} \UnicodeEncodingName{"2020}
353 \DeclareTextSymbol{\textlbrackdbl} \UnicodeEncodingName{"27E6}
354 \DeclareTextSymbol{\textrbrackdbl} \UnicodeEncodingName{"27E7}
355 \DeclareTextSymbol{\textguarani} \UnicodeEncodingName{"20B2}
```

We could make `\textcentoldstyle` and `\textdollaroldstyle` point to dollar and cent in the Unicode encoding

```
356 %\DeclareTextSymbol{\textcentoldstyle} \UnicodeEncodingName{"00A2}
357 %\DeclareTextSymbol{\textdollaroldstyle}\UnicodeEncodingName{"0024}
```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```
358 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
359 \DeclareTextSymbol{\textcentoldstyle} {TS1}{139}

360 \fi % --- END of Unicode engines specials
```

3 Font family sub-encodings setup

We declare the subsets for a good number of fonts in the kernel ...

But first the default for anything that is not declared. We use 9 which is most likely much too conservative, but with the advantage that we aren't getting missing glyphs (or at least that this is very unlikely). For nearly all font in the T_EX Live distribution of 2019 "correct" classifications are given below, so that this default is only used for new font families, and over time the right classifications can be added here too.

```
361 \DeclareEncodingSubset{TS1}{?}{9}
```

This first block contains the fonts that have been already supported by the `textcomp` package way back, i.e., the font families that have T_EX support since the mid-nineties.

```
362 \DeclareEncodingSubset{TS1}{ccr} {0}
363 \DeclareEncodingSubset{TS1}{cmbr} {0}
```

The following 4 declarations are now part of the corresponding `.fd` file, hopefully other will follow so that this list of declarations can eventually be removed from the kernel (where it doesn't belong).

```
364 %\DeclareEncodingSubset{TS1}{cmr} {0}
365 %\DeclareEncodingSubset{TS1}{cmss} {0}
366 %\DeclareEncodingSubset{TS1}{cmtt} {0}
367 %\DeclareEncodingSubset{TS1}{cmvtt} {0}

368 \DeclareEncodingSubset{TS1}{cmtl} {0}
369 \DeclareEncodingSubset{TS1}{pxr} {0}
370 \DeclareEncodingSubset{TS1}{pxss} {0}
371 \DeclareEncodingSubset{TS1}{pxtt} {0}
372 \DeclareEncodingSubset{TS1}{qag} {0}
373 \DeclareEncodingSubset{TS1}{qbk} {0}
374 \DeclareEncodingSubset{TS1}{qcr} {0}
375 \DeclareEncodingSubset{TS1}{qcs} {0}
376 \DeclareEncodingSubset{TS1}{qhvc} {0}
377 \DeclareEncodingSubset{TS1}{qhv} {0}
378 \DeclareEncodingSubset{TS1}{qpl} {0}
379 \DeclareEncodingSubset{TS1}{qtm} {0}
380 \DeclareEncodingSubset{TS1}{qzc} {0}
381 \DeclareEncodingSubset{TS1}{txr} {0}
382 \DeclareEncodingSubset{TS1}{txss} {0}
383 \DeclareEncodingSubset{TS1}{txtt} {0}

384 \DeclareEncodingSubset{TS1}{lmr} {1}
385 \DeclareEncodingSubset{TS1}{lmdh} {1}
386 \DeclareEncodingSubset{TS1}{lmss} {1}
387 \DeclareEncodingSubset{TS1}{lmssq} {1}
```

```
388 \DeclareEncodingSubset{TS1}{lmttt} {1}
```

The lmtt family is missing TM, SM, and perthousand for some reason, so the first safe sub-encoding would be 2, but that is then missing out a huge number of glyphs that are available, so we claim it is sub-encoding 1 even if this can lead to missing glyphs.

```
389 \DeclareEncodingSubset{TS1}{lmtt} {1} % missing TM, SM and pertenthousand
```

The next three families have been removed from TeX Live, but we keep the definitions

```
390 \DeclareEncodingSubset{TS1}{ptmx} {2}
```

```
391 \DeclareEncodingSubset{TS1}{ptmj} {2}
```

```
392 \DeclareEncodingSubset{TS1}{ul8} {2}
```

The next set are the early PostScript font implementations, these days there are better alternatives, but Note that, their virtual fonts contain a lot of “tofu” in form of black squares, thus they don’t even give a missing character warning if you select such a glyph. This is why they are set as sub-encoding 5.

```
393 \DeclareEncodingSubset{TS1}{bch} {5} % tofu for blank, ohm
```

```
394 \DeclareEncodingSubset{TS1}{futj} {5} % tofu for blank, interrobang/down, ohm
```

```
395 \DeclareEncodingSubset{TS1}{futs} {5} % tofu for blank, ohm
```

```
396 \DeclareEncodingSubset{TS1}{futex} {5} % probably (currently broken distrib)
```

```
397 \DeclareEncodingSubset{TS1}{pag} {5} % tofu for blank, interrobang/down, ohm
```

```
398 \DeclareEncodingSubset{TS1}{pbk} {5} % tofu for blank, interrobang/down, ohm
```

```
399 \DeclareEncodingSubset{TS1}{pcr} {5} % tofu for blank, interrobang/down, ohm
```

```
400 \DeclareEncodingSubset{TS1}{phv} {5} % tofu for blank, interrobang/down, ohm
```

```
401 \DeclareEncodingSubset{TS1}{pnc} {5} % tofu for blank, interrobang/down, ohm
```

```
402 \DeclareEncodingSubset{TS1}{pplj} {5} % tofu for blank
```

```
403 \DeclareEncodingSubset{TS1}{pplx} {5} % tofu for blank
```

```
404 \DeclareEncodingSubset{TS1}{ppl} {5} % tofu for blank interrobang/down
```

```
405 \DeclareEncodingSubset{TS1}{ptm} {5} % tofu for blank, interrobang/down, ohm
```

```
406 \DeclareEncodingSubset{TS1}{pzc} {5} % tofu for blank, interrobang/down, ohm
```

```
407 \DeclareEncodingSubset{TS1}{ul9} {5} % tofu for blank, interrobang/down, ohm
```

The next set suffers from the same problem and they contain even fewer real glyphs.

```
408 \DeclareEncodingSubset{TS1}{dayroms} {6} % tofu for blank, interrobang/down, ohm
```

```
409 \DeclareEncodingSubset{TS1}{dayrom} {6} % tofu for blank, interrobang/down, ohm
```

```
410 \DeclareEncodingSubset{TS1}{augie} {8} % really only missing euro
```

```
411 \DeclareEncodingSubset{TS1}{put} {8}
```

```
412 \DeclareEncodingSubset{TS1}{uag} {8} % probably (currently broken distrib)
```

```
413 \DeclareEncodingSubset{TS1}{ugq} {8}
```

```
414 \DeclareEncodingSubset{TS1}{zi4} {9}
```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts ...

```
415 \DeclareEncodingSubset{TS1}{hls} {5}
```

```
416 \DeclareEncodingSubset{TS1}{hlst} {5}
```

```
417 \DeclareEncodingSubset{TS1}{hlct} {5}
```

```
418 \DeclareEncodingSubset{TS1}{hlh} {5}
```

```
419 \DeclareEncodingSubset{TS1}{hlx} {8}
```

```
420 \DeclareEncodingSubset{TS1}{hlce} {8}
```

```
421 \DeclareEncodingSubset{TS1}{hlcn} {8}
```

```
422 \DeclareEncodingSubset{TS1}{hlcw} {8}
```

```
423 \DeclareEncodingSubset{TS1}{hlcf} {8}
```

Below are the newer fonts that have support files for L^AT_EX. With very few exceptions the classifications are done so that all characters are correctly produced (either being available in the font or substituted).

There are a few fonts that contain “tofu” squares in places (instead of a real glyph) and in a few cases some really seldom needed chars are unavailable, i.e., produce missing glyphs (to avoid that a large number of available chars are unnecessarily substituted).

Encoding declarations for these font families shouldn’t really be in the kernel, but part of the .fd files for the family. When we introduced the concept in 2021 we had some hope that this would happen over time and that we could take the declarations out—after all it is nearly impossible to maintain it correctly in the kernel, given that fonts may get new glyphs added (happened for several of them in the recent year) which is something we wouldn’t notice. However, so far this hasn’t happened, so in 2024, I went through the current set and adjusted the declarations in several places.

Next four are wrong and still need adjustment:

```

424 \DeclareEncodingSubset{TS1}{lato-*}      {0} % with a bunch of tofu inside
425 \DeclareEncodingSubset{TS1}{opensans-*}  {0} % with a bunch of tofu inside
426 \DeclareEncodingSubset{TS1}{cantarell-*} {0} % with a bunch of tofu inside
427 \DeclareEncodingSubset{TS1}{tli}         {1} % with lots of tofu inside
428 \DeclareEncodingSubset{TS1}{fbb-*}       {2} % missing centoldstyle

429 \DeclareEncodingSubset{TS1}{Alegreya-*}   {2}
430 \DeclareEncodingSubset{TS1}{AlegreyaSans-*} {2}
431 \DeclareEncodingSubset{TS1}{BaskervilleF-*} {2}
432 \DeclareEncodingSubset{TS1}{DejaVuSans-TLF} {2}
433 \DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF} {2}

```

Next one is missing \textfractionsolidus but is otherwise completely sub-encoding 2 so we use that sub-encoding.

```

434 \DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF} {2}

435 \DeclareEncodingSubset{TS1}{EBGaramond-*} {2}
436 \DeclareEncodingSubset{TS1}{Merriwthr-OsF} {2}
437 \DeclareEncodingSubset{TS1}{MerriwthrSans-OsF} {2}
438 \DeclareEncodingSubset{TS1}{Montserrat-*} {2}
439 \DeclareEncodingSubset{TS1}{MontserratAlternates-*} {2}
440 \DeclareEncodingSubset{TS1}{NotoSansMono-TLF} {2}
441 \DeclareEncodingSubset{TS1}{NotoSansMono-T0sF} {2}
442 \DeclareEncodingSubset{TS1}{Tempora-TLF} {2}
443 \DeclareEncodingSubset{TS1}{Tempora-T0sF} {2}
444 \DeclareEncodingSubset{TS1}{XCharter-TLF} {2}
445 \DeclareEncodingSubset{TS1}{XCharter-T0sF} {2}
446 \DeclareEncodingSubset{TS1}{erewhon-*} {2}

447 \DeclareEncodingSubset{TS1}{Arimo-TLF} {3}
448 \DeclareEncodingSubset{TS1}{Crlt-*} {3}
449 \DeclareEncodingSubset{TS1}{IBMPlexMono-TLF} {3}
450 \DeclareEncodingSubset{TS1}{IBMPlexSans-TLF} {3}
451 \DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF} {3}
452 \DeclareEncodingSubset{TS1}{SourceCodePro-TLF} {3}
453 \DeclareEncodingSubset{TS1}{SourceCodePro-T0sF} {3}
454 \DeclareEncodingSubset{TS1}{SourceSansPro-*} {3}
455 \DeclareEncodingSubset{TS1}{SourceSerifPro-*} {3}
456 \DeclareEncodingSubset{TS1}{Tinos-TLF} {3}

```

```

457 \DeclareEncodingSubset{TS1}{AccanthisADFStdNoThree-LF}{4}
458 \DeclareEncodingSubset{TS1}{Cabin-TLF}{4}
459 \DeclareEncodingSubset{TS1}{Caladea-TLF}{4}
460 \DeclareEncodingSubset{TS1}{Chivo-*}{4}
461 \DeclareEncodingSubset{TS1}{ClearSans-TLF}{4}
462 \DeclareEncodingSubset{TS1}{Coelacanth-LF}{4}
463 \DeclareEncodingSubset{TS1}{CrimsonPro-*}{4}
464 \DeclareEncodingSubset{TS1}{FiraMono-TLF}{4}
465 \DeclareEncodingSubset{TS1}{FiraMono-TOfF}{4}
466 \DeclareEncodingSubset{TS1}{FiraSans-*}{4}
467 \DeclareEncodingSubset{TS1}{Go-TLF}{4}
468 \DeclareEncodingSubset{TS1}{GoMono-TLF}{4}
469 \DeclareEncodingSubset{TS1}{InriaSans-*}{4}
470 \DeclareEncodingSubset{TS1}{InriaSerif-*}{4}
471 \DeclareEncodingSubset{TS1}{LibertinusSans-*}{4}
472 \DeclareEncodingSubset{TS1}{LibertinusSerif-*}{4}
473 \DeclareEncodingSubset{TS1}{LibreBodoni-TLF}{4}
474 \DeclareEncodingSubset{TS1}{LibreFranklin-TLF}{4}
475 \DeclareEncodingSubset{TS1}{LinguisticsPro-LF}{4}
476 \DeclareEncodingSubset{TS1}{LinguisticsPro-OfF}{4}
477 \DeclareEncodingSubset{TS1}{LinuxBioluminumT-*}{4}
478 \DeclareEncodingSubset{TS1}{LinuxLibertineT-*}{4}
479 \DeclareEncodingSubset{TS1}{MintSpirit-*}{4}
480 \DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*}{4}
481 \DeclareEncodingSubset{TS1}{PTMono-TLF}{4}
482 \DeclareEncodingSubset{TS1}{PTSans-TLF}{4}
483 \DeclareEncodingSubset{TS1}{PTSansCaption-TLF}{4}
484 \DeclareEncodingSubset{TS1}{PTSansNarrow-TLF}{4}
485 \DeclareEncodingSubset{TS1}{PTSerif-TLF}{4}
486 \DeclareEncodingSubset{TS1}{PTSerifCaption-TLF}{4}
487 \DeclareEncodingSubset{TS1}{Raleway-TLF}{4}
488 \DeclareEncodingSubset{TS1}{Raleway-TOfF}{4}
489 \DeclareEncodingSubset{TS1}{Roboto-*}{4}
490 \DeclareEncodingSubset{TS1}{RobotoMono-TLF}{4}
491 \DeclareEncodingSubset{TS1}{RobotoSlab-TLF}{4}
492 \DeclareEncodingSubset{TS1}{Rosario-*}{4}
493 \DeclareEncodingSubset{TS1}{SticksTooText-*}{4}
494 \DeclareEncodingSubset{TS1}{UniversalisADFStd-LF}{4}

495 \DeclareEncodingSubset{TS1}{Almndr-OfF}{5}
496 \DeclareEncodingSubset{TS1}{Baskervalex-*}{5}
497 \DeclareEncodingSubset{TS1}{Bttr-TLF}{5}
498 \DeclareEncodingSubset{TS1}{Cinzel-LF}{5}
499 \DeclareEncodingSubset{TS1}{CinzelDecorative-LF}{5}
500 \DeclareEncodingSubset{TS1}{Cochineal-*}{5}
501 \DeclareEncodingSubset{TS1}{DejaVuSerif-TLF}{5}
502 \DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF}{5}
503 \DeclareEncodingSubset{TS1}{GilliusADF-LF}{5}
504 \DeclareEncodingSubset{TS1}{GilliusADFCond-LF}{5}
505 \DeclareEncodingSubset{TS1}{GilliusADFNoTwo-LF}{5}
506 \DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-LF}{5}
507 \DeclareEncodingSubset{TS1}{OldStandard-TLF}{5}
508 \DeclareEncodingSubset{TS1}{PlyfrDisplay-TLF}{5}
509 \DeclareEncodingSubset{TS1}{PlyfrDisplay-TOfF}{5}
510 \DeclareEncodingSubset{TS1}{TheanoDidot-TLF}{5}

```

```

511 \DeclareEncodingSubset{TS1}{TheanoDidot-T0sF} {5}
512 \DeclareEncodingSubset{TS1}{TheanoModern-TLF} {5}
513 \DeclareEncodingSubset{TS1}{TheanoModern-T0sF} {5}
514 \DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF} {5}
515 \DeclareEncodingSubset{TS1}{TheanoOldStyle-T0sF} {5}
516 \DeclareEncodingSubset{TS1}{charssil-TLF} {5}

517 \DeclareEncodingSubset{TS1}{Crimson-TLF} {6}
518 \DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF} {6}
519 \DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
520 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF} {6}
521 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF} {6}
522 \DeclareEncodingSubset{TS1}{Ovrlck-LF} {6}

523 \DeclareEncodingSubset{TS1}{ComicNeue-TLF} {7}
524 \DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF} {7}
525 \DeclareEncodingSubset{TS1}{CormorantGaramond-*} {7}
526 \DeclareEncodingSubset{TS1}{Heuristica-TLF} {7}
527 \DeclareEncodingSubset{TS1}{Heuristica-T0sF} {7}
528 \DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF} {7}
529 \DeclareEncodingSubset{TS1}{LibreBskrvl-TLF} {7}
530 \DeclareEncodingSubset{TS1}{LibreCsln-*} {7}
531 \DeclareEncodingSubset{TS1}{Lbstr-LF} {7}
532 \DeclareEncodingSubset{TS1}{Mrcls-LF} {7}

```

Strangely enough NotoSerif and NotoSans are sub-encoding 7 as they are missing `\textminus` and several other glyphs. In contrast, the NotoSansMono is far more complete.

```

533 \DeclareEncodingSubset{TS1}{NotoSans-*} {7}
534 \DeclareEncodingSubset{TS1}{NotoSerif-*} {7}
535 \DeclareEncodingSubset{TS1}{Quattro-LF} {7}
536 \DeclareEncodingSubset{TS1}{QuattroSans-LF} {7}
537 \DeclareEncodingSubset{TS1}{Frm-LF} {7} % the superiors are missing

538 \DeclareEncodingSubset{TS1}{LibertinusMono-TLF} {8}
539 \DeclareEncodingSubset{TS1}{AlgolRevived-TLF} {9}

```

4 Legacy symbol support for lists and footnote symbols

`\UseLegacyTextSymbols`

```

540 \def\UseLegacyTextSymbols{%
541   \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
542   \DeclareTextSymbolDefault{\textbardbl}{OMS}%
543   \DeclareTextSymbolDefault{\textbullet}{OMS}%
544   \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
545   \DeclareTextSymbolDefault{\textdagger}{OMS}%
546   \DeclareTextSymbolDefault{\textparagraph}{OMS}%
547   \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
548   \DeclareTextSymbolDefault{\textsection}{OMS}%
549   \UndeclareTextCommand{\textsection}{T1}%
550   \expandafter\let\csname oldstylenums \expandafter\endcsname
551     \csname legacyoldstylenums \endcsname
552 }

```


(End of definition for \UseLegacyTextSymbols.)

\textlegacyasteriskcentered Here are new names for the legacy symbols that L^AT_EX used to pick up from the OMS
\textlegacybardbl encoded fonts (and used for itemize lists or footnote symbols).

We go the roundabout way via separate OMS declarations so that

\renewcommand\textbullet{\textlegacybullet}

doesn't produce an endless loop.

```

553 \DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3} % "03
554 \DeclareTextSymbol{\textlegacybardbl}{OMS}{107} % "6B
555 \DeclareTextSymbol{\textlegacybullet}{OMS}{15} % "0F
556 \DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122} % "7A
557 \DeclareTextSymbol{\textlegacydagger}{OMS}{121} % "79
558 \DeclareTextSymbol{\textlegacyparagraph}{OMS}{123} % "7B
559 \DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1} % "01
560 \DeclareTextSymbol{\textlegacysection}{OMS}{120} % "78

561 \DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
562 \DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
563 \DeclareTextSymbolDefault{\textlegacybullet}{OMS}
564 \DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
565 \DeclareTextSymbolDefault{\textlegacydagger}{OMS}
566 \DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
567 \DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
568 \DeclareTextSymbolDefault{\textlegacysection}{OMS}

```

(End of definition for \textlegacyasteriskcentered and others.)

Supporting rollback ...

```

569 </2ekernel | latexrelease>
570 <latexrelease>
571 <latexrelease>\IncludeInRelease{0000/00/00}%
572 <latexrelease>    {lttextcomp}{\Undefine text companion symbols}%
573 <latexrelease>
574 <latexrelease>\DeclareRobustCommand\oldstylenums[1]{%
575 <latexrelease>    \begingroup
576 <latexrelease>    \spaceskip\fontdimen\tw@font
577 <latexrelease>    \usefont{OML}{\rmdefault}{\f@series}{it}%
578 <latexrelease>    \mathgroup\symletters #1%
579 <latexrelease>    \endgroup
580 <latexrelease>}
581 <latexrelease>\let\legacyoldstylenums\@undefined
582 <latexrelease>\def\textcompsubstdefault{cmr}
583 <latexrelease>
584 <latexrelease>\let\DeclareEncodingSubset\@undefined
585 <latexrelease>\let\CheckEncodingSubset\@undefined
586 <latexrelease>
587 <latexrelease>\DeclareTextSymbolDefault{\textdollar}{OT1}
588 <latexrelease>\DeclareTextSymbolDefault{\textsterling}{OT1}
589 <latexrelease>\DeclareTextCommand{\textdollar}{OT1}{\hmode\bgroup
590 <latexrelease>    \ifdim \fontdimen\@ne\font >\z@
591 <latexrelease>        \slshape
592 <latexrelease>    \else
593 <latexrelease>        \upshape
594 <latexrelease>    \fi

```

```

595 <latexrelease> \char'{$\egroup}
596 <latexrelease>\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
597 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
598 <latexrelease> \itshape
599 <latexrelease> \else
600 <latexrelease> \fontshape{ui}\selectfont
601 <latexrelease> \fi
602 <latexrelease> \char'{$\egroup}
603 <latexrelease>\DeclareTextCommand{\textperthousand}{T1}
604 <latexrelease> {\%\char 24 }
605 <latexrelease>
606 <latexrelease>\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
607 <latexrelease>\DeclareTextSymbolDefault{\textbullet}{OMS}
608 <latexrelease>\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
609 <latexrelease>\DeclareTextSymbolDefault{\textdagger}{OMS}
610 <latexrelease>\DeclareTextSymbolDefault{\textparagraph}{OMS}
611 <latexrelease>\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
612 <latexrelease>\DeclareTextSymbolDefault{\textsection}{OMS}
613 <latexrelease>
614 <latexrelease>\DeclareTextSymbolDefault{\textbardbl}{OMS}
615 <latexrelease>\let\textbrokenbar\@undefined
616 <latexrelease>\let\textcelsius\@undefined
617 <latexrelease>\let\textcent\@undefined
618 <latexrelease>\DeclareTextCommandDefault{\textcopyright}
619 <latexrelease> {\textcircled{c}}
620 <latexrelease>\let\textdegree\@undefined
621 <latexrelease>\let\textdiv\@undefined
622 <latexrelease>\let\textlnot\@undefined
623 <latexrelease>\let\textonehalf\@undefined
624 <latexrelease>\let\textonequarter\@undefined
625 <latexrelease>\let\textonesuperior\@undefined
626 <latexrelease>\DeclareTextCommandDefault{\textordfeminine}
627 <latexrelease> {\textsuperscript{a}}
628 <latexrelease>\DeclareTextCommandDefault{\textordmasculine}
629 <latexrelease> {\textsuperscript{o}}
630 <latexrelease>\let\textpm\@undefined
631 <latexrelease>\let\textquotesingle\@undefined
632 <latexrelease>\let\textquotestraightbase\@undefined
633 <latexrelease>\let\textquotestraightdblbase\@undefined
634 <latexrelease>\DeclareTextCommandDefault{\textregistered}
635 <latexrelease> {\textcircled{%
636 <latexrelease> \check@mathfonts\fontsize\sf@size\z@
637 <latexrelease> \math@fontsfalse\selectfont R}}
638 <latexrelease>\let\textthreequartersemdash\@undefined
639 <latexrelease>\let\textthreequarters\@undefined
640 <latexrelease>\let\textthreesuperior\@undefined
641 <latexrelease>\let\texttimes\@undefined
642 <latexrelease>\DeclareTextCommandDefault{\texttrademark}
643 <latexrelease> {\textsuperscript{TM}}
644 <latexrelease>\let\texttwelveudash\@undefined
645 <latexrelease>\let\texttwosuperior\@undefined
646 <latexrelease>\let\textyen\@undefined
647 <latexrelease>
648 <latexrelease>\let\textcapitalcompwordmark\@undefined

```

```

649 <latexrelease>\let\textascendercompwordmark\@undefined
650 <latexrelease>
651 <latexrelease>\DeclareTextAccentDefault{\textcircled}{OMS}
652 <latexrelease>\DeclareTextAccentDefault{\t}{OML}
653 <latexrelease>
654 <latexrelease>\let\capitalacute\@undefined
655 <latexrelease>\let\capitalbreve\@undefined
656 <latexrelease>\let\capitalcaron\@undefined
657 <latexrelease>\let\capitalcedilla\@undefined
658 <latexrelease>\let\capitalcircumflex\@undefined
659 <latexrelease>\let\capitaldieresis\@undefined
660 <latexrelease>\let\capitaldotaccent\@undefined
661 <latexrelease>\let\capitalgrave\@undefined
662 <latexrelease>\let\capitalhungarumlaut\@undefined
663 <latexrelease>\let\capitalmacron\@undefined
664 <latexrelease>\let\capitalnewtie\@undefined
665 <latexrelease>\let\capitalogonek\@undefined
666 <latexrelease>\let\capitalring\@undefined
667 <latexrelease>\let\capitaltie\@undefined
668 <latexrelease>\let\capitaltilde\@undefined
669 <latexrelease>\let\newtie\@undefined
670 <latexrelease>
671 <latexrelease>\let\textlbrackdbl\@undefined
672 <latexrelease>\let\textrrackdbl\@undefined
673 <latexrelease>
674 <latexrelease>\let\texteightoldstyle\@undefined
675 <latexrelease>\let\textfiveoldstyle\@undefined
676 <latexrelease>\let\textfouroldstyle\@undefined
677 <latexrelease>\let\textnineoldstyle\@undefined
678 <latexrelease>\let\textoneoldstyle\@undefined
679 <latexrelease>\let\textsevenoldstyle\@undefined
680 <latexrelease>\let\textsixoldstyle\@undefined
681 <latexrelease>\let\textthreeoldstyle\@undefined
682 <latexrelease>\let\texttwooldstyle\@undefined
683 <latexrelease>\let\textzerooldstyle\@undefined
684 <latexrelease>
685 <latexrelease>\let\textacutedbl\@undefined
686 <latexrelease>\let\textasciicute\@undefined
687 <latexrelease>\let\textasciibreve\@undefined
688 <latexrelease>\let\textasciicaron\@undefined
689 <latexrelease>\let\textasciidieresis\@undefined
690 <latexrelease>\let\textasciigrave\@undefined
691 <latexrelease>\let\textasciimacron\@undefined
692 <latexrelease>\let\textgravedbl\@undefined
693 <latexrelease>\let\texttildelow\@undefined
694 <latexrelease>
695 <latexrelease>\let\textbaht\@undefined
696 <latexrelease>\let\textbigcircle\@undefined
697 <latexrelease>\let\textborn\@undefined
698 <latexrelease>\let\textcentoldstyle\@undefined
699 <latexrelease>\let\textcircledP\@undefined
700 <latexrelease>\let\textcopyleft\@undefined
701 <latexrelease>\let\textdblhyphenchar\@undefined
702 <latexrelease>\let\textdblhyphen\@undefined

```

```

703 <latexrelease>\let\textdied\@undefined
704 <latexrelease>\let\textdiscount\@undefined
705 <latexrelease>\let\textdivorced\@undefined
706 <latexrelease>\let\textdollaroldstyle\@undefined
707 <latexrelease>\let\textguarani\@undefined
708 <latexrelease>\let\textleaf\@undefined
709 <latexrelease>\let\textlquill\@undefined
710 <latexrelease>\let\textmarried\@undefined
711 <latexrelease>\let\textmho\@undefined
712 <latexrelease>\let\textmusicalnote\@undefined
713 <latexrelease>\let\textnaira\@undefined
714 <latexrelease>\let\textopenbullet\@undefined
715 <latexrelease>\let\textpeso\@undefined
716 <latexrelease>\let\textpilcrow\@undefined
717 <latexrelease>\let\textrecipe\@undefined
718 <latexrelease>\let\textreferencemark\@undefined
719 <latexrelease>\let\textarquill\@undefined
720 <latexrelease>\let\textservicemark\@undefined
721 <latexrelease>\let\textsurd\@undefined
722 <latexrelease>
723 <latexrelease>\DeclareTextCommand{\textpertenthousand}{T1}
724 <latexrelease>          {\%\char 24\char 24 }
725 <latexrelease>
726 <latexrelease>\let\textlangle\@undefined
727 <latexrelease>\let\extrangle\@undefined
728 <latexrelease>
729 <latexrelease>\let\textcolonmonetary\@undefined
730 <latexrelease>\let\textdong\@undefined
731 <latexrelease>\let\textdownarrow\@undefined
732 <latexrelease>\let\textleftarrow\@undefined
733 <latexrelease>\let\textlira\@undefined
734 <latexrelease>\let\textrightarrow\@undefined
735 <latexrelease>\let\textuparrow\@undefined
736 <latexrelease>\let\textwon\@undefined
737 <latexrelease>
738 <latexrelease>\let\textestimated\@undefined
739 <latexrelease>\let\textnumero\@undefined
740 <latexrelease>
741 <latexrelease>\let\textflorin\@undefined
742 <latexrelease>\let\textcurrency\@undefined
743 <latexrelease>
744 <latexrelease>\let\textfractionsolidus\@undefined
745 <latexrelease>\let\textohm\@undefined
746 <latexrelease>\let\textmu\@undefined
747 <latexrelease>\let\textminus\@undefined
748 <latexrelease>
749 <latexrelease>\let\textblank\@undefined
750 <latexrelease>\let\textinterrobangdown\@undefined
751 <latexrelease>\let\textinterrobang\@undefined
752 <latexrelease>
753 <latexrelease>\let\texteuro\@undefined
754 <latexrelease>
755 <latexrelease>\let\textcelsius\@undefined
756 <latexrelease>\let\tonesuperior\@undefined

```

```

757 <latexrelease>\let\textthreequartersemdash\@undefined
758 <latexrelease>\let\textthreesuperior\@undefined
759 <latexrelease>\let\texttwelveudash\@undefined
760 <latexrelease>\let\texttwosuperior\@undefined
761 <latexrelease>\let\textbardbl\@undefined
762 <latexrelease>
763 <latexrelease>\let\UseLegacyTextSymbols\@undefined
764 <latexrelease>\let\textlegacyasteriskcentered\@undefined
765 <latexrelease>\let\textlegacybardbl\@undefined
766 <latexrelease>\let\textlegacybullet\@undefined
767 <latexrelease>\let\textlegacydaggerdbl\@undefined
768 <latexrelease>\let\textlegacydagger\@undefined
769 <latexrelease>\let\textlegacyparagraph\@undefined
770 <latexrelease>\let\textlegacyperiodcentered\@undefined
771 <latexrelease>\let\textlegacysection\@undefined
772 <latexrelease>
773 <latexrelease>\EndModuleRelease

```

5 The textcomp package

For any rollback request before 2018-08-11 we make an attempt by loading the 2018 version.

```

774 <*TS1sty>
775 \DeclareRelease{}{1997-12-01}{textcomp-2018-08-11.sty}
776 \DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}
777 \DeclareCurrentRelease{}{2020-02-02}
778
779 \ProvidesPackage{textcomp}
780 [2024/04/24 v2.1b Standard LaTeX package]

```

A precaution in case this is used without rebuilding the format.

```

781 \NeedsTeXFormat{LaTeX2e}[2020/02/02]

```

This is implemented by defining the default subset:

```

782 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
783 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
784 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}
785 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{9}}

```

The default is set up in the kernel is “safe” these days for unknown fonts but LaTeX has definitions for most families so it seldom applies.

If a different default is used then one needs to check the results to ensure that there aren’t “missing glyphs”.

The next set of options define the warning level (default in the kernel is info only). Using the package options you can change this behavior.

```

786 \DeclareOption{error}
787 {\gdef\tc@errorwarn{\PackageError{textcomp}}}
788 \DeclareOption{warn}
789 {\gdef\tc@errorwarn#1#2{\PackageWarning{textcomp}{#1}}}
790 \DeclareOption{info}
791 {\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
792 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2{}}

```

The “force” option basically changes the sub-encoding to that of the default (which, unless changes, is 9 these days), i.e., it no longer depends on the font in use. This is mainly there because it might have been used in older documents, but not something that is recommended.

```

793 \DeclareOption{force}{%
794   \def\CheckEncodingSubset#1#2#3#4#5{%
795     \ifnum #4>%
796       0\csname #2:\endcsname
797       \relax
798     \expandafter\@firstoftwo
799   \else
800     \expandafter\@secondoftwo
801   \fi
802   {#1{#2}}{#3}%
803   #5}%
804 }

805 \ProcessOptions\relax

```

There is not much else to do nowadays, because everything is already set up in the L^AT_EX kernel.

```

806 \InputIfFileExists{textcomp.cfg}
807 {\PackageInfo{textcomp}{Local configuration file used}}{}
808 </TS1sty>

```

5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```

809 <*TS1oldsty>
810 \ProvidesPackage{textcomp}
811 [2018/08/11 v2.0j Standard LaTeX package]

```

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

#5 those TS1 symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-T_EX world provide only this subset.

#4 = #5 + `\texteuro`. Most newer fonts provide this.

#3 = #4 + `\textomega`. Can also be described as $TS1 \cap (ISO-Adobe \cup MacRoman)$. (Except for the missing “currency”.)

#2 = **#3** + `\textestimated` + `\textcurrency`. Can also be described as `TS1` $\cap$ Adobe-Western-2. This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

#1 = `TS1` without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full `TS1`

And here a summary to go in the transcript file:

```

812 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
813   \space\space 5 = only ISO-Adobe without
814                     \string\textcurrency\MessageBreak
815   \space\space 4 = 5 + \string\texteuro\MessageBreak
816   \space\space 3 = 4 + \string\textohm\MessageBreak
817   \space\space 2 = 3 + \noexpand\textestimated+
818                     \string\textcurrency\MessageBreak
819   \space\space 1 = TS1 - \noexpand\textcircled-
820                     \string\t\MessageBreak
821   \space\space 0 = TS1 (full)\MessageBreak
822   Font families with sub-encoding setting implement\MessageBreak
823   only a restricted character set as indicated.\MessageBreak
824   Family '?' is the default used for unknown fonts.\MessageBreak
825   See the documentation for details\@gobble}

```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by the command `\DeclareEncodingSubset` that takes the major encoding as the first argument (e.g., `TS1`), the family name as the second argument (e.g., `cmr`), and the subset encoding id as a third, (e.g., 0 for `cmr`).

The default encoding subset to use when nothing is known about the current font family is named `?`.

```

826 \def\DeclareEncodingSubset#1#2#3{%
827   \@ifundefined{#1:#2}%
828     {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
829     {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
830   \@namedef{#1:#2}{#3}}

```

In the original code this was only allowed in the preamble but now `.fds` might contain it so that even in a rollback situation it is necessary to allow it everywhere in the document.

```

831 %\@onlypreamble\DeclareEncodingSubset

```

(End of definition for `\DeclareEncodingSubset`.)

The options for the package are the following:

safe for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

euro enables the "safe" symbols plus the `\texteuro` command. Most newer fonts provide this.

full enables all `TS1` commands; useful only with fonts like EC or CM bright.

almostfull same as “full”, except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

force ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

`\iftc@forced` Switch used to implement the **force** option

```
832 \newif\iftc@forced \tc@forcedfalse
```

(End of definition for `\iftc@forced`.)

This is implemented by defining the default subset:

```
833 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
```

```
834 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
```

```
835 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
```

```
836 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}
```

The default is “almostfull” which means that old documents will work except that `\textcircled` and `\t` will use the kernel defaults (with the advantage that this also works if the current font, as often the case, doesn’t implement these glyphs).

The “force” option simply sets the switch to true.

```
837 \DeclareOption{force}{\tc@forcedtrue}
```

The suggestions to user is to use the “safe” option always unless that balks in which case they could switch to “almostfull” but then better check their output manually.

```
838 \def\tc@errorwarn{\PackageError}
```

```
839 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
```

```
840 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}
```

```
841 \ExecuteOptions{almostfull}
```

```
842 \ProcessOptions\relax
```

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: **#2** and **#5** of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute **#1{#2}#5** otherwise it runs **#3#5**, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
843 \iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
844 \def\CheckEncodingSubset#1#2#3#4#5{%
```

```
845   \ifnum #4>%
```

```
846     0\csname #2:\endcsname
```

```
847     \relax
```

```
848   \expandafter\@firstoftwo
```

```
849   \else
```



```

850 \expandafter\@secondoftwo
851 \fi
852 {#1{#2}}{#3}%
853 #5%
854 }

```

In normal circumstances the test is a bit more complicated: first check if there exists a macro `\(arg2):⟨current-family⟩` and if so use that value to test against, otherwise use the default to test against.

```

855 \else
856 \def\CheckEncodingSubset#1#2#3#4#5{%
857   \ifnum #4>%
858     \expandafter\ifx\csname #2:\f@family\endcsname\relax
859       0\csname #2:?\endcsname
860     \else
861       \csname #2:\f@family\endcsname
862     \fi
863   \relax
864   \expandafter\@firstoftwo
865 \else
866   \expandafter\@secondoftwo
867 \fi
868 {#1{#2}}{#3}%
869 #5%
870 }
871 \fi

```

(End of definition for \CheckEncodingSubset.)

`\tc@subst`

```

872 \def\tc@subst#1{%
873   \tc@errorwarn{textcomp}%
874   {Symbol \string#1 not provided by\MessageBreak
875     font family \f@family\space
876     in TS1 encoding.\MessageBreak Default family used instead}\@eha
877   \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
878 }

```

(End of definition for \tc@subst.)

`\tc@error` `\tc@error` is going to be used in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```

879 % error commands take argument:
880 % #1 symbol to be used
881 \def\tc@error#1{%
882   \PackageError{textcomp}% % should be latex error if general
883   {Accent \string#1 not provided by\MessageBreak
884     font family \f@family\space
885     in TS1 encoding}\@eha
886 }

```

(End of definition for \tc@error.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a “C” with a “=”.

```

887 \def\tc@fake@euro#1{%
888   \leavevmode
889   \PackageInfo{textcomp}{Faking \noexpand#1for font family
890                                     \f@family\MessageBreak in TS1 encoding}%
891   \valign{##\cr
892         \vfil\hbox to 0.07em{\dimen@\f@size\p@
893                                   \math@fontsfalse
894                                   \fontsize{.7\dimen@}\z@\selectfont=\hss}%
895         \vfil\cr%
896         \hbox{C}\crrcr
897   }%
898 }
```

(End of definition for \tc@fake@euro.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```

899 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
900 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
```

(End of definition for \tc@check@symbol and \tc@check@accent.)

We start with the commands that are “safe” and which can be unconditionally set up, first the accents...

```

901 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
902 \DeclareTextAccentDefault{\capitalogonek}{TS1}
903 \DeclareTextAccentDefault{\capitalgrave}{TS1}
904 \DeclareTextAccentDefault{\capitalacute}{TS1}
905 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
906 \DeclareTextAccentDefault{\capitaltilde}{TS1}
907 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
908 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
909 \DeclareTextAccentDefault{\capitalring}{TS1}
910 \DeclareTextAccentDefault{\capitalcaron}{TS1}
911 \DeclareTextAccentDefault{\capitalbreve}{TS1}
912 \DeclareTextAccentDefault{\capitalmacron}{TS1}
913 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}
```

...and then the other glyphs.

```

914 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
915 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
916 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
917 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
918 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
919 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
920 \DeclareTextSymbolDefault{\textdollar}{TS1}
```

```

921 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
922 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
923 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
924 \DeclareTextSymbolDefault{\textminus}{TS1}
925 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
926 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
927 \DeclareTextSymbolDefault{\textasciigrave}{TS1}
928 \DeclareTextSymbolDefault{\texttildelow}{TS1}
929 \DeclareTextSymbolDefault{\textasciibreve}{TS1}
930 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
931 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
932 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
933 \DeclareTextSymbolDefault{\textdagger}{TS1}
934 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
935 \DeclareTextSymbolDefault{\textbardbl}{TS1}
936 \DeclareTextSymbolDefault{\textperthousand}{TS1}
937 \DeclareTextSymbolDefault{\textbullet}{TS1}
938 \DeclareTextSymbolDefault{\textcelsius}{TS1}
939 \DeclareTextSymbolDefault{\textflorin}{TS1}
940 \DeclareTextSymbolDefault{\texttrademark}{TS1}
941 \DeclareTextSymbolDefault{\textcent}{TS1}
942 \DeclareTextSymbolDefault{\textsterling}{TS1}
943 \DeclareTextSymbolDefault{\textyen}{TS1}
944 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
945 \DeclareTextSymbolDefault{\textsection}{TS1}
946 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
947 \DeclareTextSymbolDefault{\textcopyright}{TS1}
948 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
949 \DeclareTextSymbolDefault{\textlnot}{TS1}
950 \DeclareTextSymbolDefault{\textregistered}{TS1}
951 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
952 \DeclareTextSymbolDefault{\textdegree}{TS1}
953 \DeclareTextSymbolDefault{\textpm}{TS1}
954 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
955 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
956 \DeclareTextSymbolDefault{\textasciacute}{TS1}
957 \DeclareTextSymbolDefault{\textmu}{TS1}
958 \DeclareTextSymbolDefault{\textparagraph}{TS1}
959 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
960 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
961 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
962 \DeclareTextSymbolDefault{\textonequarter}{TS1}
963 \DeclareTextSymbolDefault{\textonehalf}{TS1}
964 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
965 \DeclareTextSymbolDefault{\texttimes}{TS1}
966 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

967 \DeclareTextCommandDefault{\texteuro}
968   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

969 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```

970 \DeclareTextCommandDefault{\textestimated}%
971   {\tc@check@symbol3\textestimated}
972 \DeclareTextCommandDefault{\textcurrency}%
973   {\tc@check@symbol3\textcurrency}

```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```

974 \DeclareTextCommandDefault{\capitaltie}%
975   {\tc@check@accent2\capitaltie}
976 \DeclareTextCommandDefault{\newtie}%
977   {\tc@check@accent2\newtie}
978 \DeclareTextCommandDefault{\capitalnewtie}%
979   {\tc@check@accent2\capitalnewtie}
980 \DeclareTextCommandDefault{\textleftarrow}%
981   {\tc@check@symbol2\textleftarrow}
982 \DeclareTextCommandDefault{\textrightarrow}%
983   {\tc@check@symbol2\textrightarrow}
984 \DeclareTextCommandDefault{\textblank}%
985   {\tc@check@symbol2\textblank}
986 \DeclareTextCommandDefault{\textdblhyphen}%
987   {\tc@check@symbol2\textdblhyphen}
988 \DeclareTextCommandDefault{\textzerooldstyle}%
989   {\tc@check@symbol2\textzerooldstyle}
990 \DeclareTextCommandDefault{\textoneoldstyle}%
991   {\tc@check@symbol2\textoneoldstyle}
992 \DeclareTextCommandDefault{\texttwooldstyle}%
993   {\tc@check@symbol2\texttwooldstyle}
994 \DeclareTextCommandDefault{\textthreeoldstyle}%
995   {\tc@check@symbol2\textthreeoldstyle}
996 \DeclareTextCommandDefault{\textfouroldstyle}%
997   {\tc@check@symbol2\textfouroldstyle}
998 \DeclareTextCommandDefault{\textfiveoldstyle}%
999   {\tc@check@symbol2\textfiveoldstyle}
1000 \DeclareTextCommandDefault{\textsixoldstyle}%
1001   {\tc@check@symbol2\textsixoldstyle}
1002 \DeclareTextCommandDefault{\textsevenoldstyle}%
1003   {\tc@check@symbol2\textsevenoldstyle}
1004 \DeclareTextCommandDefault{\texteightoldstyle}%
1005   {\tc@check@symbol2\texteightoldstyle}
1006 \DeclareTextCommandDefault{\textnineoldstyle}%
1007   {\tc@check@symbol2\textnineoldstyle}
1008 \DeclareTextCommandDefault{\textlangle}%
1009   {\tc@check@symbol2\textlangle}
1010 \DeclareTextCommandDefault{\textrangle}%
1011   {\tc@check@symbol2\textrangle}
1012 \DeclareTextCommandDefault{\textmho}%
1013   {\tc@check@symbol2\textmho}
1014 \DeclareTextCommandDefault{\textbigcircle}%
1015   {\tc@check@symbol2\textbigcircle}
1016 \DeclareTextCommandDefault{\textuparrow}%
1017   {\tc@check@symbol2\textuparrow}
1018 \DeclareTextCommandDefault{\textdownarrow}%

```

```

1019     {\tc@check@symbol2\textdownarrow}
1020 \DeclareTextCommandDefault{\textborn}%
1021     {\tc@check@symbol2\textborn}
1022 \DeclareTextCommandDefault{\textdivorced}%
1023     {\tc@check@symbol2\textdivorced}
1024 \DeclareTextCommandDefault{\textdied}%
1025     {\tc@check@symbol2\textdied}
1026 \DeclareTextCommandDefault{\textleaf}%
1027     {\tc@check@symbol2\textleaf}
1028 \DeclareTextCommandDefault{\textmarried}%
1029     {\tc@check@symbol2\textmarried}
1030 \DeclareTextCommandDefault{\textmusicalnote}%
1031     {\tc@check@symbol2\textmusicalnote}
1032 \DeclareTextCommandDefault{\textdblhyphenchar}%
1033     {\tc@check@symbol2\textdblhyphenchar}
1034 \DeclareTextCommandDefault{\textdollaroldstyle}%
1035     {\tc@check@symbol2\textdollaroldstyle}
1036 \DeclareTextCommandDefault{\textcentoldstyle}%
1037     {\tc@check@symbol2\textcentoldstyle}
1038 \DeclareTextCommandDefault{\textcolonmonetary}%
1039     {\tc@check@symbol2\textcolonmonetary}
1040 \DeclareTextCommandDefault{\textwon}%
1041     {\tc@check@symbol2\textwon}
1042 \DeclareTextCommandDefault{\textnaira}%
1043     {\tc@check@symbol2\textnaira}
1044 \DeclareTextCommandDefault{\textguarani}%
1045     {\tc@check@symbol2\textguarani}
1046 \DeclareTextCommandDefault{\textpeso}%
1047     {\tc@check@symbol2\textpeso}
1048 \DeclareTextCommandDefault{\textlira}%
1049     {\tc@check@symbol2\textlira}
1050 \DeclareTextCommandDefault{\textrecipe}%
1051     {\tc@check@symbol2\textrecipe}
1052 \DeclareTextCommandDefault{\textinterrobang}%
1053     {\tc@check@symbol2\textinterrobang}
1054 \DeclareTextCommandDefault{\textinterrobangdown}%
1055     {\tc@check@symbol2\textinterrobangdown}
1056 \DeclareTextCommandDefault{\textdong}%
1057     {\tc@check@symbol2\textdong}
1058 \DeclareTextCommandDefault{\textpertenthousand}%
1059     {\tc@check@symbol2\textpertenthousand}
1060 \DeclareTextCommandDefault{\textpilcrow}%
1061     {\tc@check@symbol2\textpilcrow}
1062 \DeclareTextCommandDefault{\textbaht}%
1063     {\tc@check@symbol2\textbaht}
1064 \DeclareTextCommandDefault{\textnumero}%
1065     {\tc@check@symbol2\textnumero}
1066 \DeclareTextCommandDefault{\textdiscount}%
1067     {\tc@check@symbol2\textdiscount}
1068 \DeclareTextCommandDefault{\textopenbullet}%
1069     {\tc@check@symbol2\textopenbullet}
1070 \DeclareTextCommandDefault{\textservicemark}%
1071     {\tc@check@symbol2\textservicemark}
1072 \DeclareTextCommandDefault{\textlquill}%

```

```

1073 {\tc@check@symbol2\textlquill}
1074 \DeclareTextCommandDefault{\textrquill}%
1075 {\tc@check@symbol2\textrquill}
1076 \DeclareTextCommandDefault{\textcopyleft}%
1077 {\tc@check@symbol2\textcopyleft}
1078 \DeclareTextCommandDefault{\textcircledP}%
1079 {\tc@check@symbol2\textcircledP}
1080 \DeclareTextCommandDefault{\textreferencemark}%
1081 {\tc@check@symbol2\textreferencemark}
1082 \DeclareTextCommandDefault{\textsurd}%
1083 {\tc@check@symbol2\textsurd}

```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full TS1) we pick the symbols up from the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```

1084 \DeclareTextCommandDefault{\textcircled}
1085 {\CheckEncodingSubset\UseTextAccent{TS1}%
1086 {\UseTextAccent{OMS}}1\textcircled}
1087 \DeclareTextCommandDefault{\t}
1088 {\CheckEncodingSubset\UseTextAccent{TS1}%
1089 {\UseTextAccent{OML}}1\t}

```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimized for this encoding (and not for the default encoding).

```

1090 \input{ts1enc.def}

```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to L^AT_EX which will prevent the usage of the TS1 versions. So we better get rid of them:

```

1091 \UndeclareTextCommand{\textsterling}{OT1}
1092 \UndeclareTextCommand{\textdollar} {OT1}

```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```

1093 %\UndeclareTextCommand{\textsterling}{OT4}
1094 %\UndeclareTextCommand{\textdollar} {OT4}

```

From the T1 encoding there are two candidates for removal: `%0` and `%00` since these are both constructed from `%` followed by a tiny ‘`o`’ rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in `%■` rather than `%0` while the real glyph (at least for `\textperthousand`) is available in the PostScript version of TS1. So for the moment we compromise by removing the T1 declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although `%0` and `%00` are not taken from the same physical font) and with PostScript fonts `%00` will come out correctly while `%000` will most likely look like `%■` — which is probably an improvement over just getting a single ‘`■`’ to indicate a completely missing glyph, which would happen if we also ‘undeclared’ `\textpertenthousand`.

```

1095 \UndeclareTextCommand{\textperthousand}{T1}
1096 %\UndeclareTextCommand{\textpertenthousand}{T1}

```

5.1.1 Supporting oldstyle digits

```

1097 \DeclareRobustCommand\oldstylenums[1]{%
1098   \begingroup
1099   \ifmmode
1100     \mathgroup\symletters #1%
1101   \else
1102     \CheckEncodingSubset\@use@text@encoding{TS1}%
1103     {\PackageWarning{textcomp}%
1104      {Oldstyle digits unavailable for
1105       family \f@family.\MessageBreak
1106       Lining digits used instead}}%
1107     \tw@{#1}%
1108   \fi
1109   \endgroup
1110 }

```

5.1.2 Subset encoding defaults

For many font families commonly used in the T_EX world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg` if they own other fonts.

However, if the option “forced” was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```

1111 \iftc@forced \else
      Computer modern based fonts (e.g., CM, CM-Bright, Concrete):
1112 \DeclareEncodingSubset{TS1}{cmr}      {0}
1113 \DeclareEncodingSubset{TS1}{cmss}     {0}
1114 \DeclareEncodingSubset{TS1}{cmtt}     {0}
1115 \DeclareEncodingSubset{TS1}{cmvtt}    {0}
1116 \DeclareEncodingSubset{TS1}{cmbr}     {0}
1117 \DeclareEncodingSubset{TS1}{cmtl}     {0}
1118 \DeclareEncodingSubset{TS1}{ccr}      {0}
      PSNFSS fonts:
1119 \DeclareEncodingSubset{TS1}{ptm}      {4}
1120 \DeclareEncodingSubset{TS1}{pcr}      {4}
1121 \DeclareEncodingSubset{TS1}{phv}      {4}
1122 \DeclareEncodingSubset{TS1}{ppl}      {3}
1123 \DeclareEncodingSubset{TS1}{pag}      {4}
1124 \DeclareEncodingSubset{TS1}{pbk}      {4}
1125 \DeclareEncodingSubset{TS1}{pnc}      {4}
1126 \DeclareEncodingSubset{TS1}{pzc}      {4}
1127 \DeclareEncodingSubset{TS1}{bch}      {4}
1128 \DeclareEncodingSubset{TS1}{put}      {5}
      Other CTAN fonts (probably not complete):
1129 \DeclareEncodingSubset{TS1}{uag}      {5}
1130 \DeclareEncodingSubset{TS1}{ugq}      {5}
1131 \DeclareEncodingSubset{TS1}{ul8}      {4}
1132 \DeclareEncodingSubset{TS1}{ul9}      {4} % (LuxiSans, one day)
1133 \DeclareEncodingSubset{TS1}{augie}    {5}
1134 \DeclareEncodingSubset{TS1}{dayrom}    {3}
1135 \DeclareEncodingSubset{TS1}{dayroms}  {3}
1136 \DeclareEncodingSubset{TS1}{pxr}      {0}

```

```

1137 \DeclareEncodingSubset{TS1}{pxss}    {0}
1138 \DeclareEncodingSubset{TS1}{pxtt}    {0}
1139 \DeclareEncodingSubset{TS1}{txr}      {0}
1140 \DeclareEncodingSubset{TS1}{txss}     {0}
1141 \DeclareEncodingSubset{TS1}{txtt}     {0}

```

Latin Modern and TeX Gyre:

```

1142 \DeclareEncodingSubset{TS1}{lmr}      {0}
1143 \DeclareEncodingSubset{TS1}{lmdh}     {0}
1144 \DeclareEncodingSubset{TS1}{lmss}     {0}
1145 \DeclareEncodingSubset{TS1}{lmssq}    {0}
1146 \DeclareEncodingSubset{TS1}{lmvtt}    {0}
1147 \DeclareEncodingSubset{TS1}{lmtt}     {0}

1148 \DeclareEncodingSubset{TS1}{qhv}      {0}
1149 \DeclareEncodingSubset{TS1}{qag}      {0}
1150 \DeclareEncodingSubset{TS1}{qbk}      {0}
1151 \DeclareEncodingSubset{TS1}{qcr}      {0}
1152 \DeclareEncodingSubset{TS1}{qcs}      {0}
1153 \DeclareEncodingSubset{TS1}{qpl}      {0}
1154 \DeclareEncodingSubset{TS1}{qtm}      {0}
1155 \DeclareEncodingSubset{TS1}{qzc}      {0}
1156 \DeclareEncodingSubset{TS1}{qhvc}     {0}

```

Fourier-GUTenberg:

```

1157 \DeclareEncodingSubset{TS1}{futs}     {4}
1158 \DeclareEncodingSubset{TS1}{futex}    {4}
1159 \DeclareEncodingSubset{TS1}{futj}     {4}

```

Y&Y's Lucida Bright

```

1160 \DeclareEncodingSubset{TS1}{hlh}      {3}
1161 \DeclareEncodingSubset{TS1}{hls}      {3}
1162 \DeclareEncodingSubset{TS1}{hlst}     {3}

```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```

1163 \DeclareEncodingSubset{TS1}{hlct}     {5}
1164 \DeclareEncodingSubset{TS1}{hlx}      {5}
1165 \DeclareEncodingSubset{TS1}{hlce}     {5}
1166 \DeclareEncodingSubset{TS1}{hlcn}     {5}
1167 \DeclareEncodingSubset{TS1}{hlcw}     {5}
1168 \DeclareEncodingSubset{TS1}{hlcf}     {5}

```

Other commercial families...

```

1169 \DeclareEncodingSubset{TS1}{pplx}     {3}
1170 \DeclareEncodingSubset{TS1}{pplj}     {3}
1171 \DeclareEncodingSubset{TS1}{ptmx}     {4}
1172 \DeclareEncodingSubset{TS1}{ptmj}     {4}

```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```

1173 \InputIfFileExists{textcomp.cfg}
1174 {\PackageInfo{textcomp}{Local configuration file used}}{}

```



```

1175 \fi
1176 \end{TS1oldsty}

```

6 The checkencodingsubset.tex file

This is a simple file that asks for a name of a font family and then displays information about the TS1 encoding for this family and recommends the right encoding subset (to be used with `\DeclareEncodingsubset`) for this family.

```

1177 (*TS1check)
1178 \ProvidesFile{checkencodingsubset.tex}
1179 [2024/10/18 v0.5b Figure out safe TS1 encoding subsets]

1180 \let\typeoutdetails\typeout
1181 %\def\typeoutdetails#1{} % alternative definition used below

```

For the purpose of this check a glyph exists if the font slot is occupied—too bad if that contains the wrong glyph or some tofu. If it “exists” we return 0 otherwise 1. This way we can call this macro several times in a row and obtain a number that is 0 if all glyphs are existing or greater than 0 if any of them is missing.

The second argument (holding the command name for a symbol) is not used during these tests.

```

1182 \def\doesglyphexist#1#2{\iffontchar\testFont #1 0\else 1\relax \fi}

```

This macro also tests and outputs some information about the symbol if it is missing. This time we make use of the second argument.

```

1183 \def\glyphmissingdetails#1#2{\iffontchar\testFont #1 \else
1184   \typeoutdetails{\space\space\space ==> \string#2 (#1) is missing}\fi}

1185 \newif\ifsafesubencodingfound
1186 \newif\ifcoremisses

```

Testing a group of symbols that belong to one sub-encoding. More precisely, the symbols that become unavailable if you change from sub-encoding x (#2) to $x + 1$ (#3). As far as the code is concerned, the symbols that are supposed to be always available (the core) become available if we test the group -1 and 0.

The first argument contains the testing code and is supposed to return a single number greater or equal to zero.

```

1187 \def\testgroup#1#2#3{%
1188   \ifnum 0 = #1%
1189     \ifnum #2<0
1190       \typeoutdetails{All glyphs in core exist}%
1191     \else
1192       \typeoutdetails{All glyphs between sub-encoding #2 and #3 exist}%
1193     \fi
1194   \else
1195     \ifnum #2<0
1196       \typeoutdetails{*****}%
1197       \typeoutdetails{Some glyphs are missing from core:}%
1198       \coremisstrue
1199       \ifsafesubencodingfound \else
1200         \def\subencodingresult{#2}%
1201       \fi
1202     \else

```

```

1203     \typeoutdetails{Some glyphs are missing from sub-encoding #2:}%
1204     \ifsafesubencodingfound \else
1205         \def\subencodingresult{#3}%
1206     \fi
1207 \fi

```

If some glyphs are missing, we rerun the test code but this time using `\glyphmissingdetails`.

```

1208     {\let\doesglyphexist \glyphmissingdetails #1}%

```

And because we had misses we have definitely found the subset.

```

1209     \safesubencodingfoundtrue
1210 \fi
1211 }

```

The currently defined subset for the family is either stored in `\TS1:<family>` if it was declared, or it is the default subset which is stored in `\TS1:?`.

```

1212 \def\currsubencoding#1{\csname TS1:\ifcsname TS1:#1\endcsname #1\else ?\fi\endcsname}

```

If a font family is not found when declaring it with `\DeclareFixedFont` we end up with the following font. This can then be used as a simple test if we failed loading the TS1 font.

```

1213 \DeclareFixedFont\cmrFont{TS1}{cmr}{m}{n}{10pt}
    Check for all glyphs in all encoding subsets ...
1214 \def\testallgroups#1{%
1215     \DeclareFixedFont\testFont{TS1}{#1}{m}{n}{10pt}%
1216     \ifx\testFont\cmrFont
1217         \typeout{***** Font family #1 not found ****}%
1218     \else

```

We haven't checked anything yet.

```

1219     \safesubencodingfoundfalse
1220     \coremissesfalse
1221     \typeoutdetails{^^J-----}%
1222     \typeoutdetails{Testing font family #1^^J(currently TS1-sub-encoding
1223         \currsubencoding{#1})}%
1224     \typeout{-----}%

```

Then we start testing the groups beginning with the glyphs between sub-encoding 8 and 9. If any of them is missing (checked with `\doesglyphexist`) then we already know that 9 is the correct answer.

```

1225     \testgroup{%
1226         \doesglyphexist{21}{\texttwelveudash}%
1227         \doesglyphexist{22}{\textthreequartersemdash}%
1228         \doesglyphexist{134}{\textbardbl}%
1229         \doesglyphexist{137}{\textcelsius}%
1230         \doesglyphexist{178}{\texttwosuperior}%
1231         \doesglyphexist{179}{\textthreesuperior}%
1232         \doesglyphexist{185}{\textonesuperior}%
1233     }{8}{9}%

```

Nevertheless we go on with further groups so that the output lists all missing glyphs.

```

1234     \testgroup{%
1235         \doesglyphexist{32}{\textblank}%
1236         \doesglyphexist{148}{\textinterrobang}%
1237         \doesglyphexist{149}{\textinterrobangdown}%
1238         \doesglyphexist{191}{\texteuro}%

```

```

1239 }{7}{8}%
1240 \testgroup{%
1241     \doesglyphexist{47}{\textfractionsolidus}%
1242     \doesglyphexist{61}{\textminus}%
1243     \doesglyphexist{87}{\textohm}%
1244     \doesglyphexist{181}{\textmu}%
1245 }{6}{7}%
1246 \testgroup{%
1247     \doesglyphexist{140}{\textflorin}%
1248     \doesglyphexist{164}{\textcurrency}%
1249 }{5}{6}%
1250 \testgroup{%
1251     \doesglyphexist{155}{\textnumero}%
1252     \doesglyphexist{157}{\textestimated}%
1253 }{4}{5}%
1254 \testgroup{%
1255     \doesglyphexist{24}{\textleftarrow}%
1256     \doesglyphexist{25}{\textrightarrow}%
1257     \doesglyphexist{94}{\textuparrow}%
1258     \doesglyphexist{95}{\textdownarrow}%
1259     \doesglyphexist{141}{\textcolonmonetary}%
1260     \doesglyphexist{142}{\textwon}%
1261     \doesglyphexist{146}{\textlira}%
1262     \doesglyphexist{150}{\textdong}%
1263 }{3}{4}%
1264 \testgroup{%
1265     \doesglyphexist{60}{\textlangle}%
1266     \doesglyphexist{62}{\textrangle}%
1267 }{2}{3}%
1268 \testgroup{%
1269     \doesglyphexist{0}{\capitalgrave}%
1270     \doesglyphexist{1}{\capitalacute}%
1271     \doesglyphexist{2}{\capitalcircumflex}%
1272     \doesglyphexist{3}{\capitaltilde}%
1273     \doesglyphexist{4}{\capitaldieresis}%
1274     \doesglyphexist{5}{\capitalhungarumlaut}%
1275     \doesglyphexist{6}{\capitalring}%
1276     \doesglyphexist{7}{\capitalcaron}%
1277     \doesglyphexist{8}{\capitalbreve}%
1278     \doesglyphexist{9}{\capitalmacron}%
1279     \doesglyphexist{10}{\capitaldotaccent}%
1280     \doesglyphexist{11}{\capitalcedilla}%
1281     \doesglyphexist{12}{\capitalogonek}%
1282     \doesglyphexist{26}{\t}%
1283     \doesglyphexist{27}{\capitaltie}%
1284     \doesglyphexist{28}{\newtie}%
1285     \doesglyphexist{29}{\capitalnewtie}%
1286     \doesglyphexist{45}{\textdblhyphen}%
1287     \doesglyphexist{48}{\textzerooldstyle}%
1288     \doesglyphexist{49}{\textoneoldstyle}%
1289     \doesglyphexist{50}{\texttwooldstyle}%
1290     \doesglyphexist{51}{\textthreeoldstyle}%
1291     \doesglyphexist{52}{\textfouroldstyle}%
1292     \doesglyphexist{53}{\textfiveoldstyle}%

```

```

1293 \doesglyphexist{54}{\textsixoldstyle}%
1294 \doesglyphexist{55}{\textsevenoldstyle}%
1295 \doesglyphexist{56}{\texteightoldstyle}%
1296 \doesglyphexist{57}{\textnineoldstyle}%
1297 \doesglyphexist{77}{\textmho}%
1298 \doesglyphexist{79}{\textbigcircle}%
1299 \doesglyphexist{91}{\textlbrackdbl}%
1300 \doesglyphexist{93}{\textrbrackdbl}%
1301 \doesglyphexist{96}{\textasciigrave}%
1302 \doesglyphexist{98}{\textborn}%
1303 \doesglyphexist{99}{\textdivorced}%
1304 \doesglyphexist{100}{\textdied}%
1305 \doesglyphexist{108}{\textleaf}%
1306 \doesglyphexist{109}{\textmarried}%
1307 \doesglyphexist{110}{\textmusicalnote}%
1308 \doesglyphexist{126}{\texttildelow}%
1309 \doesglyphexist{127}{\textdblhyphenchar}%
1310 \doesglyphexist{128}{\textasciibreve}%
1311 \doesglyphexist{129}{\textasciicaron}%
1312 \doesglyphexist{175}{\textasciimacron}%
1313 \doesglyphexist{130}{\textacutedbl}%
1314 \doesglyphexist{131}{\textgravedbl}%
1315 \doesglyphexist{138}{\textdollaroldstyle}%
1316 \doesglyphexist{139}{\textcentoldstyle}%
1317 \doesglyphexist{143}{\textnaira}%
1318 \doesglyphexist{144}{\textguarani}%
1319 \doesglyphexist{145}{\textpeso}%
1320 \doesglyphexist{147}{\textrecipe}%
1321 \doesglyphexist{152}{\textpertenthousand}%
1322 \doesglyphexist{153}{\textpilcrow}%
1323 \doesglyphexist{154}{\textbaht}%
1324 \doesglyphexist{156}{\textdiscount}%
1325 \doesglyphexist{158}{\textopenbullet}%
1326 \doesglyphexist{159}{\textservicemark}%
1327 \doesglyphexist{160}{\textlquill}%
1328 \doesglyphexist{161}{\textrquill}%
1329 \doesglyphexist{168}{\textasciidieresis}%
1330 \doesglyphexist{171}{\textcopyleft}%
1331 \doesglyphexist{173}{\textcircledP}%
1332 \doesglyphexist{180}{\textasciiacute}%
1333 \doesglyphexist{184}{\textreferencemark}%
1334 \doesglyphexist{187}{\textsurd}%
1335 }{1}{2}%

```

All fonts (up to now) that belong to sub-encoding 1 do have the `\textcircled` glyph, but it is too small to be usable. So this test for this group currently doesn't do much good—but who knows maybe one day a font shows up in which this glyph is actually missing.

```

1336 \testgroup{%
1337 \doesglyphexist{79}{\textcircled}% this is not a proper test because the symbol is
1338 % usually available but not usable
1339 }{0}{1}%
1340 \testgroup{%
1341 \doesglyphexist{13}{\textquotestraightbase}%

```

```

1342 \doesglyphexist{18}{\textquotestraightdblbase}%
1343 \doesglyphexist{23}{\textcapitalcompwordmark}%
1344 \doesglyphexist{31}{\textascendercompwordmark}%
1345 \doesglyphexist{36}{\textdollar}%
1346 \doesglyphexist{39}{\textquotesingle}%
1347 \doesglyphexist{42}{\textasteriskcentered}%
1348 \doesglyphexist{132}{\textdagger}%
1349 \doesglyphexist{133}{\textdaggerdbl}%
1350 \doesglyphexist{135}{\textperthousand}%
1351 \doesglyphexist{136}{\textbullet}%
1352 \doesglyphexist{151}{\texttrademark}%
1353 \doesglyphexist{162}{\textcent}%
1354 \doesglyphexist{163}{\textsterling}%
1355 \doesglyphexist{165}{\textyen}%
1356 \doesglyphexist{166}{\textbrokenbar}%
1357 \doesglyphexist{167}{\textsection}%
1358 \doesglyphexist{169}{\textcopyright}%
1359 \doesglyphexist{170}{\textordfeminine}%
1360 \doesglyphexist{172}{\textlnot}%
1361 \doesglyphexist{174}{\textregistered}%
1362 \doesglyphexist{176}{\textdegree}%
1363 \doesglyphexist{177}{\textpm}%
1364 \doesglyphexist{182}{\textparagraph}%
1365 \doesglyphexist{183}{\textperiodcentered}%
1366 \doesglyphexist{186}{\textordmasculine}%
1367 \doesglyphexist{188}{\textonequarter}%
1368 \doesglyphexist{189}{\textonehalf}%
1369 \doesglyphexist{190}{\textthreequarters}%
1370 \doesglyphexist{214}{\texttimes}%
1371 \doesglyphexist{246}{\textdiv}%
1372 }{-1}{0}%

```

If all groups have all glyphs then we have the full encoding (subset 0).

```

1373 \ifsafesubencodingfound\else
1374 \def\subencodingresult{0}%
1375 \fi

```

If the font is missing some of the core glyphs we make a remark about this, because they will never display.

```

1376 \typeoutdetails{-----}%
1377 \typeout{TS1 encoding subset for #1\ifcoremisses \space(ignore core misses)\fi
1378 \space (\ifnum\subencodingresult =
1379 \currsubencoding{#1} ok\else bad\fi)}%
1380 \typeout{Use sub-encoding \subencodingresult
1381 \ifnum\subencodingresult = \currsubencoding{#1}\else
1382 \space (not \currsubencoding{#1})\fi}
1383 \typeout{-----^^J}%
1384 \fi
1385 }

```

This tests all declarations (or most of them) that have been added to the kernel. It is called if no family is given interactively.

```

1386 \long\def\testallkerneldefinedfamilies{%
1387 \testallgroups{ccr}% {0}
1388 \testallgroups{cmbr}% {0}

```

```

1389 %%\testallgroups{cmr}% {0} % don't test this one as it is the fallback
1390 % thus reports that the family is not found
1391 \testallgroups{cmss}% {0}
1392 \testallgroups{cmtl}% {0}
1393 \testallgroups{cmtt}% {0}
1394 \testallgroups{cmvtt}% {0}
1395 \testallgroups{pxr}% {0}
1396 \testallgroups{pxss}% {0}
1397 \testallgroups{pxtt}% {0}
1398 \testallgroups{qag}% {0}
1399 \testallgroups{qbk}% {0}
1400 \testallgroups{qcr}% {0}
1401 \testallgroups{qcs}% {0}
1402 \testallgroups{qhvc}% {0}
1403 \testallgroups{qhv}% {0}
1404 \testallgroups{qpl}% {0}
1405 \testallgroups{qtm}% {0}
1406 \testallgroups{qzc}% {0}
1407 \testallgroups{txr}% {0}
1408 \testallgroups{txss}% {0}
1409 \testallgroups{txtt}% {0}
1410 %
1411 % Next would claim to be 0 (or 2)
1412 %
1413 %%\testallgroups{lmr}% {1}
1414 %%\testallgroups{lmdh}% {1}
1415 %%\testallgroups{lms}% {1}
1416 %%\testallgroups{lmsq}% {1}
1417 %%\testallgroups{lmvtt}% {1}
1418 %%\testallgroups{lmtt}% {1} % missing TM, SM and pertenthousand so really 2
1419 %
1420 % these are no longer in TeX Live
1421 %
1422 %%\testallgroups{ptmx}% {2} % gone for a long time it seems
1423 %%\testallgroups{ptmj}% {2} % ditto
1424 %%\testallgroups{ul8}% {2} % ditto
1425 %
1426 % next block has tofu chars so results are wrong
1427 %
1428 %%\testallgroups{bch}% {5} % tofu for blank, ohm
1429 %%\testallgroups{futj}% {5} % tofu for blank, interrobang/down, ohm
1430 %%\testallgroups{futs}% {5} % tofu for blank, ohm
1431 %%\testallgroups{futz}% {5} % probably (currently broken distrib)
1432 %%\testallgroups{pag}% {5} % tofu for blank, interrobang/down, ohm
1433 %%\testallgroups{pbk}% {5} % tofu for blank, interrobang/down, ohm
1434 %%\testallgroups{pcr}% {5} % tofu for blank, interrobang/down, ohm
1435 %%\testallgroups{phv}% {5} % tofu for blank, interrobang/down, ohm
1436 %%\testallgroups{pnc}% {5} % tofu for blank, interrobang/down, ohm
1437 %%\testallgroups{pplj}% {5} % tofu for blank
1438 %%\testallgroups{pplx}% {5} % tofu for blank
1439 %%\testallgroups{ppl}% {5} % tofu for blank interrobang/down
1440 %%\testallgroups{ptm}% {5} % tofu for blank, interrobang/down, ohm
1441 %%\testallgroups{pzc}% {5} % tofu for blank, interrobang/down, ohm
1442 %%\testallgroups{ul9}% {5} % tofu for blank, interrobang/down, ohm

```

```

1443 %\testallgroups{dayroms}%{6} % tofu for blank, interrobang/down, ohm
1444 %\testallgroups{dayrom}% {6} % tofu for blank, interrobang/down, ohm
1445 %\testallgroups{augie}%{8} % really only missing euro and full of tofu
1446 %\testallgroups{put}% {8}
1447 %\testallgroups{uag}% {8} % probably (currently broken distrib)
1448 %\testallgroups{ugq}% {8}
1449 %
1450 \testallgroups{zi4}% {9}
1451 %
1452 %% not installed normally
1453 %
1454 %\testallgroups{hls}% {5}
1455 %\testallgroups{hlst}% {5}
1456 %\testallgroups{hlct}% {5}
1457 %\testallgroups{hlh}% {5}
1458 %\testallgroups{hlx}% {8}
1459 %\testallgroups{hlce}% {8}
1460 %\testallgroups{hlcn}% {8}
1461 %\testallgroups{hlcw}% {8}
1462 %\testallgroups{hlcf}% {8}
1463
1464 \testallgroups{lato-LF}% {0} % with a bunch of tofu inside --- should probably be changed
1465 \testallgroups{opensans-TLF}%{0} % with a bunch of tofu inside --- should probably be change
1466 \testallgroups{cantarell-TLF}% {0} % with a bunch of tofu inside --- should probably be cha
1467 \testallgroups{fbb-LF}% {0} % missing centoldstyle ---> 2
1468 \testallgroups{tli}% {1} % with lots of tofu inside --- should probably be changed
1469 \testallgroups{Alegreya-OsF}% {2}
1470 \testallgroups{AlegreyaSans-OsF}% {2}
1471 \testallgroups{DejaVuSans-TLF}% {2}
1472 \testallgroups{DejaVuSansCondensed-TLF}% {2}
1473 \testallgroups{DejaVuSansMono-TLF}% {2} this is missing \textfractionsolidus which makes it 7
1474 \testallgroups{EBGaramond-LF}% {2}
1475 \testallgroups{Tempora-TLF}% {2}
1476 \testallgroups{Tempora-TOfF}% {2}
1477 \testallgroups{Arimo-TLF}% {3}
1478 \testallgroups{Crlt-TLF}% {3} changed from Carlito-
1479 \testallgroups{FiraSans-LF}% {3} should be 4
1480 \testallgroups{IBMPlexSans-TLF}% {3}
1481 \testallgroups{Merriwthr-OsF}% {3} changed from Merriweather- and should be 2
1482 \testallgroups{Montserrat-LF}% {3} now 2
1483 \testallgroups{MontserratAlternates-LF}%{3} now 2
1484 \testallgroups{SourceCodePro-TLF}% {3}
1485 \testallgroups{SourceCodePro-TOfF}% {3}
1486 \testallgroups{SourceSansPro-OsF}% {3}
1487 \testallgroups{SourceSerifPro-LF}% {3}
1488 \testallgroups{Tinos-TLF}% {3}
1489 \testallgroups{AccanthisADFStdNoThree-LF}%{4}
1490 \testallgroups{Cabin-TLF}% {4}
1491 \testallgroups{Caladea-TLF}% {4}
1492 \testallgroups{Chivo-LF}% {4}
1493 \testallgroups{ClearSans-TLF}% {4}
1494 \testallgroups{Coelacanth-LF}% {4}
1495 \testallgroups{CrimsonPro-LF}% {4}
1496 \testallgroups{FiraMono-TLF}% {4}

```

```

1497 \testallgroups{FiraMono-T0sF}% {4}
1498 \testallgroups{Go-TLF}% {4}
1499 \testallgroups{GoMono-TLF}% {4}
1500 \testallgroups{InriaSans-LF}% {4}
1501 \testallgroups{InriaSerif-LF}% {4}
1502 \testallgroups{LibertinusSans-LF}% {4}
1503 \testallgroups{LibertinusSerif-LF}% {4}
1504 \testallgroups{LibreBodoni-TLF}% {4}
1505 \testallgroups{LibreFranklin-TLF}% {4}
1506 \testallgroups{LinguisticsPro-LF}% {4}
1507 \testallgroups{LinguisticsPro-0sF}% {4}
1508 \testallgroups{LinuxBiolinumT-LF}% {4}
1509 \testallgroups{LinuxLibertineT-LF}% {4}
1510 \testallgroups{MerriwthrSans-0sF}% {4} name change and now 2
1511 \testallgroups{MintSpirit-LF}% {4}
1512 \testallgroups{MintSpiritNoTwo-LF}% {4}
1513 \testallgroups{PTMono-TLF}% {4}
1514 \testallgroups{PTSans-TLF}% {4}
1515 \testallgroups{PTSansCaption-TLF}% {4}
1516 \testallgroups{PTSansNarrow-TLF}% {4}
1517 \testallgroups{PTSerif-TLF}% {4}
1518 \testallgroups{PTSerifCaption-TLF}% {4}
1519 \testallgroups{Raleway-TLF}% {4}
1520 \testallgroups{Raleway-T0sF}% {4}
1521 \testallgroups{Roboto-LF}% {4}
1522 \testallgroups{RobotoMono-TLF}% {4}
1523 \testallgroups{RobotoSlab-TLF}% {4}
1524 \testallgroups{Rosario-LF}% {4}
1525 \testallgroups{SticksTooText-LF}% {4}
1526 \testallgroups{UniversalisADFStd-LF}% {4}
1527 \testallgroups{Almndr-0sF}% {5} name change
1528 \testallgroups{Baskervaldx-LF}% {5}
1529 \testallgroups{BaskervilleF-LF}% {5} now 2
1530 \testallgroups{Bttr-TLF}% {5} name changed from Bitter-...
1531 \testallgroups{Cinzel-LF}% {5}
1532 \testallgroups{CinzelDecorative-LF}% {5}
1533 \testallgroups{DejaVuSerif-TLF}% {5}
1534 \testallgroups{DejaVuSerifCondensed-TLF}% {5}
1535 \testallgroups{GilliusADF-LF}% {5}
1536 \testallgroups{charssil-TLF}% {5} missing should be 5
1537 \testallgroups{GilliusADFCond-LF}% {5}
1538 \testallgroups{GilliusADFNoTwo-LF}% {5}
1539 \testallgroups{GilliusADFNoTwoCond-LF}% {5}
1540 \testallgroups{Lbstr-LF}% {5} name change and should be 7
1541 \testallgroups{OldStandard-TLF}% {5}
1542 \testallgroups{PlyfrDisplay-LF}% {5} name change
1543 \testallgroups{PlyfrDisplay-0sF}% {5} name change
1544 \testallgroups{TheanoDidot-TLF}% {5}
1545 \testallgroups{TheanoDidot-T0sF}% {5}
1546 \testallgroups{TheanoModern-TLF}% {5}
1547 \testallgroups{TheanoModern-T0sF}% {5}
1548 \testallgroups{TheanoOldStyle-TLF}% {5}
1549 \testallgroups{TheanoOldStyle-T0sF}% {5}
1550 \testallgroups{Crimson-TLF}% {6}

```



```

1551 \testallgroups{IBMPlexMono-TLF}% {6} now 3
1552 \testallgroups{IBMPlexSerif-TLF}% {6} now 3
1553 \testallgroups{LibertinusMono-TLF}% {6} should be 8
1554 \testallgroups{LibertinusSerifDisplay-LF}% {6}
1555 \testallgroups{LinuxLibertineDisplayT-LF}% {6}
1556 \testallgroups{LinuxLibertineMonoT-LF}% {6}
1557 \testallgroups{LinuxLibertineMonoT-TLF}% {6}
1558 \testallgroups{Ovrlck-LF}% {6} name changed
1559 \testallgroups{CormorantGaramond-LF}% {7}
1560 \testallgroups{Heuristica-TLF}% {7}
1561 \testallgroups{Heuristica-TOfF}% {7}
1562 \testallgroups{IMFELLEnglish-TLF}% {7}
1563 \testallgroups{LibreBskvl-LF}% {7} %% wrong name LibreBaskerville-TLF
1564 \testallgroups{LibreCsln-LF}% {7} changed from LibreCaslon-
1565 \testallgroups{Mrcls-LF}% {7} %% wrong name Marcellus-LF
1566 \testallgroups{NotoSans-LF}% {7}
1567 \testallgroups{NotoSansMono-TLF}% {7} now 2
1568 \testallgroups{NotoSansMono-TOfF}% {7} now 2
1569 \testallgroups{NotoSerif-LF}% {7}
1570 \testallgroups{Quattro-LF}% {7} changed from Quattrocento-
1571 \testallgroups{QuattroSans-LF}% {7} changed from QuattrocentoSans-
1572 \testallgroups{XCharter-TLF}% {7} now 2
1573 \testallgroups{XCharter-TOfF}% {7} now 2
1574 \testallgroups{erewhon-LF}% {7} now 2
1575 \testallgroups{ComicNeue-TLF}% {7}
1576 \testallgroups{ComicNeueAngular-TLF}% {7}
1577 \testallgroups{Frm-LF}% {7} % the superiors are missing; name changed from Forum-LF
1578 \testallgroups{Cochineal-TLF}% {8} now 5
1579 \testallgroups{AlgolRevived-TLF}% {9}
1580 }

```

There interaction with the user.

```

1581 \typeout{^^J=====}
1582 \typeout{[ Enter font family to check (or <enter> for kernel defined families)}
1583 \typeout{=====}
1584 \typein[\FontFamilyToCheck]{}

1585 \if!\FontFamilyToCheck!
1586 \typeout{=====}
1587 \typeout{[ Detailed output? (default no)}
1588 \typeout{=====}
1589 \typein[\Details]{}
1590 \if!\Details!
1591 \def\typeoutdetails#1{}
1592 \else
1593 \let\typeoutdetails\typeout
1594 \fi
1595 \testallkerneldefinedfamilies
1596 \else
1597 \let\typeoutdetails\typeout
1598 \testallgroups\FontFamilyToCheck
1599 \fi

1600 \stop
1601 </TS1check>

```

File 34

ltpageno.dtx

1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{<foo>}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
1 <*2ekernel>
2 \message{page nos.,}

3 \countdef\c@page=0 \c@page=1
4 \def\c1@page{}
5 \def\pagenumbering#1{%
6   \global\c@page \c@one \gdef\thepage{\csname @#1\endcsname
7     \c@page}}
8 </2ekernel>
```

File 35

ltxref.dtx

1 Cross Referencing

The user writes `\label{foo}` to define the following cross-references:

`\ref*{foo}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem, footnote and enumeration counters and other counters stepped with `\refstepcounter` are referenceable.)

`\pageref*{foo}`: page number at which `\label{foo}` command appeared. where `foo` can be any string of characters not containing `\`, `{` or `}`.

Note: The scope of the `\label` command is delimited by environments, so `\begin{theorem} \label{foo} ... \end{theorem} \label{bar}` defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

Note: the starred versions `\ref*` and `\pageref*` are provided to align with the use of `hyperref`. Without `hyperref` (or some other package using the starred form) the star is simply ignored.

Note: starting with 2023-06-01 `\label` stores also the current value of `\@currentlabelname` which should typically contain a (sanitized) title. (A reference command `\nameref` is provided by the `nameref` package.) `\label` also stores `\@currentHref` which if set should refer to a target name for links. This value is set and used by `hyperref`. Unlike the other values `\@currentHref` should be set globally. A fifth value `\@kernel@reserved@label@data` is reserved for the kernel to allow future extensions of the cross-reference system.

1.1 Cross Referencing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1 <*2ekernel>
2 \message{x-ref,}
```

This is implemented as follows. A referenceable counter `CNT` is incremented by the command `\refstepcounter{CNT}`, which sets `\@currentlabel == {CNT}{eval(\p@cnt\theCNT)}`. The command `\label{FOO}` then writes the following on file `\@auxout`:

```
\newlabel{FOO}{eval(\@currentlabel)}{eval(\thepage)}%
```

```
{eval(\@currentlabelname)}{eval(\@currentHref)}{eval(\@kernel@reserved@label@data)}
```

```
\ref{FOO} ==
BEGIN
  if \r@foo undefined
  then @refundefined := G T
  ??
  Warning: 'reference foo on page ... undefined'
```

```

        else \@car \eval(\r@F00)\@nil
      fi
    END

\pageref{foo} =
  BEGIN
    if \r@foo undefined
    then @refundundefined := G T
      ??
      Warning: 'reference foo on page ... undefined'
    else \@cdr \eval(\r@F00)\@nil
    fi
  END

```

End of historical L^AT_EX 2.09 comments.

\labelformat A reference via **\ref** produces by default the data associated with the corresponding **\label** command (typically a number); any additional formatting has to be provided by the user. If, for example, references to equations are always to be typeset as “equation (*number*)”, one has to code “**equation** (**\ref{key}**)”. With **\labelformat** there is a possibility to generate such frills automatically without resorting to low-level coding. The command takes two arguments: the first is the name of a counter and the second is its representation when referenced. This means that for a successful usage, one has to know the counter name being used for generating the label, though in practice this should not pose a problem. The current counter number is picked up as an argument. Here are two examples:

```

\labelformat{section}{section~#1}
\labelformat{equation}{equation~( #1 )}

```

\Ref A side effect of using **\labelformat** is that, depending on the defined formatting, it becomes impossible to use **\ref** at the beginning of a sentence (if its replacement text starts with a lowercase letter). To overcome this problem we introduce the command **\Ref** that behave like **\ref** except that it uppercases the first token of the generated string.

To make **\Ref** work properly the very first token in the second argument of **\labelformat** has to be a simple ASCII or UTF-8 letter, otherwise the capitalization will fail or worse, you will end up with some error messages. If you actually need something more complicated in this place (e.g., an accented letter not written as a UTF-8 character) you have to explicitly surround it with braces, to identify the part that needs to be capitalized. For example, for figure references in the Hungarian language you might want to write **\labelformat{figure}{\’a}bra~\thefigure** or use **\labelformat{figure}{ábra~\thefigure}** which avoids the brace problem.

\G@refundundefinedtrue This does not save on name-space (since **\G@refundundefinedfalse** was never needed) but **\@refundundefined** it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

Note despite its name, **\G@refundundefinedtrue** does *not* correspond to an **\if** command, and there is no matching **...false**. It would be more natural to call the command **\G@refundundefined** (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a **\ref**-like command that mimicked the definition of **\ref**, calling **\G@refundundefinedtrue**. Inspection of the T_EX archives

revealed several such packages, and so this command has been named ...true so that the definition of \ref need not be changed, and the packages will work without change.

```

3 % \newif\ifG@refundefined
4 % \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
5 % \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
6 \def\G@refundefinedtrue{%
7   \gdef\@refundefined{%
8     \@latex@warning@no@line{There were undefined references}}
9 \let\@refundefined\relax

```

(End of definition for \G@refundefinedtrue and \@refundefined.)

\ref Referencing a \label. RmS 91/10/25: added a few extra \reset@font, as suggested by
\pageref Bernd Raichle
\setref RmS 92/08/14: made \ref and \pageref robust
RmS 93/09/08: Added setting of refundefined switch.

```

10 </2ekernel>
11 <*2ekernel | latexrelease>
12 <latexrelease>\IncludeInRelease{2023/06/01}%
13 <latexrelease>          {\@kernel@sref}{store five arguments}%
14 \def\@setref#1#2#3{%
15   \ifx#1\relax
16     \protect\G@refundefinedtrue
17     \nfss@text{\reset@font\bfseries ??}%
18     \@latex@warning{Reference ‘#3’ on page \thepage \space
19       undefined}%
20   \else
21     \expandafter#2#1\@empty\@empty\@empty\@empty\null
22   \fi}

23 \long\def\@firstoffive#1#2#3#4#5{#1}
24 \long\def\@secondoffive#1#2#3#4#5{#2}
25 \def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoffive{#1}}
26 \def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
27   \@secondoffive{#1}}
28 <latexrelease>\EndIncludeInRelease
29 <latexrelease>\IncludeInRelease{2022/06/01}%
30 <latexrelease>          {\@kernel@sref}{store five arguments}%
31 <latexrelease>\def\@setref#1#2#3{%
32 <latexrelease>   \ifx#1\relax
33 <latexrelease>     \protect\G@refundefinedtrue
34 <latexrelease>     \nfss@text{\reset@font\bfseries ??}%
35 <latexrelease>     \@latex@warning{Reference ‘#3’ on page \thepage \space
36 <latexrelease>       undefined}%
37 <latexrelease>   \else
38 <latexrelease>     \expandafter#2#1\null
39 <latexrelease>   \fi}
40 <latexrelease>\let\@firstoffive\undefined
41 <latexrelease>\let\@secondoffive\undefined
42 <latexrelease>\def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
43 <latexrelease>\def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
44 <latexrelease>   \@secondoftwo{#1}}
45 <latexrelease>\EndIncludeInRelease
46 <latexrelease>\IncludeInRelease{0000/00/00}%

```

```

47 <latexrelease>          {\@kernel@sref}{store five arguments}%
48 <latexrelease>\def\@setref#1#2#3{%
49 <latexrelease>  \ifx#1\relax
50 <latexrelease>    \protect\G@refundefinedtrue
51 <latexrelease>    \nfss@text{\reset@font\bfseries ??}%
52 <latexrelease>    \@latex@warning{Reference ‘#3’ on page \thepage \space
53 <latexrelease>        undefined}%
54 <latexrelease>  \else
55 <latexrelease>    \expandafter#2#1\null
56 <latexrelease>  \fi}
57 <latexrelease>\let\@firstoffive\undefined
58 <latexrelease>\let\@secondoffive\undefined
59 <latexrelease>\let\@kernel@sref\undefined
60 <latexrelease>\let\@kernel@spageref\undefined
61 <latexrelease>\EndIncludeInRelease

62 <latexrelease>\IncludeInRelease{2022/06/01}%
63 <latexrelease>          {\ref}{Add starred reference commands}%

64 \let\@kernel@ref\@kernel@sref
65 \let\@kernel@pageref\@kernel@spageref
66 \NewDocumentCommand\ref{s}
67   {\IfBooleanTF{#1}{\@kernel@sref}{\@kernel@ref}}
68 \NewDocumentCommand\pageref{s}
69   {\IfBooleanTF{#1}{\@kernel@spageref}{\@kernel@pageref}}

```

As the commands are now protected we also need expandable versions for use in `\ifthenelse`:

```

70 \def\@kernel@pageref@exp#1{\csname cs_if_exist:cTF\endcsname
71   {r@#1}{\csname tl_item:cn\endcsname{r@#1}{2}}{0}}
72 \def\@kernel@ref@exp#1{\csname cs_if_exist:cTF\endcsname
73   {r@#1}{\csname tl_item:cn\endcsname{r@#1}{1}}{0}}

74 </2ekernel | latexrelease>
75 <latexrelease>\EndIncludeInRelease
76 <latexrelease>\IncludeInRelease{0000/00/00}%
77 <latexrelease>          {\ref}{Add starred reference commands}%
78 <latexrelease>\def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
79 <latexrelease>\def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
80 <latexrelease>          \@secondoftwo{#1}}
81 <latexrelease>
82 <latexrelease>\EndIncludeInRelease
83 <*2ekernel>

```

(End of definition for `\ref`, `\pageref`, and `\@setref`.)

`\newlabel` This command will be written to the `.aux` file to pass label information from one run to another.

`\@newl@bel` The internal form of `\newlabel` and `\bibcite`. Note that this macro does it's work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```

84 \def\@newl@bel#1#2#3{%
85   \@ifundefined{#1@#2}%
86   \relax
87   {\gdef \@multiplelabels {%

```

```

88      \@latex@warning@no@line{There were multiply-defined labels}}%
89      \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
90      \global\@namedef{#1@#2}{#3}}
91      \def\newlabel{\@newl@bel r}
92      \@onlypreamble\@newl@bel

```

(End of definition for \newlabel and \@newl@bel.)

`\ifmultiplelabels` This is redefined to produce a warning if at least one label is defined more than once. It is executed by the `\enddocument` command.

```

93      \let \multiplelabels \relax

```

(End of definition for \ifmultiplelabels and \multiplelabels.)

`\label` The commands `\label` and `\refstepcounter` have been changed to allow `\protect`’ed commands to work properly. For example,

```

\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}

```

will cause a `\label{bar}` command to define `\ref{bar}` to expand to something like `\foo{4.d}`. Change made 20 Jul 88.

```

94      </2ekernel>
95      <*2ekernel | latexrelease>
96      <latexrelease>\IncludeInRelease{2026/06/01}%
97      <latexrelease>          {\label}{disable commands while writing}%
98      \def\label#1{\@bsphack
99      \begingroup
100      \UseHookWithArguments{label}{1}{#1}%
101      \protected@write\@auxout{%

```

`\@currentlabelname` can contain a `\label` command and this can loop (gh1841). We disable the commands that are also disabled when writing `\addtocontents`.

```

102      \let\label\@gobble@om
103      \let\index\@gobble@som
104      \let\glossary\@gobble@om}%
105      {\string\newlabel{#1}{\@currentlabel}{\thepage}%
106      {\@currentlabelname}{\@currentHref}{\@kernel@reserved@label@data}}}%
107      \endgroup
108      \@esphack}
109      <latexrelease>\EndIncludeInRelease
110      <latexrelease>\IncludeInRelease{2023/06/01}%
111      <latexrelease>          {\label}{store five label arguments}%
112      \providecommand\@currentlabelname{}
113      \providecommand\@currentHref{}
114      \providecommand\@kernel@reserved@label@data{}
115      \NewHookWithArguments{label}{1}
116      <latexrelease>\def\label#1{\@bsphack
117      <latexrelease> \begingroup
118      <latexrelease> \UseHookWithArguments{label}{1}{#1}%
119      <latexrelease> \protected@write\@auxout{%
120      <latexrelease>          {\string\newlabel{#1}{\@currentlabel}{\thepage}%
121      <latexrelease>          {\@currentlabelname}{\@currentHref}{\@kernel@reserved@label@data}}}%
122      <latexrelease> \endgroup
123      <latexrelease> \esphack}
124      <latexrelease>\EndIncludeInRelease

```

```

125 <latexrelease>\IncludeInRelease{0000/00/00}%
126 <latexrelease>          {\label}{store five label arguments}%
127 <latexrelease>\def\label#1{\@bsphack
128 <latexrelease>  \protected@write\@auxout{}%
129 <latexrelease>          {\string\newlabel{#1}{\@currentlabel}{\thepage}}}%
130 <latexrelease>  \@esphack}
131 <latexrelease>\EndIncludeInRelease

```

(End of definition for \label. This function is documented on page 854.)

\refstepcounter Step the counter and allow for labels to point to its current value.

```

132 <latexrelease>\IncludeInRelease{2024/11/01}%
133 <latexrelease>          {\refstepcounter}{set theHcounter representation}%

```

refstepcounter (socket) This socket takes the whole code as argument. The default kernel plug is identity. By changing the plug hyperref can add a conditional and e.g. suppress the processing in a PDF context.

```

134 \NewSocket{refstepcounter}{1}

```

refstepcounter/target (socket) This socket takes an argument, the counter name, and should at least set from it the target name \@currentHref. With hyperref it sets also the actual target. This is done with a socket so that the target name is not set more than once to (possibly) different names. The socket is not used in \@kernel@refstepcounter. The tagging code needs the target name so it is added after this socket.

```

135 \NewSocket{refstepcounter/target}{1}

```

(refstepcounter/target) (plug)

```

136 \NewSocketPlug{refstepcounter/target}{kernel}
137 {\xdef\@currentHref {#1.\curname theH#1\endcurname}}%
138 \AssignSocketPlug{refstepcounter/target}{kernel}

```

A default definition for \@currentcounter.

```

\@currentcounter 139 \def\@currentcounter{}

```

\ltx@star@counter Saved * for testing the argument of \refstepcounter. This was implemented as part of a hotfix which is why we have it in 2024/11 rollback.

```

140 \def\ltx@star@counter{*}

141 \def\refstepcounter#1{%
142   \UseSocket{refstepcounter}{%
143     \stepcounter{#1}%
144     \edef\reserved@a{#1}%
145     \ifx\reserved@a\ltx@star@counter\else
146       \let\@currentcounter\reserved@a
147     \fi
148     \protected@edef\@currentlabel

```

By generating the second csname first the \p@... command can grab it as an argument which can be helpful for more complicated typesetting arrangements.

The trick is to ensure that \curname the#1\endcurname is turned into a single token before \p@... is expanded further. This way, if the \p@... command is a macro with one argument it will receive \the.... With the original kernel code (i.e., without the \expandafter) it will instead pick up \curname which would be disastrous.

Using `\expandafter` instead of braces delimiting the argument is better because, assuming that the `\p@...` command is not defined as a macro with one argument, the braces will stay and prohibit kerning that might otherwise happen between the glyphs generated by `\the...` and surrounding glyphs.

```

149         {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
150         \UseSocket{refstepcounter/target}{#1}%
151         \UseTaggingSocket{recordtarget}%
152     }%
153 }

```

This is a version of `\refstepcounter` which does not set and use targets.

```

\@kernel@refstepcounter
154 \def\@kernel@refstepcounter#1{%
155     \UseSocket{refstepcounter}{%
156         \stepcounter{#1}%
157         \edef\reserved@a{#1}%
158         \ifx\reserved@a\ltx@star@counter\else
159             \let\@currentcounter\reserved@a
160         \fi
161         \protected@edef\@currentlabel
162             {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}}}%

163 \<latexrelease>\EndIncludeInRelease

164 \<latexrelease>\IncludeInRelease{2022/06/01}%
165 \<latexrelease>         {\refstepcounter}{set theHcounter representation}%
166 \<latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
167 \<latexrelease>         \edef\@currentcounter{#1}%
168 \<latexrelease>         \protected@edef\@currentlabel
169 \<latexrelease>         {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
170 \<latexrelease>}}
171 \<latexrelease>\let\@kernel@refstepcounter\refstepcounter
172 \<latexrelease>\let\ltx@star@counter\@undefined
173 \<latexrelease>\EndIncludeInRelease

174 \<latexrelease>\IncludeInRelease{2020/10/01}%
175 \<latexrelease>         {\refstepcounter}{Add starred version}%
176 \<latexrelease>\def\@currentcounter{}
177 \<latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
178 \<latexrelease>         \edef\@currentcounter{#1}%
179 \<latexrelease>         \protected@edef\@currentlabel
180 \<latexrelease>         {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
181 \<latexrelease>}}
182 \<latexrelease>\EndIncludeInRelease

183 \<latexrelease>\IncludeInRelease{2019/10/01}%
184 \<latexrelease>         {\refstepcounter}{Support for \labelformat}%
185 \<latexrelease>\let\@currentcounter\@undefined
186 \<latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
187 \<latexrelease>         \protected@edef\@currentlabel
188 \<latexrelease>         {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
189 \<latexrelease>}}
190 \<latexrelease>\EndIncludeInRelease

191 \<latexrelease>\IncludeInRelease{0000/00/00}%
192 \<latexrelease>         {\refstepcounter}{Original from 2.09}%
193 \<latexrelease>\def\refstepcounter#1{\stepcounter{#1}%

```

```

194 <latexrelease> \protected@edef\@currentlabel
195 <latexrelease> { \csname p@#1\endcsname\csname the#1\endcsname}%
196 <latexrelease> }
197 <latexrelease> \EndIncludeInRelease

(End of definition for \refstepcounter and others.)

198 <latexrelease> \IncludeInRelease{2022/06/01}%
199 <latexrelease> { \Ref}{Add starred version}%

\labelformat A shortcut to set the \p@... macro for a counter. It will pick up the counter representation as an argument so that it can be specially formatted.

200 \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}

(End of definition for \labelformat.)

\Ref This macro expands the result of \ref and then uppercases the first token. Only useful if the label was generated via \labelformat and contains some lower case letter at its start. If the label starts with a complicated construct (e.g., an accented letter that is provided via a command, e.g., \textit{a} instead of a UTF-8 character like ä) one has to surround everything that needs uppercasing in a brace group in the definition of \labelformat.40

201 % \changes{v1.1s}{2024/12/10}{Replace \cs{tempa} with \cs{reserved@a} (gh/1579)}
202 \def\@kernel@Ref#1{\protected@edef\reserved@a{\@kernel@ref{#1}}%
203 \expandafter\MakeUppercase\reserved@a}
204 \def\@kernel@sRef#1{\protected@edef\reserved@a{\@kernel@sref{#1}}%
205 \expandafter\MakeUppercase\reserved@a}
206 \NewDocumentCommand\Ref{s}
207 { \IfBooleanTF{#1}{\@kernel@sRef}{\@kernel@Ref}}

(End of definition for \Ref.)

208 </2ekernel | latexrelease>
209 <latexrelease> \EndIncludeInRelease
210 <latexrelease> \IncludeInRelease{2020/10/01}%
211 <latexrelease> { \Ref}{Add starred version}%
212 <latexrelease> \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
213 <latexrelease> \DeclareRobustCommand\Ref[1]{\protected@edef\reserved@a{\ref{#1}}%
214 <latexrelease> \expandafter\MakeUppercase\reserved@a}
215 <latexrelease> \EndIncludeInRelease

216 <latexrelease> \IncludeInRelease{2019/10/01}%
217 <latexrelease> { \Ref}{Add \labelformat and \Ref}%
218 <latexrelease> \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
219 <latexrelease> \DeclareRobustCommand\Ref[1]{\protected@edef\reserved@a{\ref{#1}}%
220 <latexrelease> \expandafter\MakeUppercase\reserved@a}
221 <latexrelease> \EndIncludeInRelease

222 <latexrelease> \IncludeInRelease{0000/00/00}%
223 <latexrelease> { \Ref}{Add \labelformat and \Ref}%
224 <latexrelease>
225 <latexrelease> \let\labelformat\@undefined
226 <latexrelease> \let\Ref\@undefined
227 <latexrelease>
228 <latexrelease> \EndIncludeInRelease
229 <*2ekernel>

```

⁴⁰There is one problem with this approach: the braces are kept in a normal \ref which might spoil kerning. Perhaps one day this needs redoing.

`\@currentlabel` Default for `\label` commands that come before any environment.

```

230 \def\@currentlabel{}

(End of definition for \@currentlabel.)

231 \endkernel

```

File 36

ltproperties.dtx

Abstract

This code implements command to record and (expandably) reference document properties. It extends the standard `\label/\ref/\pageref` commands.

1 Introduction

The module allows to record the “current state” of various document properties (typically the content of macros and values of counters) and to access them in other places through a label. The list of properties that can be recorded and retrieved are not fix and can be extended by the user. The values of the properties are recorded in the `.aux` file and can be retrieved at the second compilation.

The module uses the ideas of properties and labels. A label is a document reference point: a name for the user. An property is something that L^AT_EX can track, such as a page number, section number or name. The names of labels and properties may be arbitrary. Note that there is a single namespace for each.

2 Design discussion

The design here largely follows ideas from `zref`. In particular, there are two independent concepts: properties that can be recorded between runs, and labels which consist of lists of these properties. The reason for the split is that individual labels will want to record some but not all properties. For examples, a label concerned with position would track the x and y coordinates of the current point, but not for example the page number.

In the current implementation, properties share a single namespace. This allows multiple lists to re-use the same properties, for example page number, absolute page number, etc. This does mean that *changing* a standard property is an issue. However, some properties have complex definitions (again, see `zref` at present): having them in a single shared space avoids the need to copy code.

Labels could be implemented as `prop` data. That is not done at present as there is no obvious need to map to or copy the data. As such, faster performance is available using a hash table approach as in a “classical” set up. Data written to the `.aux` file uses simple paired *balanced text* not keyvals: this avoids any restrictions on names and again offers increased performance.

The `expl3` versions of the label command do not use `\@bsphack/\@esphack` to avoid double spaces, but the L^AT_EX 2_ε command does as it lives at the document command level.

The reference commands are expandable.

Currently the code has nearly no impact on the main `\label` and `\ref` commands as too many external packages rely on the concrete implementation. There is one exception: the label names share the same namespace. That means that if both `\label{ABC}` and `\RecordProperties{ABC}{page}` are used there is a warning Label ‘ABC’ multiply defined.

3 Handling unknown labels and properties

With the standard `\label/\ref` commands the requested label is either in the `.aux`-file (and so known) or not. In the first case the stored value can be used, in the second case the reference commands print two question marks.

With flexible property lists a reference commands asks for the value of a specific property stored under a label name and we have to consider more variants:

- If the requested property is unknown (not declared) the system is not correctly set up and an error is issued.
- If the label is unknown, the default of the property is used.
- If the label is known, but doesn't provide a value for the property then again the default of the property is used.
- The command `\property_ref:nnn` allows to give a local default which is used instead of the property default in the two cases before.

4 Rerun messages

As the reference commands are expandable they can neither issue a message that the label or the label-property combination is unknown, nor can they trigger the rerun message at the end of the $\text{\LaTeX}$ run.

Where needed such messages must therefore be triggered manually. For this two commands are provided: `\property_ref_undefined_warn:` and `\property_ref_undefined_warn:nn`. See below for a description.

5 Open points

- The `xpos` and `ypos` properties require that the position is stored first but there is no (public) engine independent interface yet. Code must use `\tex_savepos:D`.

6 Code interfaces

<code>\property_new:nnnn</code>	<code>\property_new:nnnn {<property>} {<setpoint>} {<default>} {<code>}</code>
<code>\property_gset:nnnn</code>	<code>\property_gset:nnnn {<property>} {<setpoint>} {<default>} {<code>}</code>

$\text{\LaTeX} 2_{\epsilon}$ -interface: see `\NewProperty`, `\SetProperty`.

Sets the `<property>` to have the `<default>` specified, and at the `<setpoint>` (either `now` or `shipout`) to write the result of the `<code>` as part of a label. The `<code>` should be expandable. The expansion of `<code>` (the value of the property) is written to the `.aux` file and read back from there at the next compilation. Values should assume that the standard $\text{\LaTeX}$ catcode régime with `@` a letter is active then.

If the property is declared within a package it is suggested that its name is build from letters, hyphens and slashes, and is always structured as follows:
`<package-name>/<property-name>`.

<hr/>	
<code>\property_record:nN</code>	<code>\property_record:nN {<label>} <clist var></code>
<code>\property_record:nn</code>	<code>\property_record:nn {<label>} {<clist>}</code>
<code>\property_record:(nV ee)</code>	<p>LaTeX 2_ε-interface: see <code>\RecordProperties</code>.</p> <p>Writes the list of properties given by the <code><clist></code> to the <code>.aux</code> file with the <code><label></code> specified.</p>
<hr/>	
<code>\property_ref:nn *</code>	<code>\property_ref:nn {<label>} {<property>}</code>
<code>\property_ref:ee *</code>	<p>LaTeX 2_ε-interface: see <code>\RefProperty</code>.</p> <p>Expands to the value of the <code><property></code> for the <code><label></code>, if available, and the default value of the property otherwise. If <code><property></code> has not been declared with <code>\property_new:nnnn</code> an error is issued. The command raises an internal, expandable, local flag if the reference can not be resolved.</p>
<hr/>	
<code>\property_item:nn *</code>	<code>\property_item:nn {<label>} {<property>}</code>
<code>\property_item:ee *</code>	<p>Retrieves the value of the <code><property></code> for the <code><label></code> like <code>\property_ref:nn</code> but the result is returned within the <code>\unexpanded</code> primitive (<code>\exp_not:n</code>), which means that the <code><value></code> does not expand further when appearing in an <code>e</code>-type or <code>x</code>-type argument expansion. This allows for example to handle values containing user commands which are not safe in an expansion context.</p>
<hr/>	
<code>\property_ref:nnn *</code>	<code>\property_ref:nnn {<label>} {<property>} {<local default>}</code>
<code>\property_ref:een *</code>	<p>LaTeX 2_ε-interface: see <code>\RefProperty</code>.</p> <p>Expands to the value of the <code><property></code> for the <code><label></code>, if available, and to <code><local default></code> otherwise. If <code><property></code> has not been declared with <code>\property_new:nnnn</code> an error is issued. The command raises an internal, expandable local flag if the reference can not be resolved.</p>
<hr/>	
<code>\property_item:nnn *</code>	<code>\property_item:nnn {<label>} {<property>} {<local default>}</code>
<code>\property_item:een *</code>	<p>Retrieves the value of the <code><property></code> for the <code><label></code>, if available, and to <code><local default></code> otherwise, but the result is returned within the <code>\unexpanded</code> primitive (<code>\exp_not:n</code>), which means that the <code><value></code> does not expand further when appearing in an <code>e</code>-type or <code>x</code>-type argument expansion. This allows for example to handle values containing user commands which are not safe in an expansion context.</p>
<hr/>	
<code>\property_ref_undefined_warn:</code>	<p><code>\property_ref_undefined_warn:</code></p> <p>LaTeX 2_ε-interface: not provided.</p> <p>Triggers the standard warning</p> <p>LaTeX Warning: There were undefined references.</p> <p>at the end of the document if there was a recent <code>\property_ref:nn</code> or <code>\property_ref:nnn</code> which couldn't be resolved and so raised the flag. “Recent” means in the same group or in some outer group!</p>

```
\property_ref_undefined_warn:n \property_ref_undefined_warn:n {\label}}
\property_ref_undefined_warn:e
```

LaTeX 2_ε-interface: not provided.

Triggers the standard warning

LaTeX Warning: There were undefined references.

at the end of the document if $\langle label \rangle$ is not known. At the point where it is called it also issues the warning

Reference ‘ $\langle label \rangle$ ’ on page $\langle page \rangle$ undefined.

```
\property_ref_undefined_warn:nn \property_ref_undefined_warn:nn {\label}} {\property}}
\property_ref_undefined_warn:ee
```

LaTeX 2_ε-interface: see \RefUndefinedWarn.

Triggers the standard warning

LaTeX Warning: There were undefined references.

at the end of the document if the reference can not be resolved. At the point where it is called it also issues the warning

Reference ‘ $\langle label \rangle$ ’ on page $\langle page \rangle$ undefined

if the label is unknown, or the more specific

Property ‘ $\langle property \rangle$ ’ undefined for reference ‘ $\langle label \rangle$ ’ on page $\langle page \rangle$
if the label is known but doesn’t provide a value for the requested property.

```
\property_if_exist_p:n * \property_if_exist_p:n {\property}}
\property_if_exist_p:e * \property_if_exist:nTF {\property}} {\true code}} {\false code}}
\property_if_exist:nTF * LaTeX 2ε-interface: \IfPropertyExistsTF.
\property_if_exist:eTF * Tests if the \property has been declared.
```

```
\property_if_recorded_p:n * \property_if_recorded_p:n {\label}}
\property_if_recorded_p:e * \property_if_recorded:nTF {\label}} {\true code}} {\false code}}
\property_if_recorded:nTF *
\property_if_recorded:eTF *
```

LaTeX 2_ε-interface: \IfLabelExistsTF

Tests if the $\langle label \rangle$ is known. This is also true if the label has been set with the standard \label command.

```
\property_if_recorded_p:nn * \property_if_recorded_p:nn {\label}} {\property}}
\property_if_recorded_p:ee * \property_if_recorded:nnTF {\label}} {\property}} {\true code}} {\false code}}
\property_if_recorded:nnTF *
\property_if_recorded:eeTF *
```

LaTeX 2_ε-interface: \IfPropertyRecordedTF.

Tests if the label $\langle label \rangle$ is known and if it provides a value of the $\langle property \rangle$.

7 Auxiliary file interfaces

```
\new@label@record \new@label@record {\label}} {\data}}
```

This is a command only for use in the .aux file. It loads the key–value list of $\langle data \rangle$ to be available for the $\langle label \rangle$.

8 L^AT_EX 2_ε interface

The L^AT_EX interfaces always expand label and property arguments. This means that one must be careful when using active chars or commands in the names. UTF8-chars are protected and should be safe, similar most babel shorthands.

<code>\NewProperty</code>	<code>\NewProperty {⟨property⟩} {⟨setpoint⟩} {⟨default⟩} {⟨code⟩}</code>
<code>\SetProperty</code>	<code>\SetProperty {⟨property⟩} {⟨setpoint⟩} {⟨default⟩} {⟨code⟩}</code>

Sets the `⟨property⟩` to have the `⟨default⟩` specified, and at the `⟨setpoint⟩` (either `now` or `shipout`) to write the result of the `⟨code⟩` as part of a label. The `⟨code⟩` should be expandable. The expansion of `⟨code⟩` (the value of the property) is written to the `.aux` file and read back from there at the next compilation (at which point normally the standard L^AT_EX catcode régime with `@` a letter is active).

<code>\RecordProperties</code>	<code>\RecordProperties {⟨label⟩} {⟨clist⟩}</code>
--------------------------------	--

Writes the list of properties given by the `⟨clist⟩` to the `.aux` file with the `⟨label⟩` specified. Similar to the standard `\label` command the arguments are expanded. So `⟨clist⟩` can be a macro containing a list of properties. Also similar to the standard `\label` command, the command is surrounded by an `\@bsphack/\@esphack` pair to preserve spacing.

<code>\RefProperty *</code>	<code>\RefProperty [⟨local default⟩] {⟨label⟩} {⟨property⟩}</code>
-----------------------------	--

Expands to the value of the `⟨property⟩` for the `⟨label⟩`, if available, and the default value of the property or – if given – to `⟨local default⟩` otherwise. If `{⟨property⟩}` has not been declared an error is issued.

<code>\IfPropertyExistsTF</code>	<code>\IfPropertyExistsTF {⟨property⟩} {⟨true code⟩} {⟨false code⟩}</code>
<code>\IfPropertyExistsT</code>	
<code>\IfPropertyExistsF</code>	Tests if the <code>⟨property⟩</code> has been declared.

<code>\IfLabelExistsTF</code>	<code>\IfLabelExistsTF {⟨label⟩} {⟨true code⟩} {⟨false code⟩}</code>
-------------------------------	--

<code>\IfLabelExistsT</code>	
<code>\IfLabelExistsF</code>	Tests if the <code>⟨label⟩</code> has been recorded. This is also true if a label has been set with the standard <code>\label</code> command.

<code>\IfPropertyRecordedTF</code>	<code>\IfPropertyRecordedTF {⟨label⟩} {⟨property⟩} {⟨true code⟩} {⟨false code⟩}</code>
------------------------------------	--

<code>\IfPropertyRecordedT</code>	
<code>\IfPropertyRecordedF</code>	Tests if the label and a value of the <code>⟨property⟩</code> for the <code>⟨label⟩</code> are both known.

<code>\RefUndefinedWarn</code>	<code>\RefUndefinedWarn {⟨label⟩} {⟨property⟩}</code>
--------------------------------	---

Triggers the standard warning

LaTeX Warning: There were undefined references.
at the end of the document if the reference for `⟨label⟩` and `⟨property⟩` can not be resolved. At the point where it is called it also issues the warning

Reference ‘`⟨label⟩`’ on page `⟨page⟩` undefined
if the label is unknown, or the more specific

Property ‘`⟨property⟩`’ undefined for reference ‘`⟨label⟩`’ on page `⟨page⟩` if the label is known but doesn’t provide a value for the requested property.

9 Pre-declared properties

<u>abspage</u>	(shipout) The absolute value of the current page: starts at 1 and increases monotonically at each shipout.
<u>page</u>	(shipout) The current page as given by <code>\thepage</code> : this may or may not be a numerical value, depending on the current style. Contrast with <code>\abspage</code> . You get this value also with the standard <code>\label/\pageref</code> .
<u>pagenum</u>	(shipout) The current page as arabic number. This is suitable for integer operations and comparisons.
<u>label</u>	(now) The content of <code>\@currentlabel</code> . This is the value that you get also with the standard <code>\label/\ref</code> .
<u>title</u>	(now) The content of <code>\@currentlabelname</code> . This command is filled beside others by the <code>nameref</code> package and some classes (e.g. <code>memoir</code>).
<u>target</u>	(now) The content of <code>\@currentHref</code> . This command is normally filled by for example <code>hyperref</code> and gives the name of the last destination it created.
<u>pagetarget</u>	(shipout) The content of <code>\@currentHpage</code> . This command is filled for example by a recent version of <code>hyperref</code> and then gives the name of the last page destination it created.
<u>counter</u>	(now) The content of <code>\@currentcounter</code> . This command contains after a <code>\refstepcounter</code> the name of the counter.
<u>xpos</u> <u>ypos</u>	(shipout) This stores the x and y coordinates of a point previously stored with <code>\pdfsavepos/\savepos</code> . E.g. (if <code>bidi</code> is used it can be necessary to save the position before and after the label): <pre>\tex_savepos:D \property_record:nn{myposition}{xpos,ypos} \tex_savepos:D</pre>

10 The Implementation

```

1 <*2ekernel | latexrelease>
2 \ExplSyntaxOn
3 <@@=property>
4 <latexrelease>\NewModuleRelease{2023/11/01}{ltproperties}
5 <latexrelease> {Cross-referencing~properties}

```

The approach here is based closely on that from `zref`; separate out lists of properties and the properties themselves, so the latter can be used multiple times and in varying combinations. However, not everything is a straight copy. Firstly, we treat lists of properties as simple comma lists: that allows us to have either saved or dynamic lists and to avoid another data structure. The cost is that errors are detected at point-of-use, but in any real case that should be true anyway (and is true for `\zref@labelbyprop` already). Secondly, we allow properties to have arbitrary names, as the code does not require them to tokenize as control sequences.

As properties can be reset, they are not constants. But they also have various pieces of required data. So we use the same approach as `color` and make them declarations. Data-wise, we need the detail of the implementation, the default and a flag to show if the code works now or at shipout. This last entry is done using text so needs a check. We could use a set of `prop` here, but as we never need to map or copy the lists, we can gain performance using the hash table approach.

```

6 \cs_new_protected:Npn \property_new:nnnn #1#2#3#4
7 {
8   \cs_if_free:cTF { __property_code_ #1 : }
9   {
10     \exp_args:Ne \__property_gset:nnnn { \tl_to_str:n {#1} }
11     {#2} {#3} {#4}
12   }
13   {
14     \msg_error:nn { property }{ exists }{#1}
15   }
16 }
17 \cs_new_protected:Npn \property_gset:nnnn #1#2#3#4
18 {
19   \__property_gset:ennn { \tl_to_str:n {#1} }
20   {#2} {#3} {#4}
21 }
22 \cs_new_protected:Npn \__property_gset:nnnn #1#2#3#4
23 {
24   \cs_gset:cpn { __property_code_ #1 : } {#4}
25   \tl_gclear_new:c { g__property_default_ #1 _tl }
26   \tl_gset:cn { g__property_default_ #1 _tl } {#3}
27   \bool_if_exist:cF { g__property_shipout_ #1 _bool }
28   { \bool_new:c { g__property_shipout_ #1 _bool } }
29   \str_case:nnF {#2}
30   {
31     { now } { { \bool_gset_false:c { g__property_shipout_ #1 _bool } } }
32     { shipout }
33     { \bool_gset_true:c { g__property_shipout_ #1 _bool } }
34   }
35   { \msg_error:nnnn { property } { unknown-setpoint } {#1} {#2} }

```

```

36 }
37 \cs_generate_variant:Nn \__property_gset:nnnn {ennn}

```

(End of definition for `\property_new:nnnn`, `\property_gset:nnnn`, and `\__property_gset:nnnn`.
These functions are documented on page 850.)

\NewProperty For consistency we expand the property name, but this doesn't warrant a variant of the
\SetProperty L3-commands.

```

38 \cs_new_protected:Npn \NewProperty #1#2#3#4
39 {
40   \protected@edef\reserved@a{#1}
41   \exp_args:No \property_new:nnnn {\reserved@a} {#2}{#3}{#4}
42 }
43 \cs_new_protected:Npn \SetProperty #1#2#3#4
44 {
45   \protected@edef\reserved@a{#1}
46   \exp_args:No \property_gset:nnnn {\reserved@a} {#2}{#3}{#4}
47 }

```

(End of definition for `\NewProperty` and `\SetProperty`. These functions are documented on page 853.)

```

\property_record:nN Writing data when it is labelled means expanding at this stage and possibly later too.
\property_record:nn That is all pretty easy using expl3: we accept a stray comma at the end of the list as
\property_record:nV that is easier to deal with than trying to tidy up, and there is no real downside.
\property_record:ee
\property_record:oo
\__property_record:nn
\__property_record:en
\__property_record_value:n
  \_property_record_value_aux:n
    \_property_record_value_aux:e
48 \cs_new_protected:Npn \property_record:nN #1#2
49 { \property_record:nV {#1} #2 }
50 \cs_new_protected:Npn \property_record:nn #1#2
51 { \__property_record:en { \tl_to_str:n {#1} } {#2} }
52 \cs_generate_variant:Nn \property_record:nn { nV , ee, oo }
53 \cs_new_protected:Npn \__property_record:nn #1#2
54 {
55   \protected@write \@auxout {}
56   {
57     \token_to_str:N \new@label@record
58     {#1}
59     { \clist_map_function:nN {#2} \__property_record_value:n }
60   }
61 }
62 \cs_generate_variant:Nn \__property_record:nn { e }
63 \cs_new:Npn \__property_record_value:n #1
64 { \__property_record_value_aux:e { \tl_to_str:n {#1} } }
65 \cs_new:Npn \__property_record_value_aux:n #1
66 {
67   \cs_if_exist:cTF { __property_code_ #1 : }
68   {
69     {#1}
70     {
71       \bool_if:cTF { g__property_shipout_ #1 _bool }
72       { \exp_not:c }
73       { \use:c }
74       { __property_code_ #1 : }
75     }
76   }
77   { \msg_expandable_error:nnn { property } { not-declared } {#1} }

```

```

78 }
79 \cs_generate_variant:Nn \__property_record_value_aux:n { e }

```

(End of definition for `\property_record:nn` and others. These functions are documented on page 851.)

`\RecordProperties`

```

80 \NewDocumentCommand\RecordProperties { m m }
81 {
82   \@bsphack
83   \protected@edef\reserved@a{#1}
84   \protected@edef\reserved@b{#2}
85   \property_record:oo {\reserved@a}{\reserved@b}
86   \@esphack
87 }

```

(End of definition for `\RecordProperties`. This function is documented on page 853.)

10.1 Reference commands

`l__property_ref_flag` A flag that is set if a reference couldn't be resolved.

```

88 \flag_new:n { l__property_ref_flag }

```

(End of definition for `l__property_ref_flag`.)

`\property_ref:nn` Search for the label/property combination, and if not found fall back to the default of the property.

`\property_ref:ee`

```

89 \cs_new:Npn \property_ref:nn #1#2
90 {
91   \__property_ref:een
92   { \tl_to_str:n {#1} }
93   { \tl_to_str:n {#2} }
94   { \tl_use:c { g__property_default_ #2 _tl } }
95 }
96 \cs_generate_variant:Nn \property_ref:nn { ee}

```

(End of definition for `\property_ref:nn`. This function is documented on page 851.)

`\property_ref:nnn` This allows to set a local default value which overrides the default value of the property.

`\property_ref:een`

`\__property_ref:nnn`

`\__property_ref:een`

```

97 \cs_new:Npn \property_ref:nnn #1#2#3
98 {
99   \__property_ref:een
100   { \tl_to_str:n {#1} }
101   { \tl_to_str:n {#2} }
102   {#3}
103 }
104 \cs_new:Npn \__property_ref:nnn #1#2#3
105 {
106   \tl_if_exist:cTF { g__property_label_ #1 _ #2 _tl }
107   { \tl_use:c { g__property_label_ #1 _ #2 _tl } }
108   {
109     \flag_if_raised:nF
110     { l__property_ref_flag } { \flag_raise:n { l__property_ref_flag } }

```

We test for the default of the property only to check if the property has been declared.

```

111     \tl_if_exist:cTF { g__property_default_ #2 _tl }
112     { #3 }
113     { \msg_expandable_error:nnn { property } { not-declared } {#2} }
114   }
115 }
116 \cs_generate_variant:Nn \__property_ref:nnn { ee }
117 \cs_generate_variant:Nn \property_ref:nnn { een}

```

(End of definition for \property_ref:nnn and __property_ref:nnn. This function is documented on page 851.)

\property_item:nn Retrieve the value but wrap it into an \exp_not:n to avoid further expansion.

```

\property_item:ee
118 \cs_new:Npn \property_item:nn #1#2
119 {
120   \tl_if_exist:cTF { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
121   {
122     \exp_not:v { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
123   }
124   { \tl_use:c { g__property_default_ \tl_to_str:n {#2} _tl } }
125 }
126 \cs_generate_variant:Nn \property_item:nn { ee}

```

(End of definition for \property_item:nn. This function is documented on page 851.)

\property_item:nnn Same as \property_item:nn but allows setting a local default.

```

\property_item:een
127 \cs_new:Npn \property_item:nnn #1#2#3
128 {
129   \tl_if_exist:cTF { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
130   {
131     \exp_not:v { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
132   }
133   {
134     \tl_if_exist:cTF { g__property_default_ \tl_to_str:n {#2} _tl }
135     { #3 }
136     { \msg_expandable_error:nnn { property } { not-declared } {#2} }
137   }
138 }
139 \cs_generate_variant:Nn \property_item:nnn { ee}

```

(End of definition for \property_item:nnn. This function is documented on page 851.)

\RefProperty Search for the label/property combination, and if not found fall back to the default of the property or the given default.

```

140 \NewExpandableDocumentCommand \RefProperty { o m m }
141 {
142   \IfNoValueTF {#1}
143   {
144     \property_ref:ee {#2}{#3}
145   }
146   {
147     \property_ref:een {#2}{#3}{#1}
148   }
149 }

```

(End of definition for \RefProperty. This function is documented on page 853.)

```

\new@label@record A standard recursion loop.
\__property_data:nnn
150 \cs_new_protected:Npn \new@label@record #1#2
151 {
152   \tl_if_exist:cTF { r@#1 }
153   {
154     \gdef \@multiplelabels
155     { \@latex@warning@no@line { There-were-multiply-defined-labels } }
156     \@latex@warning@no@line { Label-‘#1’-multiply-defined }
157   }
158   {
159     \tl_new:c { r@#1 }
160     \tl_gset:cn { r@#1 }{#2}
161   }
162   \__property_data:nnn {#1} #2 { \q_recursion_tail } { ? } \q_recursion_stop
163 }
164 \cs_new_protected:Npn \__property_data:nnn #1#2#3
165 {
166   \quark_if_recursion_tail_stop:n {#2}
167   \tl_gclear_new:c { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
168   \tl_gset:cn { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl } {#3}
169   \__property_data:nnn {#1}
170 }

```

This command is used in \enddocument to test if some label values have changed.

```

171 \cs_new_protected:Npn \@kernel@new@label@record@testdef #1 #2
172 {
173   \tl_if_eq:cnF { r@#1 } {#2}
174   { \@tempwattrue }
175 }

```

(End of definition for \new@label@record and __property_data:nnn. This function is documented on page 852.)

10.2 Tests and warnings

```

\property_if_exist_p:n Tests if property has been declared.
\property_if_exist:nTF
176 \prg_new_conditional:Npnn \property_if_exist:n #1 { p , T , F , TF }
177 % #1 property
178 {
179   \cs_if_exist:cTF { __property_code_ #1 : }
180   {
181     \prg_return_true:
182   }
183   {
184     \prg_return_false:
185   }
186 }
187 \prg_generate_conditional_variant:Nnn \property_if_exist:n {e} { p , T , F , TF }

```

(End of definition for \property_if_exist:nTF. This function is documented on page 852.)

\IfPropertyExistsTF
\IfPropertyExistsT
\IfPropertyExistsF

```
188 \cs_new_eq:NN \IfPropertyExistsTF \property_if_exist:eTF
189 \cs_new:Npn \IfPropertyExistsT #1#2 {\property_if_exist:eTF {#1}{#2}{}}
190 \cs_new:Npn \IfPropertyExistsF #1 {\property_if_exist:eTF {#1}{}}
```

(End of definition for \IfPropertyExistsTF, \IfPropertyExistsT, and \IfPropertyExistsF. These functions are documented on page 853.)

\property_if_recorded_p:n
\property_if_recorded:nTF

Tests if the label has been set. This can then be used to setup e.g. rerun messages.

```
191 \prg_new_conditional:Npnn \property_if_recorded:n #1 { p , T , F , TF }
192   % #1 label
193   {
194     \tl_if_exist:cTF { r@#1 }
195     {
196       \prg_return_true:
197     }
198     {
199       \prg_return_false:
200     }
201   }
202 \prg_generate_conditional_variant:Nnn \property_if_recorded:n {e} { p , T , F , TF }
```

(End of definition for \property_if_recorded:nTF. This function is documented on page 852.)

\IfLabelExistsTF
\IfLabelExistsT
\IfLabelExistsF

```
203 \cs_new_eq:NN \IfLabelExistsTF \property_if_recorded:eTF
204 \cs_new:Npn \IfLabelExistsT #1#2 {\property_if_recorded:eTF {#1}{#2}{}}
205 \cs_new:Npn \IfLabelExistsF #1 {\property_if_recorded:eTF {#1}{}}
```

(End of definition for \IfLabelExistsTF, \IfLabelExistsT, and \IfLabelExistsF. These functions are documented on page 853.)

\property_if_recorded_p:nn
\property_if_recorded:nnTF

tests if the label/property combination has been set This can then be used to setup e.g. rerun messages.

```
206 \prg_new_conditional:Npnn \property_if_recorded:nn #1#2 { p , T , F , TF }
207   % #1 label #2 property
208   {
209     \tl_if_exist:cTF { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
210     {
211       \prg_return_true:
212     }
213     {
214       \prg_return_false:
215     }
216   }
217 \prg_generate_conditional_variant:Nnn \property_if_recorded:nn {ee} { p , T , F , TF }
```

(End of definition for \property_if_recorded:nnTF. This function is documented on page 852.)

\IfPropertyRecordedTF
\IfPropertyRecordedT
\IfPropertyRecordedF

```
218 \cs_new_eq:NN \IfPropertyRecordedTF \property_if_recorded:eeTF
219 \cs_new:Npn \IfPropertyRecordedT #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{#3}{}}
220 \cs_new:Npn \IfPropertyRecordedF #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{#3}{}}
```

(End of definition for \IfPropertyRecordedTF, \IfPropertyRecordedT, and \IfPropertyRecordedF. These functions are documented on page 853.)

`\property_ref_undefined_warn:` `\G@refundefinedtrue` is defined in `ltxref` and redefines a warning message.

```
221 \cs_new_protected:Npn \property_ref_undefined_warn:
222 {
223   \flag_if_raised:nT { l__property_ref_flag }
224   {
225     \G@refundefinedtrue
226   }
227 }
```

(End of definition for `\property_ref_undefined_warn:`. This function is documented on page 851.)

`\property_ref_undefined_warn:n`

```
\property_ref_undefined_warn:e
228 \cs_new_protected:Npn \property_ref_undefined_warn:n #1 %#1 label
229 {
230   \property_if_recorded:nF {#1}
231   {
232     \G@refundefinedtrue
233     \@latex@warning{Reference~‘#1’~on~page~\thepage\space undefined}%
234   }
235 }
236 \cs_generate_variant:Nn \property_ref_undefined_warn:n {e}
```

(End of definition for `\property_ref_undefined_warn:n`. This function is documented on page 852.)

`\property_ref_undefined_warn:nn`

```
\property_ref_undefined_warn:ee
\RefUndefinedWarn
237 \cs_new_protected:Npn \property_ref_undefined_warn:nn #1#2 %#1 label, #2 property
238 {
239   \property_if_recorded:nTF {#1}
240   {
241     \property_if_recorded:nnF {#1}{#2}
242     {
243       \G@refundefinedtrue
244       \@latex@warning
245       { Property~‘#2’~undefined~for~reference~‘#1’~on~page~\thepage }
246     }
247   }
248   {
249     \G@refundefinedtrue
250     \@latex@warning { Reference~‘#1’~on~page~\thepage\space undefined }%
251   }
252 }
253 \cs_generate_variant:Nn \property_ref_undefined_warn:nn {ee}
254 \cs_set_eq:NN \RefUndefinedWarn \property_ref_undefined_warn:ee
```

(End of definition for `\property_ref_undefined_warn:nn` and `\RefUndefinedWarn`. These functions are documented on page 852.)

10.3 Predeclared properties

`abspage`

```
255 \property_new:nnnn { abspage } { shipout }
256 { 0 } { \int_use:N \g_shipout_readonly_int }
```

(End of definition for `abspage`. This variable is documented on page 854.)

page

```
257 \property_new:nnnn { page } { shipout } { 0 } { \thepage }
```

(End of definition for page. This variable is documented on page 854.)

pagenum

```
258 \property_new:nnnn { pagenum } { shipout } { 0 } { \the \value { page } }
```

(End of definition for pagenum. This variable is documented on page 854.)

label

```
259 \property_new:nnnn { label } { now } { ?? } { \@currentlabel }
```

(End of definition for label. This variable is documented on page 854.)

title

```
260 \property_new:nnnn { title } { now }
```

```
261 { \exp_not:n { \textbf { ?? } } } { \@currentlabelname }
```

(End of definition for title. This variable is documented on page 854.)

target

```
262 \property_new:nnnn { target } { now } { } { \@currentHref }
```

(End of definition for target. This variable is documented on page 854.)

pagetarget

```
263 \newcommand\@currentHpage{}
```

```
264 \property_new:nnnn { pagetarget } { shipout } { } { \@currentHpage }
```

(End of definition for pagetarget. This variable is documented on page 854.)

counter

```
265 \property_new:nnnn { counter } { now } { } { \@currentcounter }
```

(End of definition for counter. This variable is documented on page 854.)

xpos

ypos

```
266 \property_new:nnnn { xpos } { shipout } { 0 } { \int_use:N \tex_lastxpos:D }
```

```
267 \property_new:nnnn { ypos } { shipout } { 0 } { \int_use:N \tex_lastypos:D }
```

(End of definition for xpos and ypos. These variables are documented on page 854.)

10.4 Messages

```

268 \msg_new:nnnn { property } { exists }
269   { Property~'#1'~ has~ already~ been~ declared. }
270   { There~ already~ exists~ a~ property~ declaration~ with~ this~
271     name.\\
272     Please~ use~ a~ different~ name~ for~ your~ property.}
273
274 \msg_new:nnnn { property } { not-declared }
275   { Property~'#1'~not-declared. }
276   {
277     LaTeX~has~been~asked~to~use~property~'#1',~but~this~
278     name~has~not~been~declared.
279   }
280 \msg_new:nnnn { property } { unknown-setpoint }
281   { Unknown~keyword~'#2'~for~setting~property~'#1'. }
282   {
283     LaTeX~has~been~asked~to~set~the~property~'#1',~but~the~keyword~
284     '#2'~is~not~one~of~the~two~known~values:~'now'~or~'shipout'.
285   }
286 %
287 <latexrelease>\IncludeInRelease{0000/00/00}{ltproperties}
288 <latexrelease>          {cross-referencing~properties~(undo)}%
289 <latexrelease>
290 <latexrelease>\let \NewProperty \@undefined
291 <latexrelease>\let \SetProperty \@undefined
292 <latexrelease>
293 <latexrelease>\let \RecordProperties \@undefined
294 <latexrelease>\let \RefProperty \@undefined
295 <latexrelease>\let \RefUndefinedWarn \@undefined
296 <latexrelease>
297 <latexrelease>\let \IfPropertyExistsTF \@undefined
298 <latexrelease>\let \IfLabelExistsTF \@undefined
299 <latexrelease>\let \IfPropertyRecordedTF \@undefined
300 <latexrelease>
301 <latexrelease>\let\new@label@record \@undefined
302 <latexrelease>\let\@kernel@new@label@record@testdef\@undefined
303 <latexrelease>\EndModuleRelease
304 \ExplSyntaxOff
305 </2ekernel | latexrelease>
306
307   Reset module prefix:
308 <@@=>

```

File 37

ltmiscen.dtx

1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘flushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 <*2ekernel>
2 \message{environments,}
```

1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@@end` is defined to be the `\end` command of T_EX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end T_EX in the middle.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\enddocument ==
BEGIN
  \@checkend{document}    %% checks for unmatched \begin
  \clearpage
  \begingroup
    if @filesw = true
    then close file @mainaux
    if G@refundefined = true
    then LaTeX Warning: 'There are undefined references.' fi
    if @multiplelabels = true
    then LaTeX Warning:
      'One or more label(s) multiply defined.'
    else
      \@setckpt {ARG1}{ARG2} == null
      \newlabel{LABEL}{VAL} ==
      BEGIN
        \reserved@a == VAL
        if def(\reserved@a) = def(\r@LABEL)
        else @tempswa := true          fi
      END
      \bibcite{LABEL}{VAL} == null
      BEGIN
        \reserved@a == VAL
        if def(\reserved@a) = def(\g@LABEL)
        else @tempswa := true          fi
      END
    fi
  fi
\endgroup
```

```

                                END
                                @tempswa := false
                                make @ a letter
                                \input \jobname.AUX
                                if @tempswa = true
                                    then LaTeX Warning: 'Label may have changed.
                                     Rerun to get cross-references right.'
                                fi
                                fi
                                fi
                                \endgroup
                                finish up
                                END

                                \@writefile{EXT}{ENTRY} ==
                                    if tf@EXT undefined
                                        else \write\tf@EXT{ENTRY}
                                    fi
                                End of historical LATEX 2.09 comments.

\currentenvir The name of the current environment. Initialized to document to so that \end{document}
works correctly.
3 \def\currentenvir{document}
(End of definition for \currentenvir.)

\if@ignore
\@ignoretrue 4 \def\@ignorefalse{\global\let\if@ignore\iffalse}
\@ignorefalse 5 \def\@ignoretrue {\global\let\if@ignore\iftrue}
6 \@ignorefalse
(End of definition for \if@ignore, \@ignoretrue, and \@ignorefalse.)

\ignorespacesafterend
7 \let\ignorespacesafterend\@ignoretrue
(End of definition for \ignorespacesafterend.)

\end{document}
8 \</2kernel>
9 \<*2kernel | latexrelease>
10 \<latexrelease>\IncludeInRelease{2023/11/01}%
11 \<latexrelease> \<\enddocument>\{check property labels}%
12 \def\enddocument{%
The \code{\end{document}} hook is executed first. If necessary it can contain a \clearpage to
output dangling floats first. In this position it can also contain something like \end{foo}
so that the whole document effectively starts and ends with some special environment.
However, this must be used with care, eg if two applications would use this without knowl-
edge of each other the order of the environments will be wrong after all. \AtEndDocument
is redefined at this point so that and such commands that get into the hook do not chase
their tail...
13 \@kernel@before@enddocument
14 \UseOneTimeHook{enddocument}%
15 \@kernel@after@enddocument

```

```

16 \checkend{document}%
17 \clearpage

18 \@kernel@before@enddocument@afterlastpage
19 \UseOneTimeHook{enddocument/afterlastpage}%
20 \@kernel@after@enddocument@afterlastpage
21 \begingroup
22   \if@files
23     \immediate\closeout\@mainaux
24     \let\@setckpt\@gobbletwo
25     \let\@newlabel\@testdef

26     \let\newlabel@record\@kernel@newlabel@record@testdef

```

The previous line is equiv to setting

```

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

We use \@input to load the .aux file, so that it doesn't show up in the list of files produced by \listfiles.

```

27 \tempswafalse
28 \global\advance\c@page\@ne
29 \makeatletter \@input\jobname.aux
30 \fi
31 \UseOneTimeHook{enddocument/afteraux}%

```

Next hook is expect to contain only code for writing info messages on the terminal.

```

32 \UseOneTimeHook{enddocument/info}%
33 \endgroup
34 \UseOneTimeHook{enddocument/end}%
35 \deadcycles\z\@end}

```

The public hooks used in \enddocument:

```

36 \NewHook{enddocument}
37 \NewHook{enddocument/afterlastpage}
38 \NewHook{enddocument/afteraux}
39 \NewHook{enddocument/info}
40 \NewHook{enddocument/end}

```

This is one of the few places where we already add data and rules to a hook already in the kernel.

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled. This would result in a harmless warning during the format generation, because in that case the code chunk label doesn't exist, and therefore can't be dropped.

```

41 \<latexrelease>\RemoveFromHook{enddocument/info}[kernel/filelist]
42 \<latexrelease>\RemoveFromHook{enddocument/info}[kernel/warnings]
43 \<latexrelease>\RemoveFromHook{enddocument/info}[kernel/release]

44 \AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
45 \AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
46 \AddToHook{enddocument/info}[kernel/release]{%
47   \let\show@release@info\wlog
48   \show@release@info{ *****}%
49   \the\LaTeXReleaseInfo

```

```

50 \show@release@info{ *****}
51
52 \DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
53 \DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}

```

(End of definition for \enddocument.)

\@enddocument@kernel@warnings

```

54 \def\@enddocument@kernel@warnings{%

```

First we check for font size substitution bigger than \fontsubfuzz. The \relax is necessary because this is a macro not a register.

```

55 \ifdim \font@submax >\fontsubfuzz\relax

```

In case you wonder about the \@gobbletwo inside the message below, this is a horrible hack to remove the tokens \on@line. that are added by \@font@warning at the end.

```

56 \font@warning{Size substitutions with differences\MessageBreak
57 up to \font@submax\space have occurred.\@gobbletwo}%
58 \fi

```

The macro \@defaultsubs is initially \relax but gets redefined to produce a warning if there have been some default font substitutions.

```

59 \@defaultsubs

```

The macro \@refundefined is initially \relax but gets redefined to produce a warning if there are undefined refs.

```

60 \@refundefined

```

If a label is defined more than once, \@tempswa will always be true and thus produce a “Label(s) may ...” warning. But since a rerun will not solve that problem (unless one uses a package like varioref that generates labels on the fly), we suppress this message.

```

61 \if@filesw
62 \ifx \@multiplelabels \relax
63 \if@tempswa
64 \@latex@warning@no@line{Label(s) may have changed.
65 Rerun to get cross-references right}%
66 \fi
67 \else
68 \@multiplelabels
69 \fi
70 \ifx \@extra@page@added \relax
71 \@latex@warning@no@line{Temporary extra page added at the end.
72 Rerun to get it removed}%
73 \fi

```

We could think of adding a warning that nothing can be corrected while \nofiles is in force. In the past the warnings related to the aux file are simply suppressed in this case.

```

74 \fi
75 }

```

(End of definition for \@enddocument@kernel@warnings.)

```

76 </2kernel | latexrelease>
77 <latexrelease>\EndIncludeInRelease
78 <latexrelease>\IncludeInRelease{2020/10/01}%
79 <latexrelease> \enddocument}{Use Hooks}%

```

```

80 <latexrelease>\def\enddocument{%
81 <latexrelease>    \kernel@before@enddocument
82 <latexrelease>    \UseOneTimeHook{enddocument}%
83 <latexrelease>    \kernel@after@enddocument
84 <latexrelease>    \checkend{document}%
85 <latexrelease>    \clearpage
86 <latexrelease>    \UseOneTimeHook{enddocument/afterlastpage}%
87 <latexrelease>    \kernel@after@enddocument@afterlastpage
88 <latexrelease>    \begingroup
89 <latexrelease>        \if@filesw
90 <latexrelease>            \immediate\closeout\@mainaux
91 <latexrelease>            \let\@setckpt\@gobbletwo
92 <latexrelease>            \let\@newl@bel\@testdef
93 <latexrelease>            \@tempswafalse
94 <latexrelease>            \makeatletter \@@input\jobname.aux
95 <latexrelease>        \fi
96 <latexrelease>        \UseOneTimeHook{enddocument/afteraux}%
97 <latexrelease>        \UseOneTimeHook{enddocument/info}%
98 <latexrelease>    \endgroup
99 <latexrelease>    \UseOneTimeHook{enddocument/end}%
100 <latexrelease>    \deadcycles\z@\@@end}
101 <latexrelease>\NewHook{enddocument}
102 <latexrelease>\NewHook{enddocument/afterlastpage}
103 <latexrelease>\NewHook{enddocument/afteraux}
104 <latexrelease>\NewHook{enddocument/info}
105 <latexrelease>\NewHook{enddocument/end}

```

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled.

```

106 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/filelist]
107 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/warnings]
108 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/release]

109 <latexrelease>\AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
110 <latexrelease>\AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
111 <latexrelease>\AddToHook{enddocument/info}[kernel/release]{%
112 <latexrelease>    \let\show@release@info\wlog
113 <latexrelease>    \show@release@info{ *****}%
114 <latexrelease>    \the\LaTeXReleaseInfo
115 <latexrelease>    \show@release@info{ *****}}
116 <latexrelease>
117 <latexrelease>\DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
118 <latexrelease>\DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}
119 <latexrelease>\def\@enddocument@kernel@warnings{%
120 <latexrelease>    \ifdim \font@submax >\fontsubfuzz\relax
121 <latexrelease>        \@font@warning{Size substitutions with differences\MessageBreak
122 <latexrelease>            up to \font@submax\space have occurred.\@gobbletwo}%
123 <latexrelease>    \fi
124 <latexrelease>    \@defaultsubs
125 <latexrelease>    \@refundefined
126 <latexrelease>    \if@filesw
127 <latexrelease>        \ifx \@multiplelabels \relax
128 <latexrelease>            \if@tempswa
129 <latexrelease>                \@latex@warning@no@line{Label(s) may have changed.
130 <latexrelease>                    Rerun to get cross-references right}%

```

```

131 <latexrelease> \fi
132 <latexrelease> \else
133 <latexrelease> \@multiplelabels
134 <latexrelease> \fi
135 <latexrelease> \ifx \@extra@page@added \relax
136 <latexrelease> \@latex@warning@no@line{Temporary extra page added at the end.
137 <latexrelease> Rerun to get it removed}%
138 <latexrelease> \fi
139 <latexrelease> \fi
140 <latexrelease>}
141 <latexrelease>\EndIncludeInRelease
142 <latexrelease>\IncludeInRelease{0000/00/00}%
143 <latexrelease> {\enddocument}{Use Hooks}%
144 <latexrelease>
145 <latexrelease>\def\enddocument{%
146 <latexrelease> \let\AtEndDocument\@firstofone
147 <latexrelease> \@enddocumenthook
148 <latexrelease> \@checkend{document}%
149 <latexrelease> \clearpage
150 <latexrelease> \begingroup
151 <latexrelease> \if@filesw
152 <latexrelease> \immediate\closeout\@mainaux
153 <latexrelease> \let\@setckpt\@gobbletwo
154 <latexrelease> \let\@newl@bel\@testdef
155 <latexrelease> \@tempswafalse
156 <latexrelease> \makeatletter \@@input\jobname.aux
157 <latexrelease> \fi
158 <latexrelease> \@dofilelist
159 <latexrelease> \ifdim \font@submax >\fontsubfuzz\relax
160 <latexrelease> \@font@warning{Size substitutions with differences\MessageBreak
161 <latexrelease> up to \font@submax\space have occurred.\@gobbletwo}%
162 <latexrelease> \fi
163 <latexrelease> \@defaultsubs
164 <latexrelease> \@refundefined
165 <latexrelease> \if@filesw
166 <latexrelease> \ifx \@multiplelabels \relax
167 <latexrelease> \if@tempswa
168 <latexrelease> \@latex@warning@no@line{Label(s) may have changed.
169 <latexrelease> Rerun to get cross-references right}%
170 <latexrelease> \fi
171 <latexrelease> \else
172 <latexrelease> \@multiplelabels
173 <latexrelease> \fi
174 <latexrelease> \fi
175 <latexrelease> \endgroup
176 <latexrelease> \deadcycles\z@\@@end}
177 <latexrelease>
178 <latexrelease>\let\@enddocument@kernel@warnings\@undefined
179 <latexrelease>
180 <latexrelease>\EndIncludeInRelease
181 <*2kernel>

```

`\@kernel@before@enddocument` The `\@kernel@before@enddocument` hook is slightly different because we initialize it with `\par` so that `\enddocument` always returns to vertical mode as its first action.


```

182 </2ekernel>
183 <*2ekernel | latexrelease>
184 <latexrelease>\IncludeInRelease{2021/06/01}%
185 <latexrelease>                {\@kernel@before@enddocument}{kernel before hook}%
186 \def\@kernel@before@enddocument{\par}
187 </2ekernel | latexrelease>
188 <latexrelease>\EndIncludeInRelease
    The rollback code renders it harmless.
189 <latexrelease>\IncludeInRelease{0000/00/00}%
190 <latexrelease>                {\@kernel@before@enddocument}{kernel before hook}%
191 <latexrelease>
192 <latexrelease>\let\@kernel@before@enddocument\@empty
193 <latexrelease>
194 <latexrelease>\EndIncludeInRelease
195 <*2ekernel>

```

(End of definition for \@kernel@before@enddocument.)

\@testdef

```

196 \def\@testdef #1#2#3{%
197   \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
198   \reserved@a \else \@tempswatrue \fi}

```

(End of definition for \@testdef.)

Reading data from auxiliary files (like `.toc` normally happens in vertical mode and it therefore doesn't matter if line endings are converted to spaces by $\text{\TeX}$ during that process.

However, especially the `.toc` file might be read in L-R mode (in cases the `\tableofcontents` attempts to put, say, a list of sub-sections as a paragraph). In that case the newlines after a line like

```
\contentsline {subsubsection}{\numberline {1.1.1}A C-head}{2}
```

might result in spurious spaces (e.g., when that level is not included).

That could be fixed by reading in the file using `\endlinechar=-1` but that has the danger that it drops some valid endlines that should be converted to spaces (for example, when the user edited the TOC and then used `\nofiles` to preserve it).

So the approach taken instead is this:

- `\addcontentsline` adds the command `\protected@file@percent` to the end of the second argument of `\@writefile` that is written to the `.aux`. As the name indicates this is a protected macro so it doesn't change if it is written out.
- When the `.aux` is read back in at the end of the run, `\@writefile` is executed and writes its second argument unmodified to the file with the extension given by its first argument. Or rather that was how it was in the past.
- Instead we change `\@writefile` slightly: basically it looks at the second argument and if the last token in there is `\protected@file@percent` then it is replaced by a percent character and that is then written out. If not (for example, if the data came from a user issued `\addtocontents`, or from some package that uses `\@writefile` for writing its own files) then the command behaves exactly as before.

`\protected@file@percent` Dummy cs to be replaced by a percent sign inside `\@writefile`. If it survives (when used incorrectly) it will expand to nothing in a typsetting context.

```

199 </2ekernel>
200 <*2ekernel|latexrelease>
201 <latexrelease>\IncludeInRelease{2018/12/01}%
202 <latexrelease>          {\protected@file@percent}{Mask line endings}%
203 \protected\def\protected@file@percent{}

```

(End of definition for `\protected@file@percent`.)

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and if the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```

204 \catcode'\^^A=9
205 \long\gdef\add@percent@to@temptokena
206   #1\protected@file@percent#2\add@percent@to@temptokena

```

When we call this macro in `\@writefile` we stick in `\@empty` at the beginning, so that in case the tokenlist consists of a single brace group the braces aren't stripped. The `\expandafter` then expands this extra token away again.

```

207   {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptokena
208     \expandafter\do@add@percent@to@temptokena\fi{#1}}
209 \long\def\dont@add@percent@to@temptokena#1{%
210   \@temptokena\expandafter{#1}}

```

`latexrelease` will read this code in high-speed mode in certain situations. During that it will only look for `\if` tests but not actually execute the `\catcode` change above. As a result it will drop anything after the `%` character in the definition. Therefore the `\fi` needs to be on the next line and we need locally another comment character to avoid getting spaces into the definition—a weird problem :-)

```

211 \begingroup
212 \catcode'\%=12
213 \catcode'\^^A=14
214 \long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{#1}\^^A

```

Can't be on the same line as the `%` — see above.

```

215   }}
216 \endgroup

```

(End of definition for `\add@percent@to@temptokena`.)

`\@writefile`

```

217 \long\def\@writefile#1#2{%
218   \@ifundefined{tf@#1}\relax
219   {%

```

If we write to the file we first prepare `#2` using `\add@percent@to@temptokena` and then write the token register out.

```

220     \add@percent@to@temptokena
221     \@empty#2\protected@file@percent
222     \add@percent@to@temptokena
223     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
224   }%
225 }

```

```

226 </2ekernel | latexrelease>
227 <latexrelease>\EndIncludeInRelease
228 <latexrelease>\IncludeInRelease{0000/00/00}%
229 <latexrelease>          {\protected@file@percent}{Mask line endings}%
230 <latexrelease>\let\protected@file@percent\@undefined
231 <latexrelease>\let\add@percent@to@temptokena\@undefined
232 <latexrelease>\let\do@add@percent@to@temptokena\@undefined
233 <latexrelease>\let\dont@add@percent@to@temptokena\@undefined
234 <latexrelease>\long\def\@writefile#1#2{%
235 <latexrelease>  \ifundefined{tf@#1}\relax
236 <latexrelease>    {\@temptokena{#2}%
237 <latexrelease>      \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
238 <latexrelease>    }%
239 <latexrelease>}
240 <latexrelease>\EndIncludeInRelease
241 <*2ekernel>

```

(End of definition for \@writefile.)

\stop

```

242 \def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}

```

(End of definition for \stop.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

243 \everypar{\@nodocument} %% To get an error if text appears before the
244 \nullfont                %% \begin{document}

```

\begin, \end, and \@checkend changed so \end{document} will catch an unmatched \begin. Changed 24 May 89 as suggested by Frank Mittelbach and Rainer Sch\"opf.

```

\begin{NAME} ==
BEGIN
  IF \NAME undefined THEN \reserved@a == BEGIN report error END
  ELSE \reserved@a ==
    (\@currenvir :=L NAME) \NAME
  FI
  @ignore :=G F      %% Added 30 Nov 88
  \begingroup
  \@endpe := F
  \@currenvir :=L NAME
  \NAME
END

```

```

\end{NAME} ==
BEGIN
  \endNAME
  \@checkend{NAME}
  \endgroup
  IF @endpe = T      %% @endpe set True by \@endparenv

```

```

        THEN \@doendpe          %% \@doendpe redefines \par and \everypar
                                %% to suppress paragraph indentation in
        FI                      %% immediately following text
        IF @ignore = T
            THEN @ignore :=G F
                \ignorespaces
        FI
    END

\@checkend{NAME} ==
BEGIN
    IF \@currenvir = NAME
        ELSE \@badend{NAME}
    FI
END

```

End of historical L^AT_EX 2.09 comments.

```

\begin
245 </2ekernel>
246 <*2ekernel | latexrelease>
247 <latexrelease>\IncludeInRelease{2020/10/01}%
248 <latexrelease>          {\begin}{Use hook system}%
249 \protected\def\begin#1{%
250     \UseHook{env/#1/before}%
251     \@ifundefined{#1}%
252         {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
253         {\def\reserved@a{\def\@currenvir{#1}%
254             \edef\@currenvline{\on@line}%
255             \@execute@begin@hook{#1}%
256             \csname #1\endcsname}}}%
257     \@ignorefalse
258     \begingroup\@endpfalse\reserved@a}

```

Before the `\document` code is executed we have to first undo the `\endgroup` as there should be none for this environment to avoid that changes on top-level unnecessarily go to T_EX's savestack, and we have to initialize all hooks in the hook system. So we need to test for this environment name. But once it has been found all this testing is no longer needed and so we redefine `\@execute@begin@hook` to simply use the hook.

```

259 \def\@execute@begin@hook #1{%
260     \expandafter\ifx\csname #1\endcsname\document
261         \endgroup
262         \gdef\@execute@begin@hook##1{\UseHook{env/##1/begin}}%
263         \expl@@@initialize@all@@
264     \fi

```

If this is an environment before `\begin{document}` we just run the hook so this can be outside the test.

```

265     \UseHook{env/#1/begin}%
266 }
267 </2ekernel | latexrelease>
268 <latexrelease>\EndIncludeInRelease

```

```

269 <latexrelease>\IncludeInRelease{2019/10/01}%
270 <latexrelease>          {\begin}{Making \begin/\end robust}%
271 <latexrelease>\DeclareRobustCommand\begin[1]{%
272 <latexrelease>  \ifundefined{#1}%
273 <latexrelease>    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
274 <latexrelease>    {\def\reserved@a{\def\@currentvir{#1}%
275 <latexrelease>      \edef\@currentvline{\on@line}%
276 <latexrelease>      \csname #1\endcsname}}}%
277 <latexrelease>  \ignorefalse
278 <latexrelease>  \begingroup\@endpfalse\reserved@a}
279 <latexrelease>\EndIncludeInRelease

```

A version that doesn't start out with `\relax` when in typesetting mode would be the following, but since `\begin` issues a `\begingroup` it wouldn't help much with respect to allowing things like `\noalign` or `\multicolumn` inside.

```

280 %\edef\begin
281 %  {\unexpanded{%
282 %    \ifx\protect\@typeset@protect
283 %      \expandafter\@gobble
284 %    \fi
285 %    \protect
286 %  }}%
287 %  \expandafter\noexpand\csname begin \endcsname
288 %  }
289 %\@namedef{begin }#1{%
290 %  \ifundefined{#1}%
291 %    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
292 %    {\def\reserved@a{\def\@currentvir{#1}%
293 %      \edef\@currentvline{\on@line}%
294 %      \csname #1\endcsname}}}%
295 %  \ignorefalse
296 %  \begingroup\@endpfalse\reserved@a}
297 <latexrelease>\IncludeInRelease{0000/00/00}%
298 <latexrelease>          {\begin}{Making \begin/\end robust}%
299 <latexrelease>\def\begin#1{%
300 <latexrelease>  \ifundefined{#1}%
301 <latexrelease>    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
302 <latexrelease>    {\def\reserved@a{\def\@currentvir{#1}%
303 <latexrelease>      \edef\@currentvline{\on@line}%
304 <latexrelease>      \csname #1\endcsname}}}%
305 <latexrelease>  \ignorefalse
306 <latexrelease>  \begingroup\@endpfalse\reserved@a}
307 <latexrelease>

```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

308 <latexrelease>\expandafter\let\csname begin \endcsname\@undefined
309 <latexrelease>
310 <latexrelease>\EndIncludeInRelease
311 <*2ekernel>

```

(End of definition for \begin.)

`\end` The top level definition for `\end`.

```

312 </2ekernel>
313 <*2ekernel | latexrelease>
314 <latexrelease>\IncludeInRelease{2019/10/01}%
315 <latexrelease>          {\end}{Making \begin/\end robust}%

```

While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafters`, the first to get

```
\csname endfoo\endcsname          \@checkend{foo}% etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo                          \@checkend{foo}% etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> \@checkend{foo}% etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end_` (when not doing typesetting) or it should produce `\end_` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end_` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo                          \@checkend{foo}% etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafters` inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```

316 \edef\end
317   {\unexpanded{%
318     \romannumeral
319     \ifx\protect\@typeset@protect
320       \expandafter      %1
321       \expandafter      %2
322       \expandafter      %1
323       \expandafter      %3 expands the \csname inside \end<space>
324       \expandafter      %1
325       \expandafter      %2 expands \end<space>
326       \expandafter      %1 expands the \else
327       \z@
328     \else
329       \expandafter\z@\expandafter\protect
330     \fi
331   }%
332   \expandafter\noexpand\csname end \endcsname
333 }
334 </2ekernel | latexrelease>
335 <latexrelease>\EndIncludeInRelease

```

And here is the original definition of `\end` the way it was in L^AT_EX for several decades now hidden in `\end_`.

```

336 <latexrelease>\IncludeInRelease{0000/00/00}%
337 <latexrelease>          {\end}{Making \begin/\end robust}%
338 <latexrelease>\def\end#1{%
339 <latexrelease>  \csname end#1\endcsname\@checkend{#1}%
340 <latexrelease>  \expandafter\endgroup\if@endpe\@doendpe\fi
341 <latexrelease>  \if@ignore\@ignorefalse\ignorespaces\fi}
342 <latexrelease>
343 <latexrelease>\EndIncludeInRelease
344 <*2ekernel>

```

(End of definition for `\end`.)

`\end\verbvisiblespace` The internal version with a space at the end.

```

345 </2ekernel>
346 <*2ekernel | latexrelease>
347 <latexrelease>\IncludeInRelease{2024/11/01}%
348 <latexrelease>          {\end!space}{New @endpe handling}%
349 \@namedef{end }#1{%
350   \romannumeral
351   \IfHookEmptyTF{env/#1/end}%
352     {\expandafter\z@}%
353     {\z@\UseHook{env/#1/end}}}%
354   \csname end#1\endcsname\@checkend{#1}%

```

We can now close the environment group and due to the new `\if@endpe` handling we no longer need to `\expandafter` out of the group.

```

355 %   \expandafter\endgroup\if@endpe\@doendpe\fi
356   \endgroup
357   \UseHook{env/#1/after}%
358   \if@ignore\@ignorefalse\ignorespaces\fi
359 }
360 </2ekernel | latexrelease>
361 <latexrelease>\EndIncludeInRelease

```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```

362 %   \begin{macrocode}
363 <latexrelease>\IncludeInRelease{2020/10/01}%
364 <latexrelease>          {\end!space}{Use hook system}%
365 <latexrelease>
366 <latexrelease>\@namedef{end }#1{%
367 <latexrelease>  \romannumeral
368 <latexrelease>    \IfHookEmptyTF{env/#1/end}%
369 <latexrelease>      {\expandafter\z@}%
370 <latexrelease>      {\z@\UseHook{env/#1/end}}}%
371 <latexrelease>    \csname end#1\endcsname\@checkend{#1}%
372 <latexrelease>    \expandafter\endgroup\if@endpe\@doendpe\fi
373 <latexrelease>    \UseHook{env/#1/after}%
374 <latexrelease>    \if@ignore\@ignorefalse\ignorespaces\fi
375 <latexrelease>}
376 <latexrelease>\EndIncludeInRelease

```

Version without the fix for tlb3722 for the record:

```

@namedef{end }#1{%
  \UseHook{env/#1/end}%
  \csname end#1\endcsname\@checkend{#1}%
  \expandafter\endgroup\if@endpe\@doendpe\fi
  \UseHook{env/#1/after}%
  \if@ignore\@ignorefalse\ignorespaces\fi}%

377 <latexrelease>\IncludeInRelease{2019/10/01}%
378 <latexrelease>          {\end!space}{Making \begin/\end robust}%
379 <latexrelease>
380 <latexrelease>\@namedef{end }#1{%
381 <latexrelease>  \csname end#1\endcsname\@checkend{#1}%
382 <latexrelease>  \expandafter\endgroup\if@endpe\@doendpe\fi
383 <latexrelease>  \if@ignore\@ignorefalse\ignorespaces\fi}
384 <latexrelease>\EndIncludeInRelease

385 <latexrelease>\IncludeInRelease{0000/00/00}%
386 <latexrelease>          {\end!space}{Making \begin/\end robust}%

```

Undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

387 <latexrelease>\expandafter\let\csname end \endcsname\@undefined
388 <latexrelease>
389 <latexrelease>\EndIncludeInRelease
390 <*2ekernel>

```

(End of definition for \end\verbvisiblespace.)

`\@checkend`

```

391 \def\@checkend#1{\def\reserved@a{#1}\ifx
392     \reserved@a\@currenvir \else\@badend{#1}\fi}

```

(End of definition for \@checkend.)

`\@currenvline` We do need a default value for `\@currenvline` on top-level since the document environment cancels the brace group. This means that a mismatch with `\begin{document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```

393 \let\@currenvline\@empty

```

(End of definition for \@currenvline.)

`\AtBeginEnvironment` We provide 4 high-level hook interfaces directly, the others only when etoolbox is loaded

```

\AtEndEnvironment 394 </2ekernel>
\BeforeBeginEnvironment 395 <*2ekernel | latexrelease>
\AfterEndEnvironment 396 <latexrelease>\IncludeInRelease{2020/10/01}%
397 <latexrelease>          {\AtBeginEnvironment}{Hooks for environments}%

398 \newcommand\AtBeginEnvironment[2][.] {\AddToHook{env/#2/begin}[#1]}
399 \newcommand\AtEndEnvironment[2][.]   {\AddToHook{env/#2/end}[#1]}
400 \newcommand\BeforeBeginEnvironment[2][.]{\AddToHook{env/#2/before}[#1]}
401 \newcommand\AfterEndEnvironment[2][.]{\AddToHook{env/#2/after}[#1]}

402 </2ekernel | latexrelease>
403 <latexrelease>\EndIncludeInRelease

```



```

404 <latexrelease>\IncludeInRelease{0000/00/00}%
405 <latexrelease>                {\AtBeginEnvironment}{Hooks for environments}%
406 <latexrelease>
407 <latexrelease>\let\AtBeginEnvironment\@undefined
408 <latexrelease>\let\AtEndEnvironment\@undefined
409 <latexrelease>\let\BeforeBeginEnvironment\@undefined
410 <latexrelease>\let\AfterEndEnvironment\@undefined
411 <latexrelease>
412 <latexrelease>\EndIncludeInRelease
413 <*2ekernel>

```

(End of definition for `\AtBeginEnvironment` and others. These functions are documented on page 241.)

1.2 Center, Flushright, Flushleft

```

414 \message{center,}

```

Historical *LaTeX* 2.09 comments (not necessarily accurate any more):

```

\center, \flushright and \flushleft set
  \rightskip = 0pt or \@flushglue (as appropriate)
  \leftskip  = 0pt or \@flushglue (as appropriate)
  \parindent = 0pt
  \parfillskip = 0pt. (except \flushleft)
  \\\          == \par \vskip -\parskip
  \|[LENGTH] == \\\ \vskip LENGTH
  \\\*         == \par \penalty 10000 \vskip -\parskip
  \\\*[LEN]    == \\\* \vskip LENGTH

```

They invoke the `trivlist` environment to handle vertical spacing before and after them.

`\centering`, `\raggedright` and `\raggedleft` are the declaration analogs of the above.

`\raggedright` has a more universal effect, however. It sets `\@rightskip := flushglue`. Every environment, like the list environments, that set `\rightskip` to its 'normal' value set it to `\@rightskip`

End of historical *LaTeX* 2.09 comments.

`\@centercr`

```

415 </2ekernel>
416 <*2ekernel | latexrelease>
417 <latexrelease>\IncludeInRelease{2020/02/02}%
418 <latexrelease>                {\@centercr}{Make robust}%
419 \protected\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
420   \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
421 </2ekernel | latexrelease>

```

```

422 <latexrelease>\EndIncludeInRelease
423 <latexrelease>\IncludeInRelease{0000/00/00}%
424 <latexrelease>          {\@centercr}{Make robust}%
425 <latexrelease>
426 <latexrelease>\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
427 <latexrelease>          \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
428 <latexrelease>
429 <latexrelease>\EndIncludeInRelease
430 <*2ekernel>

```

(End of definition for \@centercr.)

\@xcentercr

```

431 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
432   [\@icentercr\ignorespaces}

```

(End of definition for \@xcentercr.)

\@icentercr

```

433 </2ekernel>
434 <*2ekernel | latexrelease>
435 <latexrelease>\IncludeInRelease{2020/10/01}%
436 <latexrelease>          {\@icentercr}{centering, etc support calc}%
437 \def\@icentercr[#1]{\@vspace@calcify{#1}\ignorespaces}
438 </2ekernel | latexrelease>
439 <latexrelease>\EndIncludeInRelease
440 <latexrelease>\IncludeInRelease{0000/00/00}%
441 <latexrelease>          {\@icentercr}{centering, etc support calc}%
442 <latexrelease>
443 <latexrelease>\def\@icentercr[#1]{\vskip #1\ignorespaces}
444 <latexrelease>\EndIncludeInRelease
445 <*2ekernel>

```

(End of definition for \@icentercr.)

center (*env.*) We use \relax to prevent \item scanning too far.

```

446 \def\center{\trivlist \centering\item\relax}
447 \def\endcenter{\endtrivlist}
448 </2ekernel>
449 <*2ekernel | latexrelease>
450 <latexrelease>\IncludeInRelease{2020/10/01}%
451 <latexrelease>          {\centering}{Set finaldhyphenemerits}%

```

\centering

```

452 \DeclareRobustCommand\centering{%
453   \let\\ \@centercr
454   \rightskip\@flushglue\leftskip\@flushglue
455   \finalhyphenemerits=\z@
456   \parindent\z@\parfillskip\z@skip}

```

(End of definition for \centering.)

`\raggedright`

```
457 \DeclareRobustCommand\raggedright{%  
458   \let\\@centercr\@rightskip\@flushglue \rightskip\@rightskip  
459   \finalhyphendemerits=\z@  
460   \leftskip\z@skip  
461   \parindent\z@}
```

(End of definition for \raggedright.)

`\raggedleft`

```
462 \DeclareRobustCommand\raggedleft{%  
463   \let\\@centercr  
464   \rightskip\z@skip\leftskip\@flushglue  
465   \finalhyphendemerits=\z@  
466   \parindent\z@\parfillskip\z@skip}
```

(End of definition for \raggedleft.)

```
467 </2ekernel | latexrelease>  
468 <latexrelease>\EndIncludeInRelease  
469 <latexrelease>\IncludeInRelease{2019/10/01}%  
470 <latexrelease>           {\centering}{Make commands robust}%  
471 <latexrelease>  
472 <latexrelease>\DeclareRobustCommand\centering{%  
473 <latexrelease>   \let\\@centercr  
474 <latexrelease>   \rightskip\@flushglue\leftskip\@flushglue  
475 <latexrelease>   \parindent\z@\parfillskip\z@skip}  
476 <latexrelease>\DeclareRobustCommand\raggedright{%  
477 <latexrelease>   \let\\@centercr\@rightskip\@flushglue \rightskip\@rightskip  
478 <latexrelease>   \leftskip\z@skip  
479 <latexrelease>   \parindent\z@}  
480 <latexrelease>\DeclareRobustCommand\raggedleft{%  
481 <latexrelease>   \let\\@centercr  
482 <latexrelease>   \rightskip\z@skip\leftskip\@flushglue  
483 <latexrelease>   \parindent\z@\parfillskip\z@skip}  
484 <latexrelease>\EndIncludeInRelease  
485 <latexrelease>  
486 <latexrelease>\IncludeInRelease{0000/00/00}%  
487 <latexrelease>           {\centering}{Make commands robust}%  
488 <latexrelease>  
489 <latexrelease>\kernel@make@fragile\centering  
490 <latexrelease>\kernel@make@fragile\raggedright  
491 <latexrelease>\kernel@make@fragile\raggedleft  
492 <latexrelease>  
493 <latexrelease>\EndIncludeInRelease  
494 <*2ekernel>
```

`\@rightskip`

```
495 \newskip\@rightskip \@rightskip \z@skip
```

(End of definition for \@rightskip.)

`flushleft (env.)` We use `\relax` to prevent `\item` scanning too far.

```
496 \def\flushleft{\trivlist \raggedright\item\relax}  
497 \def\endflushleft{\endtrivlist}
```

`flushright` (*env.*) We use `\relax` to prevent `\item` scanning too far.

```
498 \def\flushright{\trivlist \raggedleft\item\relax}
499 \def\endflushright{\endtrivlist}
```

1.3 Verbatim

```
500 \message{verbatim,}
```

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode`'d to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The *-variants of these commands are the same, except that spaces print as the T_EXbook's space character instead of as blank spaces.

`\@vobeyspaces`

`\@vobeytabs`

```
501 </2ekernel>
502 <latexrelease>\IncludeInRelease{2023/11/01}%
503 <latexrelease>          {\@vobeytabs}{Obeyed tabs}%
504 <*2ekernel | latexrelease>
505 {\catcode'\ =\active%
506 \gdef\@vobeyspaces{\catcode'\ \active\let \@xobeysp\@vobeytabs}}
507 {\catcode'\^~I=\active
508 \gdef\@vobeytabs{\catcode'\^~I\active\let^~I\@xobeytab}%
509 \global\let^~I=\space
510 }
511 </2ekernel | latexrelease>
512 <latexrelease>\EndIncludeInRelease
513 <latexrelease>\IncludeInRelease{0000/00/00}%
514 <latexrelease>          {\@vobeytabs}{Obeyed tabs}%
515 <latexrelease>{\catcode'\ =\active%
516 <latexrelease>\gdef\@vobeyspaces{\catcode'\ \active\let \@xobeysp}}
517 <latexrelease>\let\@vobeytabs\@undefined
518 <latexrelease>\EndIncludeInRelease
519 <*2ekernel>
```

(End of definition for \@vobeyspaces and \@vobeytabs.)

`\@xobeysp`

(End of definition for \@xobeysp.)

`\@xverbatim`

`\@sxverbatim`

```
520 \begingroup \catcode '| =0 \catcode '[ = 1
521 \catcode'] =2 \catcode '\{ =12 \catcode '\} =12
522 \catcode'\ =12 |gdef|@xverbatim#1\end{verbatim} [#1|end[verbatim]]
523 |gdef|@sxverbatim#1\end{verbatim*} [#1|end[verbatim*]]
524 \endgroup
```

(End of definition for \@xverbatim and \@sxverbatim.)

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```

525 </2kernel>
526 <*2kernel | latexrelease>
527 <latexrelease>\IncludeInRelease{2017-04-15}{\@verbatim}%
528 <latexrelease>          {Disable hyphenation in verbatim}%
529 \def\@verbatim{\trivlist \item\relax
530   \if@minipage\else\vskip\parskip\fi
531   \leftskip\@totalleftmargin\rightskip\z@skip
532   \parindent\z@\parfillskip\@flushglue\parskip\z@skip

```

Added `\@@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says). Switch language when in vertical mode.

```

533   \@@par

```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```

534   \language\l@nohyphenation
535   \@tempswafalse
536   \def\par{%
537     \if@tempswa

```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```

538       \leavevmode \null \@@par\penalty\interlinepenalty
539     \else
540       \@tempswatrue
541     \ifhmode\@@par\penalty\interlinepenalty\fi
542   \fi}%

```

To allow customization we hide the font used in a separate macro.

```

543   \let\do\@makeother \dospecials
544   \obeylines \verbatim@font \@noligs

```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```

545   \everypar \expandafter{\the\everypar \unpenalty}%
546 }
547 </2kernel | latexrelease>
548 <latexrelease>\EndIncludeInRelease
549 <latexrelease>\IncludeInRelease{0000-00-00}{\@verbatim}%
550 <latexrelease>          {Disable hyphenation in verbatim}%
551 <latexrelease>\def\@verbatim{\trivlist \item\relax
552 <latexrelease>   \if@minipage\else\vskip\parskip\fi
553 <latexrelease>   \leftskip\@totalleftmargin\rightskip\z@skip
554 <latexrelease>   \parindent\z@\parfillskip\@flushglue\parskip\z@skip
555 <latexrelease>   \@@par
556 <latexrelease>   \@tempswafalse
557 <latexrelease>   \def\par{%
558 <latexrelease>     \if@tempswa
559 <latexrelease>       \leavevmode \null \@@par\penalty\interlinepenalty
560 <latexrelease>     \else
561 <latexrelease>       \@tempswatrue
562 <latexrelease>       \ifhmode\@@par\penalty\interlinepenalty\fi
563 <latexrelease>     \fi}%
564 <latexrelease>   \let\do\@makeother \dospecials

```

```

565 <latexrelease> \obeylines \verbatim@font \@noligs
566 <latexrelease> \hyphenchar\font\m@ne
567 <latexrelease> \everypar \expandafter{\the\everypar \unpenalty}%
568 <latexrelease>}
569 <latexrelease>\EndIncludeInRelease
570 <*2ekernel>

```

(End of definition for \@verbatim.)

verbatim (RmS 93/09/19) Protected against ‘missing item’ error message triggered by empty verbatim environment.

```

571 \def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
572 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

```

(End of definition for \verbatim and \endverbatim.)

\verbatim@font Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```

573 \def\verbatim@font{\normalfont\ttfamily}

```

(End of definition for \verbatim@font.)

```

574 </2ekernel>
575 <*2ekernel | latexrelease>
576 <latexrelease>\IncludeInRelease{2018/12/01}%
577 <latexrelease> \verbvisiblespace\Setup visible space for \verb}%

```

\asciispace The character in slot 32, in typewriter fonts (historically) a visible space but in other fonts a real space or something else

```

578 \DeclareRobustCommand\asciispace{\char 32 }

```

(End of definition for \asciispace.)

\verbvisiblespace This defines how to get a visible space in **\verb*** and friends. In classic T_EX this is just the slot 32, but in T_U encoded fonts we switch fonts and take the character from cmtt.

```

579 \ifx\Umathcode\@undefined
580   \let\verbvisiblespace\asciispace % Pdftex version
581 \else
582   \DeclareRobustCommand\verbvisiblespace
583     {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}} % xetex/luatex version
584 \fi

```

(End of definition for \verbvisiblespace.)

\@verbvisiblespacebox The box to hold the visible space character if it isn’t in slot 32 in the current typewriter font.

```

585 \newbox\@verbvisiblespacebox

```

(End of definition for \@verbvisiblespacebox.)

verbatim* (*env.*) For **verbatim*** we also set up the correct visible space character definition and then run **\@vobeyspaces**. As this code is not called as part of the normal verbatim environment (the method is done the other way around this time) we don’t have to check if space is already active—it shouldn’t be.

```

586 \@namedef{verbatim*}{\@verbatim
587   \@setupverbvisiblespace
588   \frenchspacing\@vobeyspaces\@sxverbatim}
589 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

```

```

590 </2ekernel | latexrelease>
591 <latexrelease>\EndIncludeInRelease
592 <latexrelease>\IncludeInRelease{0000/00/00}%
593 <latexrelease>          {\verbvisiblespace}{Setup visible space for \verb}%
594 <latexrelease>
595 <latexrelease>\@namedef{verbatim*}{\@verbatim\@sxverbatim}
596 <latexrelease>
597 <latexrelease>\let\asciispace          \@undefined
598 <latexrelease>\let\verbvisiblespace    \@undefined
599 <latexrelease>\let\@setupverbvisiblespace\@undefined
600 <latexrelease>\let\@verbvisiblespacebox \@undefined
601 <latexrelease>\EndIncludeInRelease
602 <*2ekernel>

```

`\@setupverbvisiblespace` In pdfTeX a catcode 12 space will produce the character in slot 32 which is assumed to be a visible space character (in a typewriter font in OT1 or T1 encoding). In XeTeX or LuaTeX a font in TU encoding is normally used and that has a real space in this slot. So what we do in this case is this: we check the definition of `\verbvisiblespace` and if it is `\asciispace` we assume that the char32 can be used (e.g., in pdfTeX). We then redefine `\@xobeysp` so that after running `\@xobeyspaces` we get characters from slot 32 for each active space.

```

603 </2ekernel>
604 <*2ekernel | latexrelease>
605 <latexrelease>\IncludeInRelease{2023/11/01}%
606 <latexrelease>          {\@setupverbvisiblespace}{Setup visible tab for \verb}%
607 \def\@setupverbvisiblespace{%
608   \ifx\verbvisiblespace\asciispace
609     \let\@xobeysp\asciispace
610   \else

```

Otherwise we measure the width of a character in the mono-spaced current font and place a `\verbvisiblespace` into a box of the right width which we are then using as the character for a space. By default this will be the space character from OT1 cmtt but by changing `\verbvisiblespace` one could use, for example, the `\textvisiblespace` of the current typewriter font.

```

611   \setbox\z@\hbox{x}%
612   \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
613   \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
614 \fi
615 \@setupverbvisibletab
616 }
617 </2ekernel | latexrelease>
618 <latexrelease>\EndIncludeInRelease
619 <latexrelease>\IncludeInRelease{2018/12/01}%
620 <latexrelease>          {\@setupverbvisiblespace}{Setup visible space for \verb}%
621 <latexrelease>\def\@setupverbvisiblespace{%
622 <latexrelease>   \ifx\verbvisiblespace\asciispace
623 <latexrelease>     \let\@xobeysp\asciispace
624 <latexrelease>   \else
625 <latexrelease>     \setbox\z@\hbox{x}%
626 <latexrelease>     \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
627 <latexrelease>     \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
628 <latexrelease>   \fi

```

```

629 <latexrelease>}
630 <latexrelease>\EndIncludeInRelease
631 <latexrelease>\IncludeInRelease{0000/00/00}%
632 <latexrelease>{\@setupverbvisiblespace}{Setup visible space for \verb}%
633 <latexrelease>\let\@setupverbvisiblespace\@undefined
634 <latexrelease>\EndIncludeInRelease
635 <*2ekernel>

```

(End of definition for \@setupverbvisiblespace.)

\@setupverbvisibletab A redirection: just a simple wrapper.

```

636 </2ekernel>
637 <latexrelease>\IncludeInRelease{2023/11/01}%
638 <latexrelease>{\@setupverbvisibletab}{Setup visible tab for \verb}%
639 <*2ekernel | latexrelease>
640 \def\@setupverbvisibletab{\let\@xobeytab\@xobeysp}
641 </2ekernel | latexrelease>
642 <latexrelease>\EndIncludeInRelease
643 <latexrelease>\IncludeInRelease{0000/00/00}%
644 <latexrelease>{\@setupverbvisibletab}{Setup visible tab for \verb}%
645 <latexrelease>\let\@setupverbvisibletab\@undefined
646 <latexrelease>\EndIncludeInRelease
647 <*2ekernel>

```

(End of definition for \@setupverbvisibletab.)

\@sverb Definitions of \@sverb and \@verb changed so \verb+ foo+ does not lose leading blanks
\@@sverb when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```

648 </2ekernel>
649 <*2ekernel | latexrelease>
650 <latexrelease>\IncludeInRelease{2023/11/01}%
651 <latexrelease>{\@sverb}{Support visible tabs}%

```

If the users types \verb !~! foo then surprisingly we would get the space as the delimiter and thus “!~!foo” in the output. To avoid this scenario we check if #1 has the character code of a space, if so we recurse otherwise we call \@@sverb (which is the original definition of \@sverb).

```

652 \def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}

653 \def\@@sverb#1{%
654   \catcode'#1\active
655   \lccode'\~'#1%
656   \gdef\verb@balance@group{\verb@egroup
657     \@latexerror{\noexpand\verb illegal in argument}\@ehc}%
658   \aftergroup\verb@balance@group
659   \lowercase{\let~\verb@egroup}%

```

If \@sverb is called from \@verb then space is already active and supposed to produce a real space. In this case we do nothing. Otherwise we run \@setupverbvisiblespace to setup the right visible space char and afterwards \@vobeyspaces to make it the definition for the active space character.

```

660   \ifnum0%
661     \ifnum\catcode'\ =\active\else 1\fi
662     \ifnum\catcode'\~I=\active\else 1\fi

```



```

663     =0 %
664     \else \@setupverbvisiblespace \@vobeyspaces \fi
665 }

666 </2ekernel | latexrelease>
667 <latexrelease>\EndIncludeInRelease
668 <latexrelease>\IncludeInRelease{2020/10/01}%
669 <latexrelease>          {\@sverb}{Drop spaces before \verb delimiter}%
670 <latexrelease>\def\@sverb#1{%
671 <latexrelease>  \catcode'#1\active
672 <latexrelease>  \lccode'\~'#1%
673 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
674 <latexrelease>    \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
675 <latexrelease>  \aftergroup\verb@balance@group
676 <latexrelease>  \lowercase{\let~\verb@egroup}%
677 <latexrelease>  \ifnum\catcode'\ =\active
678 <latexrelease>  \else \@setupverbvisiblespace \@vobeyspaces \fi
679 <latexrelease>}
680 <latexrelease>\EndIncludeInRelease
681 <latexrelease>\IncludeInRelease{2018/12/01}%
682 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
683 <latexrelease>
684 <latexrelease>\def\@sverb#1{%
685 <latexrelease>  \catcode'#1\active
686 <latexrelease>  \lccode'\~'#1%
687 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
688 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
689 <latexrelease>  \aftergroup\verb@balance@group
690 <latexrelease>  \lowercase{\let~\verb@egroup}%
691 <latexrelease>  \ifnum\catcode'\ =\active
692 <latexrelease>  \else \@setupverbvisiblespace \@vobeyspaces \fi
693 <latexrelease>}
694 <latexrelease>\let\@sverb\@undefined
695 <latexrelease>\EndIncludeInRelease
696 <latexrelease>
697 <latexrelease>\IncludeInRelease{0000/00/00}%
698 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
699 <latexrelease>\def\@sverb#1{%
700 <latexrelease>  \catcode'#1\active
701 <latexrelease>  \lccode'\~'#1%
702 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
703 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
704 <latexrelease>  \aftergroup\verb@balance@group
705 <latexrelease>  \lowercase{\let~\verb@egroup}}%
706 <latexrelease>
707 <latexrelease>\EndIncludeInRelease
708 <*2ekernel>

```

(End of definition for \@sverb and \@sverb.)

\@makeother

```

709 \def\@makeother#1{\catcode'#112\relax}

```

(End of definition for \@makeother.)

`\verb@balance@group`

```
710 \let\verb@balance@group\@empty
(End of definition for \verb@balance@group.)
```

`\verb@egroup`

```
711 \def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
(End of definition for \verb@egroup.)
```

`\verb@eol@error`

```
712 \begingroup
713   \obeylines%
714   \gdef\verb@eol@error{\obeylines%
715     \def~M{\verb@egroup\@latex@error{%
716       \noexpand\verb ended by end of line}\@ehc}}%
717 \endgroup
(End of definition for \verb@eol@error.)
```

`\verb` Typesetting a small piece verbatim.

```
718 </2ekernel>
719 <*2ekernel | latexrelease>
720 <latexrelease>\IncludeInRelease{2017-04-15}{\verb}%
721 <latexrelease>           {Disable hyphenation in verb}%
722 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
723   \bgroup
724     \verb@eol@error \let\do\@makeoother \dospecials
725     \verbatim@font\@noligs
Set \language here to suppress hyphenation. Done this way rather than setting
\hyphenchar as that is a global setting.
726     \language\l@nohyphenation
727     \ifstar\@sverb\@verb}
728 </2ekernel | latexrelease>
729 <latexrelease>\EndIncludeInRelease
730 <latexrelease>\IncludeInRelease{0000-00-00}{\verb}%
731 <latexrelease>           {Disable hyphenation in verb}%
732 <latexrelease>\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
733 <latexrelease>   \bgroup
734 <latexrelease>     \verb@eol@error \let\do\@makeoother \dospecials
735 <latexrelease>     \verbatim@font\@noligs
736 <latexrelease>     \ifstar\@sverb\@verb}
737 <latexrelease>\EndIncludeInRelease
738 <*2ekernel>
(End of definition for \verb.)
```

`\@verb`

```
739 \def\@verb{\@vobeyspaces \frenchspacing \@sverb}
(End of definition for \@verb.)
```

`\verbatim@nolig@list`

```
740 \def\verbatim@nolig@list{\do\‘\do\<\do\>\do\,\do\’\do\~}
```

(End of definition for \verbatim@nolig@list.)

\do@noligs

```
741 \def\do@noligs#1{%  
742   \catcode'#1\active  
743   \begingroup  
744     \lccode'\~'#1\relax  
745     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char'#1}}}
```

(End of definition for \do@noligs.)

\@noligs To stay compatible with packages that use \@noligs we keep it.

```
746 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

(End of definition for \@noligs.)

```
747 \</2ekernel>
```

File 38

ltmath.dtx

1 Math setup

This file contains a lot of the original plain T_EX code, as well as the L^AT_EX environments for math. It still needs sorting out.

```
1 <*2kernel>
2 \message{math definitions,}
```

1.1 Math commands based on plain T_EX

1.1.1 The log-like functions

\log The standard operators:

```
3 \DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
4 \DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
5 \DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
6 \DeclareRobustCommand\lim{\mathop{\operator@font lim}}
7 \DeclareRobustCommand\limsup{\mathop{\operator@font lim}\,sup}}
8 \DeclareRobustCommand\liminf{\mathop{\operator@font lim}\,inf}}
9 \DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
10 \DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
11 \DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
12 \DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
13 \DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
14 \DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
15 \DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
16 \DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
17 \DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
18 \DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
19 \DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
20 \DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
21 \DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
22 \DeclareRobustCommand\max{\mathop{\operator@font max}}
23 \DeclareRobustCommand\min{\mathop{\operator@font min}}
24 \DeclareRobustCommand\sup{\mathop{\operator@font sup}}
25 \DeclareRobustCommand\inf{\mathop{\operator@font inf}}
26 \DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
27 \DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
28 \DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
29 \DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
30 \DeclareRobustCommand\det{\mathop{\operator@font det}}
31 \DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
32 \DeclareRobustCommand\Pr{\mathop{\operator@font Pr}}
33 \DeclareRobustCommand\gcd{\mathop{\operator@font gcd}}
34 \DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

(End of definition for \log.)

\bmod And some operators have to be done by hand:

```

35 \DeclareRobustCommand\bmod{%
36   \nonscript\mskip-\medmuskip\mkern5mu%
37   \mathbin{\operator@font mod}\penalty900\mkern5mu%
38   \nonscript\mskip-\medmuskip}

```

(End of definition for \bmod.)

\pmod

```

39 \DeclareRobustCommand\pmod[1]{%
40   \allowbreak\mkern18mu({\operator@font mod}\,\,\,#1)}

```

(End of definition for \pmod.)

1.1.2 Biggggg

\big Variants on **\big** and friends for use with delimiters:

```

41 \DeclareRobustCommand\bigl{\mathopen\big}
42 \DeclareRobustCommand\bigm{\mathrel\big}
43 \DeclareRobustCommand\bigg{\mathclose\big}
44 \DeclareRobustCommand\Bigl{\mathopen\Big}
45 \DeclareRobustCommand\Bigm{\mathrel\Big}
46 \DeclareRobustCommand\Bigg{\mathclose\Big}
47 \DeclareRobustCommand\biggl{\mathopen\bigg}
48 \DeclareRobustCommand\biggm{\mathrel\bigg}
49 \DeclareRobustCommand\biggr{\mathclose\bigg}
50 \DeclareRobustCommand\Biggl{\mathopen\Bigg}
51 \DeclareRobustCommand\Biggm{\mathrel\Bigg}
52 \DeclareRobustCommand\Biggr{\mathclose\Bigg}

```

(End of definition for \big.)

1.1.3 The UNSORTED Rest

The other math commands are lifted from plain T_EX.

\jot

```

53 \newdimen\jot
54 \jot=3pt

```

(End of definition for \jot.)

\interdisplaylinepenalty

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

(End of definition for \interdisplaylinepenalty.)

\choose

```

57 \def\choose{\atopwithdelims()}

```

(End of definition for \choose.)

\brack

```

58 \def\brack{\atopwithdelims[]}

```

(End of definition for \brack.)

```

\brace
59 \def\brace{\atopwithdelims\{\}}
(End of definition for \brace.)

\mathpalette
60 \def\mathpalette#1#2{%
61   \mathchoice
62     {#1\displaystyle{#2}}%
63     {#1\textstyle{#2}}%
64     {#1\scriptstyle{#2}}%
65     {#1\scriptscriptstyle{#2}}
(End of definition for \mathpalette.)

\root
\rootbox
\root
66 \newbox\rootbox
67 \def\root#1\of{%
68   \setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}%
69   \mathpalette\root
70 \def\root#1#2{%
71   \setbox\rootbox\hbox{$\m@th#1\sqrt{#2}$}%
72   \dimen@=\ht\rootbox \advance\dimen@-\dp\rootbox
73   \mkern5mu\raise.6\dimen@\copy\rootbox
74   \mkern-10mu\box\rootbox
(End of definition for \root, \rootbox, and \root.)

\phantom
\hphantom
\mathstrut
75 \newif\ifv@
76 \newif\ifh@
77 </2ekernel>
78 <*2ekernel | latexrelease>
79 <latexrelease>\IncludeInRelease{2019/10/01}%
80 <latexrelease>{\vphantom}{Make commands robust}%
81 \DeclareRobustCommand\vphantom{\v@true\h@false\ph@nt}
82 \DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
83 \DeclareRobustCommand\phantom{\v@true\h@true\ph@nt}
84 \DeclareRobustCommand\mathstrut{\vphantom{}}
85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease>{\vphantom}{Make commands robust}%
89 <latexrelease>
90 <latexrelease>\kernel@make@fragile\vphantom
91 <latexrelease>\kernel@make@fragile\hphantom
92 <latexrelease>\kernel@make@fragile\phantom
93 <latexrelease>\kernel@make@fragile\mathstrut
94 <latexrelease>
95 <latexrelease>\EndIncludeInRelease
96 <*2ekernel>

```

```

97 \def\ph@nt{%
98   \ifmode
99     \expandafter\mathpalette\expandafter\mathph@nt
100   \else
101     \expandafter\makeph@nt
102   \fi}

103 \def\makeph@nt#1{%
104   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}

105 \def\mathph@nt#1#2{%
106   \setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}

107 </2ekernel>
108 <*2ekernel | latexrelease>
109 <latexrelease>\IncludeInRelease{2018/12/01}%
110 <latexrelease>          {\finph@nt}{Start LR-mode}%
111 \def\finph@nt{%
112   \setbox\tw@\null
113   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
114   \ifh@ \wd\tw@\wd\z@\fi

115   \leavevmode@ifvmode\box\tw@}
116 </2ekernel | latexrelease>
117 <latexrelease>\EndIncludeInRelease
118 <latexrelease>\IncludeInRelease{0000/00/00}%
119 <latexrelease>          {\finph@nt}{Start LR-mode}%
120 <latexrelease>\def\finph@nt{%
121 <latexrelease>   \setbox\tw@\null
122 <latexrelease>   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
123 <latexrelease>   \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
124 <latexrelease>\EndIncludeInRelease
125 <*2ekernel>

```

(End of definition for \phantom and others.)

\smash

```

126 \DeclareRobustCommand\smash{%
127   \relax % \relax, in case this comes first in \halign
128   \ifmode
129     \expandafter\mathpalette\expandafter\mathsm@sh
130   \else
131     \expandafter\makesm@sh
132   \fi}

133 \def\makesm@sh#1{%
134   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}

135 </2ekernel>
136 <*2ekernel | latexrelease>
137 <latexrelease>\IncludeInRelease{2022/11/01}%
138 <latexrelease>          {\mathsm@sh}{Guard against reboxing}%
139 \def\mathsm@sh#1#2{%
140   \setbox\z@\hbox{$\m@th#1{#2}$}%

```

The empty brace groups in front of the smashed box (which is placed by \finsm@sh) ensures that a \smash in math is not just producing a single box with its dimensions altered, but a box plus this second ord atom. The reason is that T_EX sometimes reboxes

a box if its the only thing in a place like the denominator of a fraction. This would then undo the smashing and the additional ord atom prevents that. Two ord atoms in a row do not alter the horizontal spacing in a formula so this is otherwise transparent.

```

141   {} \finsm@sh}

142 </2ekernel | latexrelease>
143 <latexrelease> \EndIncludeInRelease
144 <latexrelease> \IncludeInRelease{0000/00/00}%
145 <latexrelease>           {\mathsm@sh}{Guard against reboxing}%
146 <latexrelease> \def\mathsm@sh#1#2{%
147 <latexrelease>   \setbox\z@\hbox{$\m@th#1{#2}$} \finsm@sh}
148 <latexrelease> \EndIncludeInRelease
149 <*2ekernel>

```

```

150 </2ekernel>
151 <*2ekernel | latexrelease>
152 <latexrelease> \IncludeInRelease{2018/12/01}%
153 <latexrelease>           {\finsm@sh}{Start LR-mode}%
154 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevmode@ifvmode\box\z@}
155 </2ekernel | latexrelease>
156 <latexrelease> \EndIncludeInRelease
157 <latexrelease> \IncludeInRelease{0000/00/00}%
158 <latexrelease>           {\finsm@sh}{Start LR-mode}%
159 <latexrelease> \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
160 <latexrelease> \EndIncludeInRelease
161 <*2ekernel>

```

(End of definition for \smash.)

\buildrel

```

162 \def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}

```

(End of definition for \buildrel.)

```

163 </2ekernel>
164 <*2ekernel | latexrelease>
165 <latexrelease> \IncludeInRelease{2019/10/01}%
166 <latexrelease>           {\cases}{Make commands robust}%

```

\cases

```

167 \DeclareRobustCommand*\cases[1]{\left\{\!,\vcenter{\normalbaselines\m@th
168   \ialign{##\hfil$&\quad{##}\hfil\crr#1\crr}\right.}

```

(End of definition for \cases.)

\matrix

```

169 \DeclareRobustCommand*\matrix[1]{\null\!,\vcenter{\normalbaselines\m@th
170   \ialign{\hfil$##$\hfil&\quad\hfil$##$\hfil\crr
171     \mathstrut\crr\noalign{\kern-\baselineskip}
172     #1\crr\mathstrut\crr\noalign{\kern-\baselineskip}}}\!,}

```

(End of definition for \matrix.)

\pmatrix

```

173 \DeclareRobustCommand*\pmatrix[1]{\left(\matrix{#1}\right)}

```


(End of definition for `\pmatrix`.)

```

174 \end{kernel} \latexrelease
175 \latexrelease\EndIncludeInRelease
176 \latexrelease\IncludeInRelease{0000/00/00}%
177 \latexrelease\cases{\Make commands robust}%
178 \latexrelease
179 \latexrelease\kernel@make@fragile\cases
180 \latexrelease\kernel@make@fragile\matrix
181 \latexrelease\kernel@make@fragile\pmatrix
182 \latexrelease
183 \latexrelease\EndIncludeInRelease
184 \end{kernel}

```

`\bordermatrix`

```

185 \def\bordermatrix#1{\begingroup \m@th
186   \@tempdima 8.75\p@
187   \setbox\z@\vbox{%
188     \def\crr{\crr\noalign{\kern2\p@\global\let\crr\endline}}%
189     \ialign{##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
190       &&\quad\hfil$##$\hfil\crr
191       \omit\strut\hfil\crr\noalign{\kern-\baselineskip}%
192       #1\crr\omit\strut\crr}%
193   \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
194   \setbox\th@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
195   \setbox\th@\hbox{$\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
196     \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
197     \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,,\right)$}%
198   \null;\vbox{\kern\ht\@ne\box\th@}\endgroup}

```

(End of definition for `\bordermatrix`.)

`\openup`

```

199 \protected\def\openup{\afterassignment\@openup\dimen@}
200 \def\@openup{\advance\lineskip\dimen@
201   \advance\baselineskip\dimen@
202   \advance\lineskiplimit\dimen@}

```

(End of definition for `\openup`.)

`\displaylines`

```

203 \newif\ifdt@p
204 \def\disply{\global\dt@ptrue\openup\jot\m@th
205   \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
206     \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
207     \else \penalty\interdisplaylinepenalty \fi}}}
208 \def\@lign{\tabskip\z@skip\everycr{}} % restore inside \disply
209 \def\displaylines#1{\disply \tabskip\z@skip
210   \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crr
211     #1\crr}}

```

(End of definition for `\displaylines`.)

```

\sp
\sb
212 \let\sp=~
213 \let\sb=_

(End of definition for \sp and \sb.)

\tmspace Originally LATEX only provided a small set of spacing commands for use in text and math,
\ some of the commands like \; were only supported in math mode. amsmath normalized
\tmspace and provided all of them in text and math. This code has now been moved to the kernel
\! so that it is generally available.
\negthinspace
214 </2ekernel>
\;
215 <*2ekernel | latexrelease>
\medspace
216 <latexrelease>\IncludeInRelease{2020/10/01}%
\negmedspace
217 <latexrelease> \tmspace\amsmath spacing commands}%
\;
\tmspace is really meant to be an internal command so it doesn't necessarily has to be
\thickspace robust but it was robust in amsmath so we leave it like that.
\negthickspace
218 \DeclareRobustCommand\tmspace[3]{%
219 \ifmmode\mskip#1#2\else\leavevmode@ifvmode\kern#1#3\fi\relax}

In amsmath the text kern is .1667em. For compatibility reasons we keep the longer one.
220 \DeclareRobustCommand\,{\tmspace+\thinmuskip{.16667em}}
221 \let\thinspace\,
222 \DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
223 \let\negthinspace\!
224 \DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
225 \let\medspace\;
LATEX has a second name for this in its manual:
226 \let\>=\:
227 \DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
228 \DeclareRobustCommand\;{\tmspace+\thickmuskip{.2777em}}
229 \let\thickspace\;
230 \DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
231 </2ekernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease> \tmspace\amsmath spacing commands}%
235 <latexrelease>
236 <latexrelease>\let\tmspace\@undefined
237 <latexrelease>\DeclareRobustCommand{\,,}{%
238 <latexrelease> \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
239 <latexrelease>\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
240 <latexrelease>\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
241 <latexrelease>\def\>{\mskip\medmuskip}
242 <latexrelease>\let\:=\>
243 <latexrelease>\def\;{\mskip\thickmuskip}
244 <latexrelease>\def\!{\mskip-\thinmuskip}
245 <latexrelease>

```

```

246 <latexrelease>\let\negmedspace\@undefined
247 <latexrelease>\let\negthickspace\@undefined
248 <latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <*2ekernel>

(End of definition for \tmspace and others.)

\*
251 \DeclareRobustCommand\*{\discretionary{\thinspace\the\textfont2\char2}{-}{-}}

(End of definition for \*.)

\: Nickname for the medium space since \> is not available inside tabbing.
252 %\let\:=\>

(End of definition for \:.)

\active@math@prime This is the definition of the active math prime.
253 \def\active@math@prime{^{\bgroup\prim@s}}

(End of definition for \active@math@prime.)

\prime@s
254 {\catcode'\='=active \global\let'\active@math@prime}
255 \def\prim@s{%
256   \prime\futurelet\@let@token\pr@m@s}
257 \def\pr@m@s{%
258   \ifx'\@let@token
259     \expandafter\pr@@@s
260   \else
261     \ifx'\@let@token
262       \expandafter\expandafter\expandafter\pr@@@t
263     \else
264       \egroup
265     \fi
266   \fi}
267 \def\pr@@@s#1{\prim@s}
268 \def\pr@@@t#1#2{#2\egroup}

(End of definition for \prime@s.)
269 {\catcode'\_='=active \gdef\_{} } % _ in math is
270                                     % either subscript or \_

```

1.2 Math Environments

`\(` Produces $\dots$ with checks that `\(` isn't used in math mode, and that `\)` is only used in math mode begun with `\(`.

```

271 \endkernel
272 \latexrelease\IncludeInRelease{2015/01/01}{\{()\}{Make \(\ robust}%
273 \*2ekernel | latexrelease
274 \DeclareRobustCommand\({%
275   \relax\ifmmode\@badmath\else$\fi}%
276 \DeclareRobustCommand\){%
277   \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
278 \endkernel | latexrelease
279 \latexrelease\EndIncludeInRelease
280 \latexrelease\IncludeInRelease{0000/00/00}{\{()\}{Make \(\ robust}%
281 \latexrelease\def\({%
282 \latexrelease \relax\ifmmode\@badmath\else$\fi}%
283 \latexrelease\expandafter\let\csname\string( \endcsname\@undefined
284 \latexrelease\def\){%
285 \latexrelease \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
286 \latexrelease\expandafter\let\csname\string) \endcsname\@undefined
287 \latexrelease\EndIncludeInRelease
288 \*2ekernel

```

(End of definition for `\(` and `\)`.)

`\dollar\dollar@begin` Two commands for starting and ending display math formulas to be used in the kernel and by packages. They are needed for tagging support and should be used by all display math environments so that these environments can be used when accessible PDFs are produced. We pretend that they have been always available, so that rollback doesn't break packages that have not set up their own rollback.

```

289 \endkernel
290 \*2ekernel | latexrelease
291 \latexrelease\IncludeInRelease{2025/06/01}%
292 \latexrelease { \dollar\dollar@begin}{Start and end display math}%
293 \def\dollar\dollar@begin{$$}
294 \def\dollar\dollar@end{$$}
295 \endkernel | latexrelease
296 \latexrelease\EndIncludeInRelease
297 \latexrelease\IncludeInRelease{0000/00/00}%
298 \latexrelease { \dollar\dollar@begin}{Start and end display math}%
299 \latexrelease
300 \latexrelease\def\dollar\dollar@begin{$$}
301 \latexrelease\def\dollar\dollar@end{$$}
302 \latexrelease\EndIncludeInRelease
303 \*2ekernel
304
305

```

(End of definition for `\dollar\dollar@begin` and `\dollar\dollar@end`.)

`\[` Produces $\dots$ with checks that `\[` isn't used in math mode, and that `\]` is only used in display math mode (though there is no real test that this display math started with `\[` and not with `$$`).

```

306 </2ekernel>
307 <latexrelease>\IncludeInRelease{2015/01/01}{\[]{}{Make \[ robust}}%
308 <*2ekernel|latexrelease>
309 \DeclareRobustCommand\[%
310   \relax\ifmmode
311     \@badmath
312   \else
313     \ifvmode
314       \nointerlineskip
315       \makebox[.6\linewidth]{}%
316     \fi
317     $$$ % amsthm tries to patch this and expects a $
318 %     \dollar\dollar@begin % in the def so we can't hide it for now
319 %                               % will be adjusted when amsthm changes
320   \fi
321 ]%
322 \DeclareRobustCommand\[%
323   \relax\ifmmode
324     \ifinner
325       \@badmath
326     \else
327       \dollar\dollar@end
328     \fi
329   \else
330     \@badmath
331   \fi
332   \ignorespaces
333 ]%
334 </2ekernel|latexrelease>
335 <latexrelease>\EndIncludeInRelease
336 <latexrelease>\IncludeInRelease{2015/01/01}{\[]{}{Make \[ robust}}%
337 <latexrelease>\DeclareRobustCommand\[%
338 <latexrelease>   \relax\ifmmode
339 <latexrelease>     \@badmath
340 <latexrelease>   \else
341 <latexrelease>     \ifvmode
342 <latexrelease>       \nointerlineskip
343 <latexrelease>       \makebox[.6\linewidth]{}%
344 <latexrelease>     \fi
345 <latexrelease>     $$$%$$$ BRACE MATCH HACK
346 <latexrelease>   \fi
347 <latexrelease>}%
348 <latexrelease>\DeclareRobustCommand\[%
349 <latexrelease>   \relax\ifmmode
350 <latexrelease>     \ifinner
351 <latexrelease>       \@badmath
352 <latexrelease>     \else
353 <latexrelease>       $$$%$$$ BRACE MATCH HACK
354 <latexrelease>     \fi
355 <latexrelease>   \else
356 <latexrelease>     \@badmath
357 <latexrelease>   \fi
358 <latexrelease>   \ignorespaces

```

```

359 <latexrelease>}%
360 <latexrelease>\EndIncludeInRelease

361 <latexrelease>\IncludeInRelease{0000/00/00}{\[]{}{Make \[ robust}}%
362 <latexrelease>\def\[%
363 <latexrelease> \relax\ifmmode
364 <latexrelease> \@badmath
365 <latexrelease> \else
366 <latexrelease> \ifvmode
367 <latexrelease> \nointerlineskip
368 <latexrelease> \makebox[.6\linewidth]{}%
369 <latexrelease> \fi
370 <latexrelease> $$$%$$ BRACE MATCH HACK
371 <latexrelease> \fi
372 <latexrelease>}%
373 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined

374 <latexrelease>\def\[%
375 <latexrelease> \relax\ifmmode
376 <latexrelease> \ifinner
377 <latexrelease> \@badmath
378 <latexrelease> \else
379 <latexrelease> $$$%$$ BRACE MATCH HACK
380 <latexrelease> \fi
381 <latexrelease> \else
382 <latexrelease> \@badmath
383 <latexrelease> \fi
384 <latexrelease> \ignorespaces
385 <latexrelease>}%
386 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined
387 <latexrelease>\EndIncludeInRelease
388 <*2ekernel>

```

(End of definition for \[and \].)

math (*env.*) Disguises for $\left(\dots\right)$ and $\left[\dots\right]$.

```

displaymath (env.)
389 \let\math=\(
390 \let\endmath=\)

391 \def\displaymath{\[]
392 \def\enddisplaymath{\}\@ignoretrue}

```

equation (*env.*) Numbered equations, using the counter `\c@equation`. *Note:* The document style must define `\theequation` etc., and do the appropriate `\@addtoreset`. It should also redefine `\@eqnnum` if another format for the equation number is desired other than the standard (...), or to move the equation numbers to the flushleft. (See comment on the `\def` of `\@eqnnum`.)

```

393 \@definecounter{equation}
394 </2ekernel>
395 <*2ekernel | latexrelease>
396 <latexrelease>\IncludeInRelease{2025/06/01}%
397 <latexrelease> \{equation\}{Tagging support}}%
398 \def\equation{\dollar@begin
399 \refstepcounter{equation}}
400 \def\endequation{\eqno \hbox{\@eqnnum}\dollar@end\@ignoretrue}

```

```

401 </2ekernel | latexrelease>
402 <latexrelease>\EndIncludeInRelease
403 <latexrelease>\IncludeInRelease{0000/00/00}%
404 <latexrelease>{\equation}{Tagging support}%
405 <latexrelease>
406 <latexrelease>\def\equation{$$\refstepcounter{equation}}
407 <latexrelease>\def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}
408 <latexrelease>\EndIncludeInRelease
409 <*2ekernel>
(End of definition for \c@equation.)

\@eqnnum Produces the equation number for equation and eqnarray environments. The following
definition is for flushright numbers; for flushleft numbers, see leqno.clo. The equation
number is set in black roman type even if an eqnarray environment appears in an italic
environment.
410 \def\@eqnnum{\normalfont \normalcolor (\theequation)}
(End of definition for \@eqnnum.)

\stackrel A disguise for plain TEX's buildrel.
411 \DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{#2}\limits^{#1}}}}
(End of definition for \stackrel.)

\frac A disguise for plain TEX's \over.
412 \DeclareRobustCommand\frac[2]{\begingroup#1\endgroup\over#2}}
(End of definition for \frac.)

\sqrt Add an optional argument to plain's \sqrt to give the nth root of an expression  $\sqrt[n]{e}$ .
\@sqrt
413 \DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
414 \def\@sqrt[#1]{\root #1\of}
(End of definition for \sqrt and \@sqrt.)

\eqnarray Here's the eqnarray environment: Default is for left-hand side of equations to be
flushright. To make them flushleft, \let\@eqnrel = \hfil.
\@eqnrel
\@eqpen
415 \newcount\@eqcnt
\if@eqnsw
416 \newcount\@eqpen
417 \newif\if@eqnsw\@eqnswtrue
418 \newskip\@centering
419 \@centering = 0pt plus 1000pt
To get a proper \@currentlabel we have to redefine it for the whole display. Note that
we can't use \refstepcounter as this results in \@currentlabel getting restored at the
wrong and thus always writing the first label to the .aux file.
420 </2ekernel>
421 <*2ekernel | latexrelease>
422 <latexrelease>\IncludeInRelease{2025/06/01}%
423 <latexrelease>{\eqnarray}{Support for tagging}%

```

```

424 \def\eqnarray{%
425     \stepcounter{equation}%
426     \def\@currentlabel{\p@equation\theequation}%
427     \def\@currentcounter{equation}%
428     \global\@eqnswtrue
429     \m@th
430     \global\@eqcnt\z@
431     \tabskip\@centering
432     \let\\\@eqnocr
433     \dollar\dollar\@begin\everycr{}\halign to\displaywidth\bgroup
434         \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnsele
435         &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
436         &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
437         $\displaystyle{##}$\hfil\tabskip\@centering
438         &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
439         \tabskip\z@skip
440     \cr
441 }

442 \def\endeqnarray{%
443     \@@eqnocr
444     \egroup
445     \global\advance\c@equation\m@ne
446     \dollar\dollar\@end\@ignoretrue
447 }

448 </2ekernel | latexrelease>
449 <latexrelease>\EndIncludeInRelease
450 <latexrelease>\IncludeInRelease{0000/00/00}%
451 <latexrelease>          {\eqnarray}{Support for tagging}%
452 <latexrelease>
453 <latexrelease>\def\eqnarray{%
454 <latexrelease>     \stepcounter{equation}%
455 <latexrelease>     \def\@currentlabel{\p@equation\theequation}%
456 <latexrelease>     \def\@currentcounter{equation}%
457 <latexrelease>     \global\@eqnswtrue
458 <latexrelease>     \m@th
459 <latexrelease>     \global\@eqcnt\z@
460 <latexrelease>     \tabskip\@centering
461 <latexrelease>     \let\\\@eqnocr
462 <latexrelease>     $$\everycr{}\halign to\displaywidth\bgroup
463 <latexrelease>         \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnsele
464 <latexrelease>         &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
465 <latexrelease>         &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
466 <latexrelease>         $\displaystyle{##}$\hfil\tabskip\@centering
467 <latexrelease>         &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
468 <latexrelease>         \tabskip\z@skip
469 <latexrelease>     \cr
470 <latexrelease>}
471 <latexrelease>\def\endeqnarray{%
472 <latexrelease>     \@@eqnocr
473 <latexrelease>     \egroup
474 <latexrelease>     \global\advance\c@equation\m@ne
475 <latexrelease>     $$\@ignoretrue
476 <latexrelease>}

```



```

477 <latexrelease>\EndIncludeInRelease
478 <*2ekernel>

479 \let\@eqnrel=\relax
(End of definition for \@eqcnt and others.)

\nonumber Switches off equation numbering.
480 \def\nonumber{\global\@eqnswfalse}
(End of definition for \nonumber.)

\@eqnrel
\@xeqnrel 481 \protected\def\@eqnrel{%
\@yeqnrel 482 {\ifnum0='}\fi
483 \@ifstar{%
484 \global\@eqpen\@M\@yeqnrel
485 }{%
486 \global\@eqpen\interdisplaylinepenalty \@yeqnrel
487 }%
488 }

489 \def\@yeqnrel{\@testopt\@xeqnrel\z@skip}

490 </2ekernel>
491 <*2ekernel | latexrelease>
492 <latexrelease>\IncludeInRelease{2020/10/01}%
493 <latexrelease> {\@xeqnrel}{eqnarray support calc syntax}%
494 \def\@xeqnrel[#1]{%
495 \ifnum0='{ \fi}%
496 \@eqnrel
497 \noalign{\penalty\@eqpen\vskip\jot\@vspace@calcify{#1}}%
498 }
499 </2ekernel | latexrelease>
500 <latexrelease>\EndIncludeInRelease

501 <latexrelease>\IncludeInRelease{0000/00/00}%
502 <latexrelease> {\@xeqnrel}{eqnarray support calc syntax}%
503 <latexrelease>
504 <latexrelease>\def\@xeqnrel[#1]{%
505 <latexrelease> \ifnum0='{ \fi}%
506 <latexrelease> \@eqnrel
507 <latexrelease> \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
508 <latexrelease>}
509 <latexrelease>\EndIncludeInRelease
510 <*2ekernel>

(End of definition for \@eqnrel, \@xeqnrel, and \@yeqnrel.)

\@eqnrel
511 \def\@eqnrel{\let\reserved@a\relax
512 \ifcase\@eqcnt \def\reserved@a{& &}\or \def\reserved@a{& &}%
513 \or \def\reserved@a{} \else
514 \let\reserved@a\empty
515 \latexerror{Too many columns in eqnarray environment}\@ehc\fi
516 \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
517 \global\@eqnswtrue\global\@eqcnt\z@\cr}

```

(End of definition for `\@eqnocr`.)

`\@eqnocr`) Here's the `eqnarray*` environment:

```
518 \let\@seqnocr=\@eqnocr
519 \@namedef{eqnarray*}{\protected\def\@eqnocr{\nonumber\@seqnocr}\eqnarray}
520 \@namedef{endeqnarray*}{\nonumber\endeqnarray}
(End of definition for \@seqnocr.)
```

`\lefteqn` `\lefteqn{FORMULA}` typesets FORMULA in display math style flushleft in a box of width zero.

```
521 \def\lefteqn#1{\rlap{$\displaystyle #1$}}
(End of definition for \lefteqn.)
```

`\ensuremath` In math mode, `\ensuremath{text}` is equivalent to `text`; in LR or paragraph mode, it is equivalent to `$text$`. `\relax` is not needed in front of the `\ifmmode` as `\protect` will be `\let` to `\relax`. This version (due to Donald Arseneau) avoids duplicating its argument in the ‘then’ and ‘else’ part of the `\ifmath` which is necessary in nested ‘tabular’ like environments. See `amslatex/2104`.

```
522 \DeclareRobustCommand{\ensuremath}{%
523   \ifmmode
524     \expandafter\@firstofone
525   \else
526     \expandafter\@ensuredmath
527   \fi}
```

(End of definition for `\ensuremath`.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
528 \long\def\@ensuredmath#1{$\relax#1$}
(End of definition for \@ensuredmath.)
```

Lua_{TeX} contains new math primitives to place expression over or under horizontally extensible glyphs. Before Lua_{TeX} 1.14 these did not work correctly with the `\mathstyle` primitive and sometimes did not use cramped style in consistent ways. For newer version, we opt into the corrected behavior.

```
529 \ifx\mathdefaultsmode\undefined\else
530   \mathdefaultsmode=1
531 \fi
```

`\eqno` Ensure the (deprecated) `$$..\eqno 1 $$` ignores spaces.

```
\leqno 532 \</2kernel>
533 \< *2kernel | latexrelease>
534 \< latexrelease> \IncludeInRelease{2023/06/01}%
535 \< latexrelease> { \eqno } { add ignorespaces to eqno } %
536 \expandafter \let \expandafter \@kernel@eqno \csname tex_eqno:D \endcsname
537 \expandafter \let \expandafter \@kernel@leqno \csname tex_leqno:D \endcsname
538 \protected\def \eqno { \@kernel@eqno \aftergroup \ignorespaces }
539 \protected\def \leqno { \@kernel@leqno \aftergroup \ignorespaces }
540 \< /2kernel | latexrelease>
541 \< latexrelease> \EndIncludeInRelease
542 \< latexrelease> \IncludeInRelease{0000/00/00}%
```

```

543 <latexrelease>                {\eqno}{add ignorespaces to eqno}%
544 <latexrelease>\let\eqno\@kernel@eqno
545 <latexrelease>\let\leqno\@kernel@leqno
546 <latexrelease>\EndIncludeInRelease

```

(End of definition for \eqno and \leqno.)

1.3 External options to the standard document classes

1.3.1 Left equation numbering

\@eqnnum To put the equation number on the left side of an equation we have to use a little trick. The number is shifted `\displaywidth` to the left inside a box of (approximately) zero width. This fails when the equation is too wide, the equation number than may overprint the equation itself.

```

547 <*\leqno>
548 \renewcommand\@eqnnum{\hb@xt@.01\p@{}}%
549                               \rlap{\normalfont\normalcolor
550                               \hskip -\displaywidth(\theequation)}}
551 </\leqno>

```

(End of definition for \@eqnnum.)

1.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of L^AT_EX 2_ε's displayed math environments.

\mathindent The amount of indentation of the equations is stored in a register.

```

552 <*\fleqn>
553 \newskip\mathindent

```

The setting of `\mathindent` has to be deferred until the class file has been processed, because `\leftmargini` is still 0pt wide at the moment `fleqn.clo` is read in.

```

554 \AtEndOfClass{\mathindent\leftmargini}

```

(End of definition for \mathindent.)

\[Begin display math;

```

555 \IncludeInRelease{2015/01/01}{\[\]{Make \[ robust}%
556 \DeclareRobustCommand\[\relax
557                               \ifmmode\@badmath
558                               \else
559                               \begin{trivlist}%
560                               \@beginparpenalty\predisplaypenalty
561                               \@endparpenalty\postdisplaypenalty
562                               \item[]\leavevmode
563                               \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
564                               \hskip\mathindent\bgroup
565                               \fi}
566 \EndIncludeInRelease

```

```

567 \IncludeInRelease{0000/00/00}{\[]}{Make \[ robust}%
568 \renewcommand\[\relax
569     \ifmmode\@badmath
570     \else
571     \begin{trivlist}%
572     \@beginparpenalty\predisplaypenalty
573     \@endparpenalty\postdisplaypenalty
574     \item[]\leavevmode
575     \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
576     \hskip\mathindent\bgroup
577     \fi}
578 \EndIncludeInRelease

```

(End of definition for \[.)

\] end display math;

```

579 \IncludeInRelease{2015/01/01}{\[]}{Make \] robust}%
580 \DeclareRobustCommand\]\relax
581     \ifmmode
582     \egroup $\hfil% $
583     \egroup
584     \end{trivlist}%
585     \else \@badmath
586     \fi}
587 \EndIncludeInRelease

588 \IncludeInRelease{0000/00/00}{\[]}{Make \] robust}%
589 \renewcommand\]\relax
590     \ifmmode
591     \egroup $\hfil% $
592     \egroup
593     \end{trivlist}%
594     \else \@badmath
595     \fi}
596 \EndIncludeInRelease

```

(End of definition for \].)

equation (*env.*) The equation environment

```

597 \renewenvironment{equation}%
598 {\@beginparpenalty\predisplaypenalty
599 \@endparpenalty\postdisplaypenalty
600 \refstepcounter{equation}%
601 \trivlist \item[]\leavevmode
602 \hb@xt@\linewidth\bgroup $\m@th% $
603 \displaystyle
604 \hskip\mathindent}%

```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```

605 {\$ \hskip .3em minus.3em\hfil % $
606 \displaywidth\linewidth\hbox{\@eqnnum}%
607 \egroup
608 \endtrivlist}

```

`eqnarray` (*env.*) The `eqnarray` environment

```

609 \renewenvironment{eqnarray}{%
610   \stepcounter{equation}%
611   \def\@currentlabel{\p@equation\theequation}%
612   \def\@currentcounter{equation}%
613   \global\@eqnswtrue\m@th
614   \global\@eqcnt\z@
615   \tabskip\mathindent
616   \let\@=\@eqncr
617   \setlength\abovedisplayskip{\topsep}%
618   \ifvmode
619     \addtolength\abovedisplayskip{\partopsep}%
620   \fi

```

When the documentclass uses a non-zero `\parskip` setting the `\topsep` might have a negative value to compensate for that. Therefore we add `\parskip` to `\abovedisplayskip`.

```

621   \addtolength\abovedisplayskip{\parskip}%
622   \setlength\belowdisplayskip{\abovedisplayskip}%
623   \setlength\belowdisplayshortskip{\abovedisplayskip}%
624   \setlength\abovedisplayshortskip{\abovedisplayskip}%
625   \dollar\dollar@begin\everycr{}\halign to\linewidth%
626   \bgroup
627     \hskip\@centering
628     $\displaystyle\tabskip\z@skip{##}$\@eqnse1&%
629     \global\@eqcnt\@ne \hskip \tw@arraycolsep \hfil${##}$\hfil&%
630     \global\@eqcnt\tw@ \hskip \tw@arraycolsep
631     $\displaystyle{##}$\hfil \tabskip\@centering&%
632     \global\@eqcnt\thr@@
633     \hb@xt@\z@\bgroup\hss#\egroup\tabskip\z@skip\cr}%
634     {\@eqncr
635   \egroup
636   \global\advance\c@equation\m@ne\dollar\dollar@end
637   \@ignoretrue
638   }
639 \fleqn

```

File 39

ltlists.dtx

1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

```
\list{<LABEL>}{<COMMANDS>} ... \endlist
```

which can be invoked by the user as the list environment. The LABEL argument specifies item labeling. COMMANDS contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by ITEMLABEL. If the argument is missing, then the LABEL argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{<ITEMLABEL>}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

```
\makelabel{<ARG>} == BEGIN \hfil ARG END
```

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s COMMANDS argument.

A `\usecounter{<foo>}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place–i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation == \list{}{...}\item\relax`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexibility in designing the format, including the ability to let the indentation of the first paragraph be different from that of the subsequent ones.

The trivlist environment is equivalent to a list environment whose second argument sets the following parameter values:

`\leftmargin = 0`: causes no indentation of left margin

`\labelwidth = 0`: see below for precise effect this has.

`\itemindent = 0`: with a null label, makes first paragraph have no indentation. Succeeding paragraphs have `\parindent` indentation. To give first paragraph same indentation, set `\itemindent = \parindent` before the `\item[]`.

Every `\item` in a `trivlist` environment must have an argument—in many cases, this will be the null argument (`\item[]`). The `trivlist` environment is mainly used for paragraphing environments, like `verbatim`, in which there is no margin change. It provides the same vertical spacing as the `list` environment, and works reasonably well when it occurs immediately after an `\item` command in an enclosing list.

1.1 List and Trivlist

The following variables are used inside a list environment:

`\totalleftmargin` The distance that the prevailing left margin is indented from the outermost left margin,

`\linewidth` The width of the current line. Must be initialized to `\hsize`.

`\listdepth` A count for holding current list nesting depth.

`\makelabel` A macro with a single argument, used to generate the label from the argument (given or implied) of the `\item` command. Initialized to `\mklab` by the `\list` command. This command must produce some stretch—i.e., an `\hfil`.

`@inlabel` A switch that is false except between the time an `\item` is encountered and the time that `TeX` actually enters horizontal mode. Should be tested by commands that can be messed up by the list environment's use of `\everypar`.

`\box\@labels` When `@inlabel = true`, it holds the labels to be put out by `\everypar`.

`@noparitem` A switch set by `\list` when `@inlabel = true`. Handles the case of a `\list` being the first thing in an item.

`@nopalist` A switch set true for a list that begins an item. No `\topsep` space is added before or after `\item`'s such a list.

`@newlist` Set true by `\list`, set false by the first text (by `\everypar`).

`@noitemarg` Set true when executing an `\item` with no explicit argument. Used to save space. To save time, make two separate `\@item` commands.

`@nbrlist` Set true by `\usecounter` command, causes list to be numbered.

`\@listctr` `\def`'ed by `\usecounter` to name of counter.

`@noskipsec` A switch set true by a sectioning command when it is creating an in-text heading with `\everypar`.

Throughout a list environment, `\hsize` is the width of the current line, measured from the outermost left margin to the outermost right margin. Environments like `tabbing` should use `\linewidth` instead of `\hsize`.

Here are the parameters of a list that can be set by commands in the `\list's` `COMMANDS` argument. These parameters are all `TeX` skips or dimensions (defined by `\newskip` or `\newdimen`), so the usual `TeX` or `LATeX` commands can be used to set them. The commands will be executed in `vmode` if and only if the `\list` was preceded by a `\par` (or something like an `\end{list}`), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

1.2 Vertical Spacing (skips)

`\topsep`: Space between first item and preceding paragraph.

`\partopsep`: Extra space added to `\topsep` when environment starts a new paragraph (is called in `vmode`).

`\itemsep`: Space between successive items.

`\parsep`: Space between paragraphs within an item – the `\parskip` for this environment.

1.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

1.4 Horizontal Spacing (dimens)

`\leftmargin`: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

`\rightmargin`: analogous.

`\listparindent`: extra indentation at beginning of every paragraph of a list except the one started by the `\item` command. May be negative! Usually, labeled lists have `\listparindent` equal to zero.

`\itemindent`: extra indentation added right BEFORE an item label.

`\labelwidth`: nominal width of box that contains the label. If the natural width of the label $\leq \text{\labelwidth}$, then the label is flushed right inside a box of width `\labelwidth` (with an `\hfil`). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep`: space between end of label box and text of first item.

1.5 Default Values

Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, one of the commands `\@listi`, `\@listii`, ... , `\@listvi` is called, depending upon the current level of the list. The `\@list ...` commands should be defined by the document style. A convention that the document style should follow is to set `\leftmargin` to `\leftmargini`, ..., `\leftmarginvi` for the appropriate level. Items that aren't changed may be left alone, but everything that could possibly be changed must be reset. *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```
\list{LABEL}{COMMANDS} ==  
  BEGIN  
    if \@listdepth > 5  
      then LaTeX error: 'Too deeply nested'  
      else \@listdepth :=G \@listdepth + 1
```



```

fi
\rightmargin      := 0pt
\listparindent    := 0pt
\itemindent       := 0pt
\eval{@list \romannumeral\the\@listdepth} %% Set default values:
\itemlabel        :=L LABEL
\makelabel        == \@mklab
@nmbrlist         :=L false
COMMANDS

\@trivlist          % commands common to \list and \trivlist

\parskip           :=L \parsep
\parindent         :=L \listparindent
\linewidth         :=L \linewidth - \rightmargin - \leftmargin
\@totalleftmargin :=L \@totalleftmargin + \leftmargin
\parshape 1 \@totalleftmargin \linewidth
\ignorespaces      % gobble space up to \item
END

\endlist == BEGIN \@listdepth :=G \@listdepth -1
              \endtrivlist
              END

\@trivlist ==
BEGIN
  if @newlist = T then \@noitemerr fi
                      %% This command removed for some forgotten reason.
  \@topsepadd :=L \topsep
  if @noskipsec then leave vertical mode fi %% Added 11 Jun 85
  if vertical mode
  then \@topsepadd :=L \@topsepadd + \partopsep
  else \unskip \par % remove glue from end of last line
  fi
  if @inlabel = true
  then @noparitem :=L true
       @noparlist :=L true
  else @noparlist :=L false
       \@topsep   :=L \@topsepadd
  fi
  \@topsep      :=L \@topsep + \parskip %% Change 4 Sep 85
  \leftskip     :=L 0pt % Restore paragraphing parameters
  \rightskip    :=L \@rightskip
  \parfillskip  :=L 0pt + 1fil

NOTE: \@setpar called on every \list in case \par has been
temporarily munged before the \list command.
\@setpar{if @newlist = false then {\@@par} fi}
\@newlist      :=G T
\@outerparskip :=L \parskip

```

```

END

\trivlist ==
BEGIN
  \parsep      := \parskip
  @nmbrlist := F
  \@trivlist
  \labelwidth := 0
  \leftmargin := 0
  \itemindent := \parindent
  \@itemlabel :=L "empty"      %% added 93/12/13
  \makelabel{LABEL} == LABEL
END

\endtrivlist ==
BEGIN
  if @inlabel = T then \indent fi
  if horizontal mode then \unskip \par fi
  if @noparlist = true
    else if \lastskip > 0
      then \@tempkipa := \lastskip
           \vskip - \lastskip
           \vskip \@tempkipa -\@outerparskip + \parskip
        fi
      \@endparenv
    fi
  END

\@endparenv ==
BEGIN
  \addpenalty{@endparpenalty}
  \addvspace{@topsepadd}
  \endgroup    %% ends the \begin command's \begingroup
  \par == BEGIN
    \@restorepar
    \everypar{}
    \par
  END
  \everypar == BEGIN remove \lastbox \everypar{} END
  \begingroup %% to match the \end commands \endgroup
END

\item == BEGIN if math mode then WARNING fi
          if next char = [
            then \@item
            else @noitemarg := true
                 \@item[@itemlabel]
          END

\@item[LAB] ==

```

```

BEGIN
if @noperitem = true
then @noperitem := false
    % NOTE: then clause hardly every taken,
    % so made a macro \@donoperitem
    \box\@labels :=G \hbox{\hskip -\leftmargin
                        \box\@labels
                        \hskip \leftmargin }

if @minipage = false then
    \@tempskipa := \lastskip
    \vskip -\lastskip
    \vskip \@tempskipa + \@outerparskip - \parskip
fi
else if @inlabel = true
then \indent \par % previous item empty.
fi
if hmode then 2 \unskip's
    % To remove any space at end of prev.
    % paragraph that could cause a blank line.
    \par
fi
if @newlist = T
then if @nobreak = T % Kludge if list follows \section
then \addvspace{\@outerparskip - \parskip}
else \addpenalty{\@beginparpenalty}
    \addvspace{\@topsep}
    \addvspace{-\parskip} %% added 4 Sep 85
fi
else \addpenalty{\@itempenalty}
    \addvspace{\itemsep}
fi
@inlabel :=G true
fi

\everypar{ @minipage :=G F
@newlist :=G F
if @inlabel = true
then @inlabel :=G false
    \hskip -\parindent
    \box\@labels
    \penalty 0
    %% 3 Oct 85 - allow line break here
    \box\@labels :=G null
fi
\everypar{} }

@nobreak :=G false
if @noitemarg = true
then @noitemarg := false
if @nmbrlist
then \refstepcounter{\@listctr}

```

```

fi      fi
\@tempboxa :=L \hbox{\makelabel{LAB}}
\box\@labels :=G \@labels \hskip \itemindent
               \hskip - (\labelwidth + \labelsep)
               if \wd \@tempboxa > \labelwidth
                   then \box\@tempboxa
                   else \hbox to \labelwidth {\unhbox\@tempboxa}
               fi
               \hskip\labelsep
\ignorespaces %gobble space up to text
END

```

```

\makelabel{LABEL} == ERROR %% default to catch lonely \item

```

```

\usecounter{CTR} == BEGIN @nmbrlist :=L true
                        \@listctr == CTR
                        \setcounter{CTR}{0}
                        END

```

DEFINE \dimen's and \count
End of historical L^AT_EX 2.09 comments.

```

\topsep
\partopsep 1 (*2kernel)
\itemsep    2 \newskip\topsep
\parsep     3 \newskip\partopsep
\@topsep    4 \newskip\itemsep
\@topsepadd 5 \newskip\parsep
\@outerparskip 6 \newskip\@topsep
              7 \newskip\@topsepadd
              8 \newskip\@outerparskip

```

(End of definition for \topsep and others.)

```

\leftmargin
\rightmargin 9 \newdimen\leftmargin
\listparindent 10 \newdimen\rightmargin
\itemindent   11 \newdimen\listparindent
\labelwidth   12 \newdimen\itemindent
\labelsep     13 \newdimen\labelwidth
\linewidth    14 \newdimen\labelsep
\@totalleftmargin 15 \newdimen\linewidth
               16 \newdimen\@totalleftmargin \@totalleftmargin=\z@

```

(End of definition for \leftmargin and others.)

```

\leftmargini
\leftmarginii 17 \newdimen\leftmargini
\leftmarginiii 18 \newdimen\leftmarginii
\leftmarginiv 19 \newdimen\leftmarginiii
\leftmarginv 20 \newdimen\leftmarginiv
\leftmarginvi 21 \newdimen\leftmarginv
                22 \newdimen\leftmarginvi

                (End of definition for \leftmargini and others.)

\@listdepth
\@itempenalty 23 \newcount\@listdepth \@listdepth=0
\@beginparpenalty 24 \newcount\@itempenalty
\@endparpenalty 25 \newcount\@beginparpenalty
                26 \newcount\@endparpenalty

                (End of definition for \@listdepth and others.)

\@labels
                27 \newbox\@labels

                (End of definition for \@labels.)

\if@inlabel
\@inlabelfalse 28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue
                (End of definition for \if@inlabel, \@inlabelfalse, and \@inlabeltrue.)

\if@newlist
\@newlistfalse 29 \newif\if@newlist \@newlistfalse
\@newlisttrue
                (End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

\if@noparitem
\@noparitemfalse 30 \newif\if@noparitem \@noparitemfalse
\@noparitemtrue
                (End of definition for \if@noparitem, \@noparitemfalse, and \@noparitemtrue.)

\if@noparlist
\@noparlistfalse 31 \newif\if@noparlist \@noparlistfalse
\@noparlisttrue
                (End of definition for \if@noparlist, \@noparlistfalse, and \@noparlisttrue.)

\if@noitemarg
\@noitemargfalse 32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue
                (End of definition for \if@noitemarg, \@noitemargfalse, and \@noitemargtrue.)

\if@newlist
\@newlistfalse 33 \newif\if@nmbrlist \@nmbrlistfalse
\@newlisttrue
                (End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

```

`list\list.)`

```

34 \def\list#1#2{%
35   \ifnum \@listdepth >5\relax
36     \toodeep
37   \else
38     \global\advance\@listdepth\@ne
39   \fi
40   \rightmargin\z@
41   \listparindent\z@
42   \itemindent\z@
43   \csname @list\romannumeral\the\@listdepth\endcsname
44   \def\@itemlabel{#1}%
45   \let\makelabel\@mklab
46   \@nmbrlistfalse
47   #2\relax
48   \@trivlist
49   \parskip\parsep
50   \parindent\listparindent
51   \advance\linewidth -\rightmargin
52   \advance\linewidth -\leftmargin
53   \advance\@totalleftmargin \leftmargin
54   \parshape \@ne \@totalleftmargin \linewidth
55   \ignorespaces}

```

(End of definition for \list.)

`\par@deathcycles`

```

56 \newcount\par@deathcycles

```

(End of definition for \par@deathcycles.)

`\@trivlist` Because `\par` is sometimes made a no-op it is possible for a missing `\item` to produce a loop that does not fill memory and so never gets trapped by T_EX. We thus need to trap this here by setting `\par` to count the number of times a paragraph is called with no progress being made started.

```

57 \def\@trivlist{%
58   \if@noskipsec \leavevmode \fi
59   \@topsepadd \topsep
60   \ifvmode
61     \advance\@topsepadd \partopsep
62   \else
63     \unskip \par
64   \fi
65   \if@inlabel
66     \@nparitemtrue
67     \@nparlisttrue
68   \else
69     \if@newlist \@noitemerr \fi
70     \@nparlistfalse
71     \@topsep \@topsepadd
72   \fi
73   \advance\@topsep \parskip
74   \leftskip \z@skip
75   \rightskip \@rightskip
76   \parfillskip \@flushglue

```

```

77 \par@deathcycles \z@
78 \@setpar{\if@newlist
79     \advance\par@deathcycles \@ne
80     \ifnum \par@deathcycles >\@m
81         \@noitemerr
82         {\@@par}%
83     \fi
84     \else
85         {\@@par}%
86     \fi}%
87 \global \@newlisttrue
88 \@outerparskip \parskip}

```

(End of definition for \@trivlist.)

~~trivlist~~ *trivlist*

```

89 \def\trivlist{%
90     \parsep\parskip
91     \@nmbrlistfalse
92     \@trivlist
93     \labelwidth\z@
94     \leftmargin\z@
95     \itemindent\z@

```

We initialize \@itemlabel so that a trivlist with an \item not having an optional argument doesn't produce an error message.

```

96 \let\@itemlabel\@empty
97 \def\makelabel##1{##1}}

```

(End of definition for \trivlist.)

\endlist

```

98 \def\endlist{%
99     \global\advance\@listdepth\m@ne
100     \endtrivlist}

```

(End of definition for \endlist.)

The definition of \trivlist used to be in ltspc.dtx so that other commands could be 'let to it'. They now use \def.

\endtrivlist

```

101 \def\endtrivlist{%
102     \if@inlabel
103         \leavevmode
104         \global \@inlabelfalse
105     \fi
106     \if@newlist
107         \@noitemerr
108         \global \@newlistfalse
109     \fi
110     \ifhmode\unskip \par

```

We also check if we are in math mode and issue an error message if so (hoping that \@currentvir resolves suitably). Otherwise the usual “perhaps a missing item” error will get triggered later which is confusing.

```

111 \else

```

```

112     \inmatherr{\end{\@currenvir}}}%
113   \fi
114   \if@nolist \else
115     \ifdim\lastskip >\z@
116       \@tempskipa\lastskip \vskip -\lastskip
117       \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
118       \vskip\@tempskipa
119     \fi
120   \endparenv
121 \fi
122 }

```

(End of definition for `\endtrivlist`.)

`\@endparenv` To suppress the paragraph indentation in text immediately following a paragraph-making environment, `\everypar` is changed to remove the space, and `\par` is redefined to restore `\everypar`. Instead of redefining `\par` and `\everypar`, `\@endparenv` was changed to set the `@endpe` switch, letting `\end` redefine `\par` and `\everypar`.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

In 2024 this logic was partially replaced with a new algorithm:

- `\if@endpe` is now set globally to `true` or `false`.
- In addition `\@endpetrue` initiates an `\aftergroup` call to `\propagate@doendpe` if it is used inside a group.
- `\propagate@doendpe` in turn checks the status of `\if@endpe` and if that is `true` it calls `\@doendpe` otherwise it does nothing.
- `\@doendpe` in turn calls `\@endpetrue` and also makes the necessary changes to `\par` and `\everypar` so that they handle as before any empty line that follows the environment.
- Because of the `\@endpetrue` we get another `\aftergroup`, so the mechanism slowly migrates out of several groups if those follow immediately after the end of the environment. If, however, there is a new paragraph started or an explicit `\par` before the next group ends then this will result in a call to `\@endpefalse` and the migration stops (note that `\propagate@doendpe` is still called once after the group but does nothing).
- Using this approach something like

```

{% some customization here
\begin{equation}
  x=y
\end{equation}}
some text

```

is still correctly identified as a paragraph continuation so that there is no indentation before `some text`.

- We can get away with using global settings of `\if@endpe` even in nested situations (without keeping track of the status in a stack), because the switch change is made only at the very end of such environments, basically directly before the `\endgroup` in `\end`, and it is later set back to `false` by the next `\everypar` or the next `\par`. Even if the environment is called without using `\begin \end`, the situation doesn't change (or rather cause a problem).
- However, there is one scenario where the new approach would change the behavior. If a box is being built, e.g., with `\setbox`, we have now the case that a `\@endpetrue` inside would migrate out into a context in which it should not be true (because the box might get used elsewhere, e.g., a float). In the past, due to local switch changes, that didn't happen, i.e., `\if@endpe` would revert to `false` at the end of the box definition.
- Thus, to avoid that one has to explicitly set it back to `false` at the end of such constructions, just as we also need to prevent colors from migrating out. Thus the correct place to do this is in `\color@endgroup` because that is always called at that point.

```

123 \def\@endparenv{%
124   \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
125 \<latexrelease>\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
126 \def\@doendpe{\@endpetrue
127   \def\par{\@restorepar

```

If a section heading changes `\clubpenalty` to keep lines after it together then this modification is restored via the `\everypar` mechanism at the start of the next paragraph. As we destroy the contents of this token here we explicitly set `\clubpenalty` back to its default.

```

128       \clubpenalty\@clubpenalty
129       \@endpefalse
130       \everypar{}\par}%

```

Use `\setbox0=\lastbox` instead of `\hskip -\parindent` so that a `\noindent` becomes a no-op when used before a line immediately following a list environment(23 Oct 86).

```

131   \everypar
132     {\setbox\z@\lastbox}%
133     \everypar{}\@endpefalse}}
134 \<latexrelease>\EndIncludeInRelease
135 \<latexrelease>\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
136 \<latexrelease>\def\@doendpe{\@endpetrue
137 \<latexrelease>   \def\par{\@restorepar\everypar{}\par\@endpefalse}\everypar
138 \<latexrelease>     {\setbox\z@\lastbox}\everypar{}\@endpefalse}}
139 \<latexrelease>\EndIncludeInRelease

```

(End of definition for \@endparenv and \@doendpe.)

```

\if@endpe As outlined above these are no longer simple switches, but we keep the name because it
\@endpefalse is used all over the place.
\@endpetrue
\propagate@doendpe 140 \newif\if@endpe

```

```

141 </2ekernel>
142 <*2ekernel | latexrelease>
143 <latexrelease>\IncludeInRelease{2024/11/01}%
144 <latexrelease>                {\@endpetrue}{New @endpe handling}%
145 \def\@endpefalse{\global\let\if@endpe\iffalse}
146 \def\@endpetrue {%
147   \global\let\if@endpe\iftrue

```

If we are inside a simple or a semi-simple group then propagate to the outside, for all other group types do nothing. Normally, we would start out in the group opened by `\begin` (type 14). When we migrate out of that we are either on top-level (type 0) or in another semi-simple group (type 14) or in some other group. Thus, the best order of tests is to first test for 14, then for 0 and finally for 1 (simple group).

```

148   \ifnum\currentgrouptype =14          % semi-simple group
149     \aftergroup\propagate@doendpe
150   \else
151     \ifnum\currentgrouptype =\z@        % no group: top-level
152     \else
153       \ifnum\currentgrouptype =\@ne    % simple group
154       \aftergroup\propagate@doendpe
155     \fi
156   \fi
157 \fi
158 }

```

If `\if@endpe` is still true after the group ends, we run `\@doendpe` that in turn runs another `\@endpetrue` (besides other things), thus propagating further if necessary. However, if the endpe situation got resolved and `\if@endpe` is false then nothing further happens.

```

159 \def\propagate@doendpe{\if@endpe \@doendpe \fi}
160 </2ekernel | latexrelease>
161 <latexrelease>\EndIncludeInRelease
162 <latexrelease>\IncludeInRelease{0000/00/00}%
163 <latexrelease>                {\@endpetrue}{New @endpe handling}%
164 <latexrelease>
165 <latexrelease>
166 <latexrelease>\def\@endpefalse{\let\if@endpe\iffalse}
167 <latexrelease>\def\@endpetrue{\let\if@endpe\iftrue}
168 <latexrelease>
169 <latexrelease>\EndIncludeInRelease
170 <*2ekernel>

```

(End of definition for \if@endpe and others.)

`\@mklab`

```

171 \def\@mklab#1{\hfil #1}

```

(End of definition for \@mklab.)

`\item`

```

172 \def\item{%
173   \@inmatherr\item
174   \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}

```

(End of definition for \item.)

`\@donoparitem`

```

175 \def\@donoparitem{%
176   \@noparitemfalse
177   \global\setbox\@labels\hbox{\hskip -\leftmargin
178                                 \unhbox\@labels
179                                 \hskip \leftmargin}%
180   \if@minipage\else
181     \@tempskipa\lastskip
182     \vskip -\lastskip
183     \advance\@tempskipa\@outerparskip
184     \advance\@tempskipa -\parskip
185     \vskip\@tempskipa
186   \fi}

```

(End of definition for \@donoparitem.)

`\@item`

```

187 \def\@item[#1]{%
188   \if@noparitem
189     \@donoparitem
190   \else
191     \if@inlabel
192       \indent \par
193     \fi
194     \ifhmode
195       \unskip\unskip \par
196     \fi
197     \if@newlist
198       \if@nobreak
199         \@nbitem
200       \else
201         \addpenalty\@beginparpenalty
202         \addvspace\@topsep
203         \addvspace{-\parskip}%
204       \fi
205     \else
206       \addpenalty\@itempenalty
207       \addvspace\itemsep
208     \fi
209     \global\@inlabeltrue
210   \fi
211   \everypar{%
212     \@minipagefalse
213     \global\@newlistfalse

```

This `\if@inlabel` check is needed in case an item starts of inside a group so that `\everypar` does not become empty outside that group.

```

214   \if@inlabel
215     \global\@inlabelfalse

```

The paragraph indent is now removed by using `\setbox...` since this makes `\noindent` a no-op here, as it should be. Thus the following comment is redundant but is left here for the sake of future historians: this next command was changed from an `hskip` to a kern to avoid a break point after the `parindent` box: the skip could cause a line-break if a very long label occurs in `raggedright` setting. If `\noindent` was used after `\item` want

to cancel the `\itemindent` skip. This case can be detected as the indentation box will be void.

```

216      {\setbox\z@\lastbox
217       \ifvoid\z@
218         \kern-\itemindent
219       \fi}%
220     \box\@labels
221     \penalty\z@
222   \fi

```

This code is intended to prevent a page break after the first line of an item that comes immediately after a section title. It may be sensible to always forbid a page break after one line of an item? As with all such settings of `\clubpenalty` it is local so will have no effect if the item starts in a group.

Only resetting `@nobreak` when it is true is now essential since now it is sometimes set locally.

```

223     \if@nobreak
224       \@nobreakfalse
225       \clubpenalty \@M
226     \else
227       \clubpenalty \@clubpenalty
228       \everypar{}%
229     \fi}%
230   \if@noitemarg
231     \@noitemargfalse
232     \if@nmbrlist
233       \refstepcounter\@listctr
234     \fi
235   \fi

```

We use `\sbox` to support color commands.

```

236   \sbox\@tempboxa{\makelabel{#1}}%
237   \global\setbox\@labels\hbox{%
238     \unhbox\@labels
239     \hskip \itemindent
240     \hskip -\labelwidth
241     \hskip -\labelsep
242     \ifdim \wd\@tempboxa >\labelwidth
243       \box\@tempboxa
244     \else
245       \hbox to\labelwidth {\unhbox\@tempboxa}%
246     \fi
247     \hskip \labelsep}%
248   \ignorespaces}

```

(End of definition for \item.)

`\makelabel`

```

249 \def\makelabel#1{%
250   \@latex@error{Lonely \string\item--perhaps a missing
251     list environment}\@ehc}

```

(End of definition for `\makelabel`.)

`\@nbitem`

```
252 \def\@nbitem{%  
253   \@tempskipa\@outerparskip  
254   \advance\@tempskipa -\parskip  
255   \addvspace\@tempskipa}
```

(End of definition for `\@nbitem`.)

`\usecounter`

```
256 \def\usecounter#1{\@nmblisttrue\def\@listctr{#1}\setcounter{#1}\z@}
```

(End of definition for `\usecounter`.)

1.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enumN` controls the numbering of the Nth level enumeration. The label is generated by the commands `\labelenumi` ... `\labelenumiv`, which should be defined by the document style. Note that `\p@enumN\theenumN` defines the output of a `\ref` command. A typical definition might be:

```
\def\theenumii{\alph{enumii}}  
\def\p@enumii{\theenumi}  
\def\labelenumii{(\theenumii)}
```

which will print the labels as ‘(a)’, ‘(b)’, ... and print a `\ref` as ‘3a’.

The item numbers are moved to the right of the label box, so they are always a distance of `\labelsep` from the item.

`\@enumdepth` holds the current enumeration nesting depth.

Itemization is controlled by four commands: `\labelitemi`, `\labelitemii`, `\labelitemiii`, and `\labelitemiv`. To cause the second-level list to be bulleted, you just define `\labelitemii` to be `•`. `\@itemspacing` and `\@itemdepth` are the analogs of `\@enumspacing` and `\@enumdepth`.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\enumerate ==  
BEGIN  
  if \@enumdepth > 3  
    then errormessage: “Too deeply nested”.  
  else \@enumdepth :=L \@enumdepth + 1  
    \@enumctr :=L eval(enum@\romannumeral\the\@enumdepth)  
    \list{\label(\@enumctr)}  
      {\usecounter{\@enumctr}  
       \makelabel{LABEL} == \hss \llap{LABEL}}  
  fi  
END
```

```
\endenumerate == \endlist
```

End of historical L^AT_EX 2.09 comments.

```

\@enumdepth
257 \newcount\@enumdepth \@enumdepth = 0
(End of definition for \@enumdepth.)

\c@enumi
\c@enumii 258 \@definecounter{enumi}
\c@enumii 259 \@definecounter{enumii}
\c@enumiv 260 \@definecounter{enumiii}
261 \@definecounter{enumiv}
(End of definition for \c@enumi and others.)

enumerate (env.)
262 \def\enumerate{%
263   \ifnum \@enumdepth >\thr@@\toodeep\else
264     \advance\@enumdepth\@ne
265     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%

266     \expandafter
267     \list
268       \csname label\@enumctr\endcsname
269       {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
270   \fi}
271 \let\endenumerate =\endlist

Historical LATEX 2.09 comments (not necessarily accurate any more):
\itemize ==
BEGIN
  if \@itemdepth > 3
  then errormessage: 'Too deeply nested'.
  else \@itemdepth :=L \@itemdepth + 1
        \@itemitem == eval(labelitem\romannumeral\the\@itemdepth)
        \list{\@nameuse{\@itemitem}}
              {\makelabel{LABEL} == \hss \llap{LABEL}}
  fi
END

\enditemize == \endlist

End of historical LATEX 2.09 comments.

\@itemdepth
272 \newcount\@itemdepth \@itemdepth = 0
(End of definition for \@itemdepth.)

itemize (env.)
273 \def\itemize{%
274   \ifnum \@itemdepth >\thr@@\toodeep\else
275     \advance\@itemdepth\@ne
276     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%

```

```

277 \expandafter
278 \list
279 \csname\@itemitem\endcsname
280 {\def\makelabel##1{\hss\llap{##1}}}%
281 \fi}

282 \let\enditemize =\endlist
283 </2ekernel>

```

File 40

ltboxes.dtx

1 L^AT_EX Box commands

`\makebox` `\makebox[ $\langle wid \rangle$ ][ $\langle pos \rangle$ ]{ $\langle obj \rangle$ }`
Puts $\langle obj \rangle$ in an `\hbox` of width $\langle wid \rangle$, positioned by $\langle pos \rangle$.
The possible $\langle pos \rangle$ are:
s stretched,
l flushleft,
r flushright,
c (default) centered.
If $\langle wid \rangle$ is missing, then $\langle pos \rangle$ is also missing and $\langle obj \rangle$ is put in an `\hbox` of its natural width.
`\makebox( $\langle x \rangle$ , $\langle y \rangle$ )[ $\langle pos \rangle$ ]{ $\langle obj \rangle$ }`
Puts $\langle obj \rangle$ in an `\hbox` of width $x * \text{\unitlength}$ and height $y * \text{\unitlength}$.
 $\langle pos \rangle$ arguments are s, l, r or c (default) for stretched, flushleft, flushright or centered, and t or b for top, bottom – or combinations like tr or rb. Default for horizontal and vertical are centered. Note that in this picture mode version of `\makebox` a [b] aligns on the *bottom* of the text as documented. If you want to align on the *baseline* use `\makebox( , ) [b]{\raisebox{0pt}[ $\langle height \rangle$ ][0pt]{xyz}}` or `\makebox( , ) [b]{\smash{xyz}}`
`\mbox` `\mbox{ $\langle obj \rangle$ }` The same as `\makebox{ $\langle obj \rangle$ }`, but is more efficient as no checking for optional arguments is done.
`\newsavebox` `\newsavebox{ $\langle cmd \rangle$ }` : If $\langle cmd \rangle$ is undefined, then defines it to be a T_EX box register.
`\savebox` `\savebox{ $\langle cmd \rangle$ }` ... : $\langle cmd \rangle$ is defined to be a T_EX box register, and the ‘...’ are any `\makebox` arguments. It is like `\makebox`, except it doesn’t produce text but saves the value in `\box  $\langle cmd \rangle$` .
`\sbox` `\sbox{ $\langle cmd \rangle$ }{ $\langle obj \rangle$ }` is an efficient abbreviation for `\savebox{ $\langle cmd \rangle$ }{ $\langle obj \rangle$ }`.
`lrbox (env.)` `\begin{lrbox}{ $\langle cmd \rangle$ }\text{\end{lrbox}}` is equivalent to `\sbox{ $\langle cmd \rangle$ }{ $\langle text \rangle$ }` except that any white space at the beginning and end of $\langle text \rangle$ is ignored.
`\framebox` `\framebox ...` : like `\makebox`, except it puts a ‘frame’ around the box. The frame is made of lines of thickness `\fboxrule`, separated by space `\fboxsep` from the text – except for `\framebox(X,Y) ...`, where the thickness of the lines is as for the picture environment, and there is no separation added.
`\fbox` `\fbox{ $\langle obj \rangle$ }` is an abbreviation for `\framebox{ $\langle obj \rangle$ }`.
`\parbox` `\parbox[ $\langle pos \rangle$ ][ $\langle height \rangle$ ][ $\langle inner-pos \rangle$ ]{ $\langle width \rangle$ }{ $\langle text \rangle$ }` : Makes a box with `\hsize  $\langle width \rangle$` , positioned by $\langle pos \rangle$ as follows: c : `\vcenter` (placed in $\$...\$$ if not in math mode) b : `\vbox` t : `\vtop` default value is c. Sets `\hsize :=  $\langle width \rangle$` and calls `\@parboxrestore`, which does the following: Restores the original definitions of:
`\par`
`\`
`\-`
`\,`
`\‘`
`\=`


```

Resets the following parameters:
\parindent      = 0pt
\parskip        = 0pt
\linewidth      = \hsize
\@totalleftmargin = 0pt
\leftskip       = 0pt
\rightskip      = 0pt
\@rightskip     = 0pt
\parfillskip    = 0pt plus 1fil
\lineskip       = \normallineskip
\baselineskip   = \normalbaselineskip
Calls \sloppy
Note: \arrayparboxrestore same as \@parboxrestore but it doesn't restore \.
minipage (env.) minipage : Similar to \parbox, except it also makes this look like a page by setting
\textwidth == \columnwidth == box width
changes footnotes by redefining:
\@mpfn == mpfootnote
\thempfn == \thempfootnote
\@footnotetext == \@mpfootnotetext
resets the following list environment parameters:
\@listdepth == \@mplistdepth
where \@mplistdepth is initialized to zero,
and executes \@minipagerestore to allow the document style to reset any other
parameters it desires. It sets @minipage true, and resets \everypar to set it false. This
switch keeps \addvspace from putting space at the top of a minipage.
Change added 24 May 89: \minipage sets @minipage globally; \endminipage resets
it false.
\rule \rule[\langle raised \rangle]{\langle width \rangle}{\langle height \rangle} : Makes a \langle width \rangle * \langle height \rangle rule, raised
\langle raised \rangle.
\underline \underline{\langle text \rangle} : Makes an underlined hbox with \langle text \rangle in it.
\raisebox \raisebox{\langle distance \rangle}[\langle height \rangle][\langle depth \rangle]{\langle box \rangle} :
Raises \langle box \rangle up by \langle distance \rangle length (down if \langle distance \rangle negative). Makes TEX think
that the new box extends \langle height \rangle above the line and \langle depth \rangle below, for a total vertical
length of \langle height \rangle + \langle depth \rangle. Default values of \langle height \rangle & \langle depth \rangle = actual height and
depth of box in new position.
1 \langle *2kernel \rangle
2 \message{boxes,}

\makebox \makebox User level command just looks for optional [ or (.
3 \langle /2kernel \rangle
4 \langle latexrelease \rangle \IncludeInRelease{2015/01/01}%
5 \langle latexrelease \rangle {\makebox}{\Make \makebox robust}%
6 \langle *2kernel | latexrelease \rangle
7 \DeclareRobustCommand\makebox{%
8 \leavevmode
9 \@ifnextchar(%)
10 \@makepicbox
11 {\@ifnextchar[\@makebox\mbox}}}%
12 \langle /2kernel | latexrelease \rangle
13 \langle latexrelease \rangle \EndIncludeInRelease

```

```

14 <latexrelease>\IncludeInRelease{0000/00/00}%
15 <latexrelease>          {\makebox}{Make \makebox robust}%
16 <latexrelease>\def\makebox{%
17 <latexrelease>  \leavevmode
18 <latexrelease>  \@ifnextchar(%)
19 <latexrelease>    \makepicbox
20 <latexrelease>    {\@ifnextchar[\@makebox\mbox}}%
21 <latexrelease>\expandafter\let\csname makebox \endcsname\@undefined
22 <latexrelease>\EndIncludeInRelease
23 <*2ekernel>

```

(End of definition for \makebox.)

\mbox The basic horizontal box command for L^AT_EX.

```
24 \DeclareRobustCommand\mbox[1]{\leavevmode\hbox{#1}}
```

(End of definition for \mbox.)

\@makebox Look for a possible second optional argument (defaults to c).

```
25 \def\@makebox[#1]{%
26   \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}
```

(End of definition for \@makebox.)

\@begin@tempboxa Helper macro for supporting \height, \width etc. Grab #1 into \@tempboxa and measure it.

```

27 \long\def\@begin@tempboxa#1#2{%
28   \begingroup
29   \setbox\@tempboxa#1\color@begingroup#2\color@endgroup}%
30   \def\width{\wd\@tempboxa}%
31   \def\height{\ht\@tempboxa}%
32   \def\depth{\dp\@tempboxa}%
33   \let\totalheight\@ovri
34   \totalheight\height
35   \advance\totalheight\depth}

```

(End of definition for \@begin@tempboxa.)

\@end@tempboxa End the group started by \@begin@tempboxa, so that the scope of \height only includes the ‘length’ argument to the user-command.

```
36 \let\@end@tempboxa\endgroup
```

(End of definition for \@end@tempboxa.)

\bm@c Set up spacing.

```

\bm@l 37 \def\bm@c{\hss\unhbox\@tempboxa\hss}
\bm@r 38 \def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\bm@s 39 \def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\bm@t 40 \def\bm@s{\unhbox\@tempboxa}
\bm@b

```

(End of definition for \bm@c and others.)

`\@imakebox` Internal form of `\makebox`.

```

41 </2ekernel>
42 <latexrelease>\IncludeInRelease{2023/11/01}%
43 <latexrelease>          {\@imakebox}{Unknown alignment warning}%
44 <*2ekernel | latexrelease>
45 \long\def\@imakebox[#1][#2]#3{%
46   \@begin@tempboxa\hbox{#3}%
47   \setlength\@tempdima{#1}%      support calc
48   \hb@xt@\@tempdima{%
49     \expandafter\ifx\csname bm@#2\endcsname\relax
50       \bm@c
51       \@latex@warning{Unexpected alignment #2}%
52     \else
53       \csname bm@#2\endcsname
54     \fi}%
55   \@end@tempboxa}
56 </2ekernel | latexrelease>
57 <latexrelease>\EndIncludeInRelease
58 <latexrelease>\IncludeInRelease{0000/00/00}%
59 <latexrelease>          {\@imakebox}{Unknown alignment warning}%
60 <latexrelease>\long\def\@imakebox[#1][#2]#3{%
61 <latexrelease>   \@begin@tempboxa\hbox{#3}%
62 <latexrelease>   \setlength\@tempdima{#1}%      support calc
63 <latexrelease>   \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
64 <latexrelease>   \@end@tempboxa}
65 <latexrelease>\EndIncludeInRelease
66 <*2ekernel>

```

(End of definition for \@imakebox.)

`\@makepicbox` Picture mode form of `\makebox`.

```

67 \def\@makepicbox(#1,#2){%
68   \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2) []}}

```

(End of definition for \@makepicbox.)

`\@imakepicbox` picture mode version

```

69 </2ekernel>
70 <*2ekernel | latexrelease>
71 <latexrelease>\IncludeInRelease{2020/10/01}%
72 <latexrelease>          {\@imakepicbox}{default units}%
73 \long\def\@imakepicbox(#1,#2)[#3]#4{%
74   \@defaultunitsset\@tempdimc{#2}\unitlength
75   \vbox to\@tempdimc
76   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
77    \let\mb@t\vss
78    \@tfor\reserved@a :=#3\do{%
79      \if s\reserved@a
80        \let\mb@l\relax\let\mb@r\relax
81      \else
82        \expandafter\let\csname mb@\reserved@a\endcsname\relax
83      \fi}%
84   \mb@t
85   \@defaultunitsset\@tempdimc{#1}\unitlength

```

```

86 \hb@xt@{\@tempdimc{\mb@l #4\mb@r}}%
87 \mb@b

```

This kern ensures that a `b` option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the L^AT_EX Book. The kern is removed in compatibility mode.

```

88 \kern\z@}}
89 </2ekernel | latexrelease>

90 <latexrelease>\EndIncludeInRelease
91 <latexrelease>\IncludeInRelease{0000/00/00}%
92 <latexrelease> \imakepicbox{default units}%
93 <latexrelease>\long\def\imakepicbox(#1,#2)[#3]#4{%
94 <latexrelease> \vbox to#2\unitlength
95 <latexrelease> {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
96 <latexrelease> \let\mb@t\vss
97 <latexrelease> \@tfor\reserved@a :=#3\do{%
98 <latexrelease> \if s\reserved@a
99 <latexrelease> \let\mb@l\relax\let\mb@r\relax
100 <latexrelease> \else
101 <latexrelease> \expandafter\let\csname mb@\reserved@a\endcsname\relax
102 <latexrelease> \fi}%
103 <latexrelease> \mb@t
104 <latexrelease> \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
105 <latexrelease> \mb@b
106 <latexrelease> \kern\z@}}
107 <latexrelease>\EndIncludeInRelease
108 <*2ekernel>

```

(End of definition for \imakepicbox.)

`\set@color` This macro is initially a no-op, but the color package will redefine it to insert a `\special`.

```

109 \let\set@color\relax

```

(End of definition for \set@color.)

`\color@begingroup` In the past these macros were initially no-ops, and the color package redefined them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`, `\color@setgroup` them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`, `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and `<set to main document color>` respectively.

`\normalcolor` Nowadays we always set the group already in the kernel as this makes the coding simpler.

`\color@hbox`

`\color@vbox`

`\color@endbox`

```

110 </2ekernel>
111 <*2ekernel | latexrelease>
112 <latexrelease>\IncludeInRelease{2021/06/01}%
113 <latexrelease> \color@begingroup{color group settings}%
114 \let\color@begingroup\begingroup
115 \def\color@setgroup{\color@begingroup} % changed further in color package
116 \let\normalcolor\relax % remains untouched; only changed in a color package
117 \def\color@hbox{\hbox\bgroup\color@begingroup}
118 \def\color@vbox{\vbox\bgroup\color@begingroup}
119 \def\color@endbox{\color@endgroup\egroup}
120 </2ekernel | latexrelease>
121 <latexrelease>\EndIncludeInRelease

```

```

122 <latexrelease>\IncludeInRelease{0000/00/00}%
123 <latexrelease>                {\color@begingroup}{color group settings}%
124 <latexrelease>
125 <latexrelease>\let\color@begingroup\relax
126 <latexrelease>\let\color@setgroup\relax
127 <latexrelease>\let\normalcolor\relax
128 <latexrelease>\let\color@hbox\relax
129 <latexrelease>\let\color@vbox\relax
130 <latexrelease>\let\color@endbox\relax
131 <latexrelease>
132 <latexrelease>\EndIncludeInRelease
133 <*2ekernel>

```

(End of definition for \color@begingroup and others.)

\color@endgroup This macro is separated out because it received an update in 2024, so requires its own rollback.

```

134 </2ekernel>
135 <*2ekernel | latexrelease>
136 <latexrelease>\IncludeInRelease{2026/06/01}%
137 <latexrelease>                {\color@endgroup}{color group settings}%

```

Beside **\endgraf** for handling vertical boxes we also reset **\if@endpe** as we are leaving the context but only if this isn't an **\hbox** we are closing.

```

138 \def\color@endgroup{\endgraf\ifvmode\@endpefalse\fi\endgroup}
139 <latexrelease>
140 </2ekernel | latexrelease>
141 <latexrelease>\EndIncludeInRelease
142 <latexrelease>\IncludeInRelease{2024/11/01}%
143 <latexrelease>                {\color@endgroup}{color group settings}%
144 <latexrelease>\def\color@endgroup{\endgraf\@endpefalse\endgroup}
145 <latexrelease>
146 <latexrelease>\EndIncludeInRelease
147 <latexrelease>\IncludeInRelease{2021/06/01}%
148 <latexrelease>                {\color@endgroup}{color group settings}%
149 <latexrelease>\def\color@endgroup{\endgraf\endgroup}
150 <latexrelease>
151 <latexrelease>\EndIncludeInRelease
152 <latexrelease>\IncludeInRelease{0000/00/00}%
153 <latexrelease>                {\color@endgroup}{color group settings}%
154 <latexrelease>
155 <latexrelease>\let\color@endgroup\relax
156 <latexrelease>
157 <latexrelease>\EndIncludeInRelease
158 <*2ekernel>

```

(End of definition for \color@endgroup.)

\newsavebox Allocate a new 'savebox'.

```

159 \def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}

```

(End of definition for \newsavebox.)

`\savebox` Save #1 in a box register.

```
160 </2ekernel>
161 <latexrelease>\IncludeInRelease{2015/01/01}%
162 <latexrelease>          {\savebox}{Make \savebox robust}%
163 <*2ekernel | latexrelease>
164 \DeclareRobustCommand\savebox[1]{%
165   \ifnextchar(%)
166     {\@savepicbox#1}{\ifnextchar[{\@savebox#1}{\sbox#1}}}%
167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{0000/00/00}%
170 <latexrelease>          {\savebox}{Make \savebox robust}%
171 <latexrelease>\def\savebox#1{%
172 <latexrelease>  \ifnextchar(%)
173 <latexrelease>    {\@savepicbox#1}{\ifnextchar[{\@savebox#1}{\sbox#1}}}%
174 <latexrelease>\expandafter\let\csname savebox \endcsname\undefined
175 <latexrelease>\EndIncludeInRelease
176 <*2ekernel>
```

(End of definition for \savebox.)

`\sbox` Save #1 in a box register.

```
177 \DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
178   \color@setgroup#2\color@endgroup}}
```

(End of definition for \sbox.)

`\@savebox` Look for second optional argument.

```
179 \def\@savebox#1[#2]{%
180   \ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}}
```

(End of definition for \@savebox.)

`\@isavebox`

```
181 \long\def\@isavebox#1[#2][#3]#4{%
182   \sbox#1{\@imakebox[#2][#3]{#4}}}
```

(End of definition for \@isavebox.)

`\@savepicbox` Picture mode version of `\savebox`.

```
183 \def\@savepicbox#1(#2,#3){%
184   \ifnextchar[%]
185     {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3)[]}}
```

(End of definition for \@savepicbox.)

`\@isavepicbox` Picture mode version of `\savebox`.

```
186 \long\def\@isavepicbox#1(#2,#3)[#4]#5{%
187   \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}}
```

(End of definition for \@isavepicbox.)

`\lrbox` `lrbox`: the new environment form of `\sbox`. Use `\aftergroup` tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the `lrbox` environment.

```

188 \def\lrbox#1{%
189   \edef\reserved@a{%
190     \endgroup
191     \setbox#1\hbox{%
192       \begingroup\aftergroup}%
193       \def\noexpand\@currenvir{\@currenvir}%
194       \def\noexpand\@currenvline{\on@line}}%
195   \reserved@a
196   \@endpefalse
197   \color@setgroup
198   \ignorespaces}

```

(End of definition for `\lrbox`.)

`\endlrbox` End the `lrbox` environment.

```

199 \def\endlrbox{\unskip\color@endgroup}

```

(End of definition for `\endlrbox`.)

`\usebox` unchanged

```

200 \DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}

```

(End of definition for `\usebox`.)

`\frame` The following definition of `\frame` was written by Pavel Curtis (Extra space removed 14 Jan 88) RmS 92/08/24: Replaced occurrence of `\@halfwidth` by `\@wholewidth`

```

201 \DeclareRobustCommand\frame[1]{%
202   \leavevmode
203   \hbox{%
204     \hskip-\@wholewidth
205     \vbox{%
206       \vskip-\@wholewidth
207       \hrule \@height\@wholewidth
208       \hbox{%
209         \vrule\@width\@wholewidth
210         #1%
211         \vrule\@width\@wholewidth}%
212       \hrule \@height\@wholewidth
213       \vskip-\@wholewidth}%
214     \hskip-\@wholewidth}}

```

(End of definition for `\frame`.)

`\fboxrule` user level parameters,

`\fboxsep`

```

215 \newdimen\fboxrule
216 \newdimen\fboxsep

```

(End of definition for `\fboxrule` and `\fboxsep`.)

`\fbox` Abbreviated framed box command.

```

217 \DeclareRobustCommand\fbox[1]{%
218   \leavevmode
219   \setbox\@tempboxa\hbox{%
220     \color@bgroup
221     \kern\fboxsep{#1}\kern\fboxsep
222     \color@endgroup}%
223   \@frameb@x\relax}

```

(End of definition for \fbox.)

`\framebox` Framed version of `\makebox`.

```

224 </2ekernel>
225 <latexrelease>\IncludeInRelease{2015/01/01}%
226 <latexrelease>           {\framebox}{Make \framebox robust}%
227 <*2ekernel | latexrelease>
228 \DeclareRobustCommand\framebox{%
229   \@ifnextchar(%)
230     \framepicbox{\@ifnextchar[\@framebox\fbox}}%
231 </2ekernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease>           {\framebox}{Make \framebox robust}%
235 <latexrelease>\def\framebox{%
236 <latexrelease>   \@ifnextchar(%)
237 <latexrelease>     \framepicbox{\@ifnextchar[\@framebox\fbox}}%
238 <latexrelease>\expandafter\let\csname framebox \endcsname\undefined
239 <latexrelease>\EndIncludeInRelease
240 <*2ekernel>

```

(End of definition for \framebox.)

`\@framebox` Deal with optional arguments.

```

241 \def\@framebox[#1]{%
242   \@ifnextchar[%]
243     {\@ifframebox[#1]}%
244     {\@ifframebox[#1][c]}

```

(End of definition for \@framebox.)

`\@ifframebox` The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```

245 </2ekernel>
246 <latexrelease>\IncludeInRelease{2023/11/01}%
247 <latexrelease>           {\@ifframebox}{Unknown alignment warning}%
248 <*2ekernel | latexrelease>
249 \long\def\@ifframebox[#1][#2]#3{%
250   \leavevmode
251   \@begin@tempboxa\hbox{#3}%
252   \setlength\@tempdima{#1}%
253   \setbox\@tempboxa\hb@xt@\@tempdima
254     {\kern\fboxsep
255     \expandafter\ifx\csname bm@#2\endcsname\relax

```



```

256         \bm@c
257         \latex@warning{Unexpected alignment #2}%
258     \else
259         \csname bm@#2\endcsname
260     \fi
261     \kern\fboxsep}%
262     \@frameb@x{\kern-\fboxrule}%
263     \@end@tempboxa}
264 \
```

(End of definition for \@iframbox.)

\@frameb@x Common part of \framebox and \fbox. #1 is a negative kern in the \framebox case so that the vertical rules do not add to the width of the box.

```

278 \def\@frameb@x#1{%
279     \@tempdima\fboxrule
280     \advance\@tempdima\fboxsep
281     \advance\@tempdima\dp\@tempboxa
282     \hbox{%
283         \lower\@tempdima\hbox{%
284             \vbox{%
285                 \hrule\@height\fboxrule
286                 \hbox{%
287                     \vrule\@width\fboxrule
288                     #1%
289                     \vbox{%
290                         \vskip\fboxsep
291                         \box\@tempboxa
292                         \vskip\fboxsep}%
293                     #1%
294                     \vrule\@width\fboxrule}%
295                 \hrule\@height\fboxrule}%
296                 }%
297             }%
298 }
```

(End of definition for \@frameb@x.)

\@framepicbox Picture mode version.

```

299 \def\@framepicbox(#1,#2){%
300     \ifnextchar[{\@framepicbox(#1,#2)}{\@framepicbox(#1,#2) []}}
```

(End of definition for \@framepicbox.)

\@ifframepicbox Picture mode version.

```
301 \long\def\@ifframepicbox(#1,#2)[#3]#4{%  
302   \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

(End of definition for \@ifframepicbox.)

\parbox The main vertical-box command for L^AT_EX.

```
303 </2ekernel>  
304 <latexrelease>\IncludeInRelease{2015/01/01}%  
305 <latexrelease>          {\parbox}{Make \parbox robust}%  
306 <*2ekernel | latexrelease>  
307 \DeclareRobustCommand\parbox{%  
308   \@ifnextchar[%]  
309     \@iparbox  
310     {\@iiiparbox c\relax[s]}}%  
311 </2ekernel | latexrelease>  
312 <latexrelease>\EndIncludeInRelease  
313 <latexrelease>\IncludeInRelease{0000/00/00}%  
314 <latexrelease>          {\parbox}{Make \parbox robust}%  
315 <latexrelease>\def\parbox{%  
316 <latexrelease>  \@ifnextchar[%]  
317 <latexrelease>    \@iparbox  
318 <latexrelease>    {\@iiiparbox c\relax[s]}}%  
319 <latexrelease>\expandafter\let\csname parbox \endcsname\undefined  
320 <latexrelease>\EndIncludeInRelease  
321 <*2ekernel>
```

(End of definition for \parbox.)

\@iparbox Optional argument handling.

```
322 \def\@iparbox[#1]{%  
323   \@ifnextchar[%]  
324     {\@iparbox{#1}}%  
325     {\@iiiparbox{#1}\relax[s]}}
```

(End of definition for \@iparbox.)

\@iiiparbox Optional argument handling.

```
326 \def\@iiiparbox#1[#2]{%  
327   \@ifnextchar[%]  
328     {\@iiiparbox{#1}{#2}}%  
329     {\@iiiparbox{#1}{#2}[#1]}}
```

(End of definition for \@iiiparbox.)

\vcenter@text We have this idea in the L₃ layer, but for use in tabulars, we need close-to primitive syntax: that means some \afterassignment and \aftergroup fun (remember \afterassignment for boxes inserts immediately after the start of the box). To keep the calculation here as close as possible to the one used by \vcenter, we implement the algorithm from tex.web:

\vcenter@text@auxi
\vcenter@text@auxii
\vcenter@text@axis

```

delta:=height(v)+depth(v);
height(v):=axis_height(cur_size)+half(delta);
depth(v):=delta-height(v);

```

where

```

@p function half(@!x:integer):integer;
begin if odd(x) then half:=(x+1) div 2
else half:=x @!div 2;
end;

```

As the total height is needed a couple of times, this is worked out in a scratch dimen, before doing everything else using expressions. This way we get pretty much exactly the same as `tex.web`.

```

330 </2ekernel>
331 <latexrelease>\IncludeInRelease{2026/06/01}%
332 <latexrelease>          {\vcenter@text}{Use real text vcenter}%
333 <*2ekernel | latexrelease>
334 \protected\def\vcenter@text{%
335   \check@mathfonts
336   \afterassignment\vcenter@text@auxi
337   \setbox0\vbox}
338 \protected\def\vcenter@text@auxi{%
339   \aftergroup\vcenter@text@auxii
340 }
341 \protected\def\vcenter@text@auxii{%
342   \begingroup
343   \dimen0\dimexpr\ht0+\dp0\relax
344   \ht0\dimexpr\dimexpr\ifodd\dimen0 1sp + \fi\dimen0\relax/2
345   +\vcenter@text@axis\relax
346   \dp0\dimexpr\dimen0-\ht0\relax
347   \box0 %
348   \endgroup
349 }
350 \ifdefined\directlua
351   \newcommand*\vcenter@text@axis{\Umathaxis\textstyle}
352 \else
353   \newcommand*\vcenter@text@axis{\fontdimen22 \textfont2 }
354 \fi
355 </2ekernel | latexrelease>
356 <latexrelease>\EndIncludeInRelease
357 <latexrelease>\IncludeInRelease{0000/00/00}%
358 <latexrelease>          {\vcenter@text}{Use real text vcenter}%
359 <latexrelease>\let\vcenter@text\undefined
360 <latexrelease>\let\vcenter@text@auxi\undefined
361 <latexrelease>\let\vcenter@text@auxii\undefined
362 <latexrelease>\let\vcenter@text@axis\undefined
363 <latexrelease>\EndIncludeInRelease
364 <*2ekernel>

```

(End of definition for \vcenter@text and others.)

`\@iiiparbox` The internal version of `\parbox`.

`\@parboxto` 365 `\let\@parboxto\@empty`

```

366 </2ekernel>
367 <latexrelease>\IncludeInRelease{2026/06/01}%
368 <latexrelease>          {\@iiiparbox}{Use real text vcenter}%
369 <*2ekernel | latexrelease>
370 \long\def\@iiiparbox#1#2[#3]#4#5{%
371   \leavevmode
372   \setlength\@tempdima{#4}%
373   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
374   \ifx\relax#2\else
375     \setlength\@tempdimb{#2}%
376     \edef\@parboxto{to\the\@tempdimb}%
377     \fi
378     \if#1b\vbox
379     \else\if #1t\vtop
380     \else\vcenter@text
381     \fi\fi
382     \@parboxto{\let\hss\vss\let\unhbox\unvbox
383     \expandafter\ifx\csname bm@#3\endcsname\relax
384       \bm@c
385       \@latex@warning{Unexpected alignment #3}%
386     \else
387       \csname bm@#3\endcsname
388     \fi}%
389   \@end@tempboxa}
390 </2ekernel | latexrelease>
391 <latexrelease>\EndIncludeInRelease
392 <latexrelease>\IncludeInRelease{2023/11/01}%
393 <latexrelease>          {\@iiiparbox}{Use real text vcenter}%
394 <latexrelease>\long\def\@iiiparbox#1#2[#3]#4#5{%
395 <latexrelease>   \leavevmode
396 <latexrelease>   \@pboxswfalse
397 <latexrelease>   \setlength\@tempdima{#4}%
398 <latexrelease>   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
399 <latexrelease>   \ifx\relax#2\else
400 <latexrelease>     \setlength\@tempdimb{#2}%
401 <latexrelease>     \edef\@parboxto{to\the\@tempdimb}%
402 <latexrelease>     \fi
403 <latexrelease>     \if#1b\vbox
404 <latexrelease>     \else\if #1t\vtop
405 <latexrelease>     \else\ifmmode\vcenter
406 <latexrelease>     \else\@pboxswtrue $\vcenter
407 <latexrelease>     \fi\fi\fi
408 <latexrelease>     \@parboxto{\let\hss\vss\let\unhbox\unvbox
409 <latexrelease>     \expandafter\ifx\csname bm@#3\endcsname\relax
410 <latexrelease>       \bm@c
411 <latexrelease>       \@latex@warning{Unexpected alignment #3}%
412 <latexrelease>     \else
413 <latexrelease>       \csname bm@#3\endcsname
414 <latexrelease>     \fi}%
415 <latexrelease>     \if@pboxsw \m@th$\fi
416 <latexrelease>   \@end@tempboxa}
417 <latexrelease>\EndIncludeInRelease
418 <latexrelease>\IncludeInRelease{0000/00/00}%
419 <latexrelease>          {\@iiiparbox}{Use real text vcenter}%

```

```

420 <latexrelease>\long\def\@iiiparbox#1#2[#3]#4#5{%
421 <latexrelease> \leavevmode
422 <latexrelease> \pboxswfalse
423 <latexrelease> \setlength\@tempdima{#4}%
424 <latexrelease> \begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%
425 <latexrelease> \ifx\relax#2\else
426 <latexrelease> \setlength\@tempdimb{#2}%
427 <latexrelease> \edef\@parboxto{to\the\@tempdimb}%
428 <latexrelease> \fi
429 <latexrelease> \if#1b\vbox
430 <latexrelease> \else\if #1t\vtop
431 <latexrelease> \else\ifmmode\vcenter
432 <latexrelease> \else\pboxswtrue $\vcenter
433 <latexrelease> \fi\fi\fi
434 <latexrelease> \@parboxto{\let\hss\vss\let\unhbox\unvbox
435 <latexrelease> \csname bm@#3\endcsname}%
436 <latexrelease> \if@pboxsw \m@th$\fi
437 <latexrelease> \end@tempboxa}
438 <latexrelease>\EndIncludeInRelease
439 <*2ekernel>

```

(End of definition for \@iiiparbox and \@parboxto.)

\@arrayparboxrestore Restore various paragraph parameters.

The rationale for allowing two normally global flags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within boxes or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in \setnobreak; otherwise this command will be redundant.

```

440 </2ekernel>
441 <latexrelease>\IncludeInRelease{2017-04-15}%
442 <latexrelease> \if\normallineskiplimit
443 <latexrelease> \reset \lineskiplimit}%
444 <*2ekernel | latexrelease>
445 \def\@arrayparboxrestore{%
446 \let\if@nobreak\iffalse
447 \let\if@noskipsec\iffalse
448 \let\par\@par
449 \let\-\@dischph

```

Redefined accents to allow changes in font encoding

```

450 \let\'@acci\let\'@accii\let\=@acciii
451 \parindent\z@ \parskip\z@skip
452 \everypar{}%
453 \linewidth\hsize
454 \@totalleftmargin\z@
455 \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
456 \parfillskip\@flushglue
457 \lineskip\normallineskip
458 \lineskiplimit\normallineskiplimit

```

```

459 \baselineskip\normalbaselineskip
460 \sloppy}
461 </2ekernel|latexrelease>
462 <latexrelease>\EndIncludeInRelease
463 <latexrelease>\IncludeInRelease{0000-00-00}%
464 <latexrelease> \let\if@nobreak\iffalse
465 <latexrelease> \let\if@noskipsec\iffalse
466 <latexrelease>\def\@arrayparboxrestore{%
467 <latexrelease> \let\par\@@par
468 <latexrelease> \let\-\@dischph
469 <latexrelease> \let\'@accii\let\'@accii\let\=\@acciii
470 <latexrelease> \parindent\z@ \parskip\z@skip
471 <latexrelease> \everypar{}%
472 <latexrelease> \linewidth\hsize
473 <latexrelease> \totalleftmargin\z@
474 <latexrelease> \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
475 <latexrelease> \parfillskip\@flushglue \lineskip\normalbaselineskip
476 <latexrelease> \baselineskip\normalbaselineskip
477 <latexrelease> \sloppy}
478 <latexrelease>\EndIncludeInRelease
479 <*2ekernel>

```

(End of definition for \@arrayparboxrestore.)

\@parboxrestore Restore various paragraph parameters, and also \.

```

482 \def\@parboxrestore{\@arrayparboxrestore\let\\ \@normalcr}

```

(End of definition for \@parboxrestore.)

\if@minipage Switch that is true at the start of a minipage.

```

483 \def\@minipagefalse{\global\let\if@minipage\iffalse}
484 \def\@minipagetrue {\global\let\if@minipage\iftrue}
485 \@minipagefalse

```

(End of definition for \if@minipage.)

\if@in@minipage@env

```

486 \newif\if@in@minipage@env

```

(End of definition for \if@in@minipage@env.)

\minipage Essentially an environment form of \parbox.

```

487 \def\minipage{%
488 \ifnextchar[%]
489 \@iiminipage
490 {\@iiminipage c\relax[s]}}

```

(End of definition for \minipage.)

\@iminipage Optional argument handling.

```

491 \def\@iminipage[#1]{%
492 \@ifnextchar[%]
493 {\@iiminipage{#1}}%
494 {\@iiminipage{#1}\relax[s]}}

```

(End of definition for \@iminipage.)

\@iiminipage Optional argument handling.

```
495 \def\@iiminipage#1[#2]{%  
496   \@ifnextchar[%  
497     {\@iiminipage{#1}{#2}}%  
498     {\@iiminipage{#1}{#2}[#1]}}
```

(End of definition for \@iiminipage.)

\@iiiminipage Internal form of minipage.

```
499 \def\@iiiminipage#1#2[#3]#4{%  
500   \leavevmode  
501   \@pboxswfalse  
502   \setlength\@tempdima{#4}%  
503   \def\@mpargs{{#1}{#2}[#3]{#4}}%  
504   \setbox\@tempboxa\vbox\bgroup  
505     \color@begingroup  
506     \hsize\@tempdima  
507     \textwidth\hsize \columnwidth\hsize
```

We check for nested minipages inside the box so that there is always a group resetting the switch even if the code does not use `\begin` to start the minipage.

```
508   \if@in@minipage@env
```

We only issue a warning if the outer minipage contained footnotes because that is the problematical case.

```
509     \ifvoid\@mpfootins\else  
510       \@latex@warning{Nested minipage:  
511         footnotes may be misplaced}%  
512     \fi  
513   \else  
514     \@in@minipage@envtrue  
515   \fi  
  
516   \@parboxrestore  
517   \def\@mpfn{\mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@  
518   \let\@footnotetext\@mpfootnotetext  
519   \let\@listdepth\@mplistdepth \@mplistdepth\z@  
520   \@minipagerestore  
521   \@setminipage}
```

(End of definition for \@iiiminipage.)

\@minipagerestore Hook so that other styles can reset other commands in a minipage.

```
522   \let\@minipagerestore=\relax
```

(End of definition for \@minipagerestore.)

\endminipage

```
523 \def\endminipage{%  
524   \par  
525   \unskip  
526   \ifvoid\@mpfootins\else  
527     \vskip\skip\@mpfootins  
528     \normalcolor
```

```

529     \footnoterule
530     \unvbox\@mpfootins
531   \fi
532   \@minipagefalse    %% added 24 May 89
533   \color@endgroup
534   \egroup
535   \expandafter\@iiparbox\@mpargs{\unvbox\@tempboxa}}

```

(End of definition for \endminipage.)

\@mplistdepth Versions of \@listdepth and \footins local to minipage.

```

\@mpfootins 536 \newcount\@mplistdepth
537 \newinsert\@mpfootins

```

(End of definition for \@mplistdepth and \@mpfootins.)

\@mpfootnotetext Minipage version of \@footnotetext.

Final \strut added 27 Mar 89, on suggestion by Don Hosek

```

538 \</2ekernel>
539 \< *2ekernel | latexrelease>
540 \< latexrelease> \IncludeInRelease{2021/11/15}%
541 \< latexrelease> { \@mpfootnotetext } {footnotetext tagging}%
542 \long\def\@mpfootnotetext#1{%
543   \global\setbox\@mpfootins\vbox{%
544     \unvbox\@mpfootins
545     \reset@font\footnotesize
546     \hsize\columnwidth
547     \@parboxrestore
548     \def\@currentcounter{mpfootnote}%
549     \protected@edef\@currentlabel
550       {\csname p@mpfootnote\endcsname \@thefnmark}%
551     \color@begingroup
552     \makefnintext{%
553       \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
554     \par
555     \color@endgroup}}
556 \< /2ekernel | latexrelease>
557 \< latexrelease> \EndIncludeInRelease

558 \< latexrelease> \IncludeInRelease{2021/06/01}%
559 \< latexrelease> { \@mpfootnotetext } {footnotetext tagging}%
560 \< latexrelease> \long\def\@mpfootnotetext#1{%
561 \< latexrelease>   \global\setbox\@mpfootins\vbox{%
562 \< latexrelease>     \unvbox\@mpfootins
563 \< latexrelease>     \reset@font\footnotesize
564 \< latexrelease>     \hsize\columnwidth
565 \< latexrelease>     \@parboxrestore
566 \< latexrelease>     \protected@edef\@currentlabel
567 \< latexrelease>       {\csname p@mpfootnote\endcsname \@thefnmark}%
568 \< latexrelease>     \color@begingroup
569 \< latexrelease>     \makefnintext{%
570 \< latexrelease>       \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
571 \< latexrelease>     \par
572 \< latexrelease>     \color@endgroup}}
573 \< latexrelease> \EndIncludeInRelease

```



```

574 <latexrelease>\IncludeInRelease{0000/00/00}%
575 <latexrelease>          {\mpfootnotetext}{footnotetext tagging}%
576 <latexrelease>
577 <latexrelease>\long\def\@mpfootnotetext#1{%
578 <latexrelease>  \global\setbox\@mpfootins\vbox{%
579 <latexrelease>    \unvbox\@mpfootins
580 <latexrelease>    \reset@font\footnotesize
581 <latexrelease>    \hsize\columnwidth
582 <latexrelease>    \@parboxrestore
583 <latexrelease>    \protected@edef\@currentlabel
584 <latexrelease>      {\csname p@mpfootnote\endcsname\@thefnmark}%
585 <latexrelease>    \color@begingroup
586 <latexrelease>      \@makefntext{%
587 <latexrelease>        \rule{z@footnotesep\ignorespaces#1\@finalstrut\strutbox}%
588 <latexrelease>      \color@endgroup}}
589 <latexrelease>
590 <latexrelease>\EndIncludeInRelease
591 <*2ekernel>

```

(End of definition for \mpfootnotetext.)

```
592 \newif\if@pboxsw
```

\rule Draw a rule of the specified size.

```

593 </2ekernel>
594 <latexrelease>\IncludeInRelease{2015/01/01}%
595 <latexrelease>          {\rule}{Make \rule robust}%
596 <*2ekernel | latexrelease>
597 \DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
598 </2ekernel | latexrelease>
599 <latexrelease>\EndIncludeInRelease
600 <latexrelease>\IncludeInRelease{0000/00/00}%
601 <latexrelease>          {\rule}{Make \rule robust}%
602 <latexrelease>\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
603 <latexrelease>\expandafter\let\csname rule \endcsname\@undefined
604 <latexrelease>\EndIncludeInRelease
605 <*2ekernel>

```

(End of definition for \rule.)

\@rule Internal form of \rule.

```

606 \def\@rule[#1]#2#3{%
607 \leavevmode
608 \hbox{%
609 \setlength\@tempdima{#1}%
610 \setlength\@tempdimb{#2}%
611 \setlength\@tempdimc{#3}%
612 \advance\@tempdimc\@tempdima
613 \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}

```

(End of definition for \@rule.)

\@@underline Saved primitive \underline.

```
614 \let\@@underline\underline
```

(End of definition for \@@underline.)

$\underline{\hspace{1cm}}$ $\underline{\hspace{1cm}}\text{a}$ $\underline{\hspace{1cm}}\text{a}\text{v}$ $\underline{\hspace{1cm}}\text{a}\text{t}$ $\underline{\hspace{1cm}}\text{a}\text{t}$	<p>As with other areas where we have removed math mode usage from text mode material, we get as close as possible to primitive behaviors. As in other places, we have to do a low-level box size change rather than add a kern inside the box for the vgap. To preserve glue setting, we need to add an additional box level compared to the $\text{\TeX}$ primitive: using $\text{\unhcopy}$ fails if the content contains any glue components.</p>
--	--

```

615 </2ekernel>
616 <latexrelease>\IncludeInRelease{2026-11-01}%
617 <latexrelease>          {\underline}{Avoid 'fake math'}%
618 <*2ekernel | latexrelease>
619 \DeclareRobustCommand\underline[1]{%
620   \relax
621   \ifmmode
622     \expandafter\@@underline
623   \else
624     \expandafter\underline@text
625   \fi
626   {#1}%
627 }
628 \protected\def\underline@text#1{%
629   \leavevmode
630   \check@mathfonts
631   \begingroup
632     \setbox0=\hbox{#1}%
633     \setbox0=\vtop{%
634       \hsize\wd0 %
635       \copy0 %
636       \kern\dimexpr\underline@text@vgap\relax
637       \hrule height\dimexpr\underline@text@thickness\relax
638       depth 0pt\relax
639     }%
640     \dp0\dimexpr\dp0+\underline@text@kern\relax
641     \box0 %
642   \endgroup
643 }
644 \ifdefined\directlua
645   \newcommand*\underline@text@vgap{\Umathunderbarvgap\textstyle}
646   \newcommand*\underline@text@thickness{\Umathunderbarrule\textstyle}
647   \newcommand*\underline@text@kern{\Umathunderbarkern\textstyle}
648 \else
649   \newcommand*\underline@text@vgap{3\fontdimen8\textfont3 }
650   \newcommand*\underline@text@thickness{\fontdimen8\textfont3 }
651   \newcommand*\underline@text@kern{\fontdimen8\textfont3 }
652 \fi
653 </2ekernel | latexrelease>
654 <latexrelease>\EndIncludeInRelease
655 <latexrelease>\IncludeInRelease{0000/00/00}%
656 <latexrelease>          {\underline}{Avoid 'fake math'}%
657 <latexrelease>\DeclareRobustCommand\underline[1]{%
658 <latexrelease>  \relax
659 <latexrelease>  \ifmmode\@@underline{#1}%
660 <latexrelease>  \else $\@@underline{\hbox{#1}}\m@th$\relax\fi
661 <latexrelease>\let\underline@text\@undefined
662 <latexrelease>\let\underline@text@vgap\@undefined

```

```

663 <latexrelease>\let\underline@text@thickness\@undefined
664 <latexrelease>\let\underline@text@kern\@undefined
665 <latexrelease>\EndIncludeInRelease
666 <*2ekernel>

```

(End of definition for \underline and others.)

\raisebox Raise a box, and change its vertical dimensions.

```

667 </2ekernel>
668 <latexrelease>\IncludeInRelease{2015/01/01}%
669 <latexrelease>          {\raisebox}{Make \raisebox robust}%
670 <*2ekernel | latexrelease>
671 \DeclareRobustCommand\raisebox[1]{%
672   \leavevmode
673   \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
674 </2ekernel | latexrelease>
675 <latexrelease>\EndIncludeInRelease
676 <latexrelease>\IncludeInRelease{0000/00/00}%
677 <latexrelease>          {\raisebox}{Make \raisebox robust}%
678 <latexrelease>\def\raisebox#1{%
679 <latexrelease>  \leavevmode
680 <latexrelease>  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
681 <latexrelease>\expandafter\let\csname raisebox \endcsname\@undefined
682 <latexrelease>\EndIncludeInRelease
683 <*2ekernel>

```

(End of definition for \raisebox.)

\@rsbox Optional argument handling.

```

684 \def\@rsbox#1[#2]{%
685   \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}

```

(End of definition for \@rsbox.)

\@argrsbox ...

(End of definition for \@argrsbox.)

\@irsbox Internal version of \raisebox (less than two optional args).

```

686 \long\def\@irsbox#1[#2]#3{%
687   \@begin@tempboxa\hbox{#3}%
688   \setlength\@tempdima{#1}%
689   \ifx\#2\\\else\setlength\@tempdimb{#2}\fi
690   \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
691   \ifx\#2\\\else\ht\@tempboxa\@tempdimb\fi
692   \box\@tempboxa
693   \@end@tempboxa}

```

(End of definition for \@irsbox.)

\@iirsbox Internal version of \raisebox (two optional args).

```

694 \long\def\@iirsbox#1[#2][#3]#4{%
695   \@begin@tempboxa\hbox{#4}%
696   \setlength\@tempdima{#1}%
697   \setlength\@tempdimb{#2}%

```

```

698 \setlength\dimen@{#3}%
699 \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
700 \ht\@tempboxa\@tempdimb
701 \dp\@tempboxa\dimen@
702 \box\@tempboxa
703 \@end@tempboxa}

```

(End of definition for \@iirsbox.)

\@finalstrut This macro adds a special strut the *depth* of the box given as #1, and height and width Opt. It is used for ensuring that the last line of a paragraph has the correct depth in ‘p’ columns of tables and in footnotes. In vertical mode nothing is done, as adding the strut (as done in 2.09) would start a new paragraph. It would be possible to inspect \prevdepth to check the depth of the just-completed paragraph, but we do not do that here. Actually we do even less now, skip the vmode test as it broke tabular ‘p’ columns.

The \nobreak was added (1995/10/31) to allow hyphenation of the final word of the paragraph.

In 2024 we changed the macro to account for vertical mode. In that case we use a strut produced with \hrule to avoid starting a new paragraph (resulting in spurious extra line) and also account for the \prevdepth of the previous line.

```

704 </2ekernel>
705 <*2ekernel | latexrelease>
706 <latexrelease>\IncludeInRelease{2024/06/01}%
707 <latexrelease> \{\@finalstrut\}{final strut correction}%
708 \def\@finalstrut#1{%
709 \unskip
710 \ifhmode \nobreak
711 \else

```

If we are in vmode we now back up by a baseline.

```

712 \vskip-\baselineskip
713 \fi

```

Finally we unconditionally use \vrule.

```

714 \vrule\@width\z@\@height\z@\@depth\dp#1}
715 </2ekernel | latexrelease>
716 <latexrelease>\EndIncludeInRelease
717 <latexrelease>\IncludeInRelease{0000/00/00}%
718 <latexrelease> \{\@finalstrut\}{final strut correction}%
719 <latexrelease>\def\@finalstrut#1{%
720 <latexrelease> \unskip\ifhmode\nobreak\fi
721 <latexrelease> \vrule\@width\z@\@height\z@\@depth\dp#1}
722 <latexrelease>
723 <latexrelease>\EndIncludeInRelease
724 <*2ekernel>

```

(End of definition for \@finalstrut.)

1.1 Some low-level constructs

The following commands are basically inherited from plain T_EX.

`\leftline` These macros place text on a full line either centered or left or right adjusted.

`\rightline` 725 `\def\@oline{\hb@xt@\hsize}`

`\centerline` 726 `\DeclareRobustCommand\leftline[1]{\@oline{#1\hss}}`

`\@oline` 727 `\DeclareRobustCommand\rightline[1]{\@oline{\hss#1}}`

728 `\DeclareRobustCommand\centerline[1]{\@oline{\hss#1\hss}}`

(End of definition for \leftline and others.)

`\rlap` These macros place text to the left or right of the current reference point without taking

`\llap` up space.

`\clap` 729 `\DeclareRobustCommand\rlap[1]{\hb@xt@\z@{#1\hss}}`

730 `\DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss#1}}`

And here is the version that centers, it was initially introduced by `mathtools`.

731 `\DeclareRobustCommand\clap[1]{\hb@xt@\z@{\hss#1\hss}}`

(End of definition for \rlap, \llap, and \clap.)

732 `\</2ekernel>`

File 41

lftab.dtx

1 Tabbing, Tabular and Array Environments

This section deals with ‘Lining It Up in Columns’. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original L^AT_EX 2.09 version, not the extended version described in *The L^AT_EX Companion*. Use the `array` package to obtain the extended version.

1.1 tabbing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\dimen(\@firsttab + i)` = distance of tab stop `i` from left margin
0 <= `i` <= 15 (?).

`\dimen\@firsttab` is initialized to `\@totalleftmargin`, so it starts at the prevailing left margin.

`\@maxtab` = number of highest defined tab register
probably = `\@firsttab + 12`

`\@nxttabmar` = tab stop number of next line’s left margin

`\@curtabmar` = tab stop number of current line’s left margin

`\@curtab` = number of the current tab. At start of line,
it equals `\@curtabmar`

`\@hightab` = largest tab number currently defined.

`\@tabpush` = depth of `\pushtab`’s

`\box\@curline` = contents of current line, excluding left margin
skip, and excluding contents of current field

`\box\@curfield` = contents of current field

`@rjfield` = switch: T iff the last field of the line should
be right-justified at the right margin.

`\tabbingsep` = distance left by the `\’` command between the
current position and the field that is
“left-shifted”.

UTILITY MACROS

`\@stopfield` : closes the current field

`\@addfield` : adds the current field to the current line.

`\@contfield` : continues the current field

`\@startfield` : begins the next field

`\@stopline` : closes the current line and outputs it

```

\@startline : starts the next line
\@ifatmargin : an \if that is true iff the current line.
                has width zero

\@startline ==
BEGIN
  \@curtabmar :=G \@nxttabmar
  \@curtab :=G \@curtabmar
  \box\@curline :=G null
  \@startfield
  \strut
END

\@stopline ==
BEGIN
  \unskip
  \@stopfield
  if @rjfield = T
    then @rjfield :=G F
      \@tempdima := \@totalleftmargin + \linewidth
      \hb@xt@ \@tempdima{\@itemfudge
                          \hskip \dimen\@curtabmar
                          \box\@curline
                          \hfil
                          \box\@curfield}
    else \@addfield
      \hbox {\@itemfudge
            \hskip \dimen\@curtabmar
            \box\@curline}
    fi
  END

\@startfield ==
BEGIN
  \box\@curfield :=G \hbox {
END

\@stopfield ==
BEGIN
  }
END

\@contfield ==
BEGIN
  \box\@curfield :=G \hbox { \unhbox\@currfield %%} brace matching
END

\@addfield ==
BEGIN
  \box\@curline :=G \unbox\@curline * \unbox\@curfield
END

```

```

\@ifatmargin ==
BEGIN
  if dim of box\@curline = 0pt then
  END

\tabbing ==
BEGIN
  \lineskip :=L 0pt
  \> == \@rtab
  \< == \@ltab
  \= == \@settab
  \+ == \@tabplus
  \- == \@tabminus
  \‘ == \@tabrj
  \’ == \@tablab
  \\\ == BEGIN \@stopline \@startline END
  \\[DIST] == BEGIN
    \@stopline \vskip DIST \@startline\ignorespaces END
  \\\* == BEGIN \@stopline \penalty 10000 \@startline END
  \\\*[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
    \@startline\ignorespaces END
  \@hightab := \@nxttabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totalleftmargin
  @rjfield :=G F
  \trivlist \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabfbox = \rlap{\indent\the\everypar}
    % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabfbox END
  \@startline
  \ignorespaces
END

\@endtabbing ==
BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs" fi
  \endtrivlist
END

\@rtab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@hightab
    then \@curtab :=G \@curtab + 1
    else error message "Undefined Tab" fi

```



```

\@tempdima := \dimen\@curtab - \dimen\@curtabmar
              - width of box \@curline
\box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
\@startfield
END

\@settab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@maxtab
    then \@curtab :=G \@curtab+1
    else error message: "Too many tabs"    fi
  if \@curtab > \@hightab
    then \@hightab :=L \@curtab    fi
  \dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
  \@startfield
END

\@ltab ==
BEGIN
  \@ifatmargin
  then if \@curtabmar > \@firsttab
    then \@curtab :=G \@curtab - 1
        \@curtabmar :=G \@curtabmar - 1
    else error message "Too many untab"    fi
  else error message "Left tab in middle of line"
  fi
END

\@tabplus ==
BEGIN
  if \@nxttabmar < \@hightab
    then \@nxttabmar :=G \@nxttabmar+1
    else error message "Undefined tab"
  fi
END

\@tabminus ==
BEGIN
  if \@nxttabmar > \@firsttab
    then \@nxttabmar :=G \@nxttabmar-1
    else error message "Too many untab"
  fi
END

\@tabrj ==
BEGIN \@stopfield
  \@addfield
  @rjfield :=G T

```

```

\@startfield
END

\@tablab ==
BEGIN \@stopfield
\box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
\hskip - width of \box\@curfield
\hskip -\tabbingsep
\box\@curfield
\hskip \tabbingsep }

\@startfield
END

\pushtabs ==
BEGIN
\@stopfield
\@tabpush :=G \@tabpush + 1
\beginngroup
\@contfield
END

\poptabs ==
BEGIN
\@stopfield
if \@tabpush > 0
then \endgroup
\@tabpush :=G \@tabpush - 1
else error message: "Too many \poptabs"
fi
\@contfield
END

```

End of historical L^AT_EX 2.09 comments.

\a The accents \‘, \’ , and \= that have been redefined inside a tabbing environment can be called by typing \a‘, \a’ , and \a=. The macro \a is defined in `ltoutenc.dtx`.

(End of definition for \a.)

The ‘2ekernel’ code ensures that a `\usepackage{autotabg}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```

1 <2ekernel>\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion

\@firsttab
\@maxtab
2 <*2ekernel>
3 \newdimen\@gtempa
4 \chardef\@firsttab=\the\allocationnumber
5 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
6 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
7 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
8 \newdimen\@gtempa
9 \chardef\@maxtab=\the\allocationnumber
10 \dimen\@firsttab=0pt

```

```

        (End of definition for \@firsttab and \@maxtab.)

\@nxttabmar
\@curtabmar 11 \newcount\@nxttabmar
  \@curtab 12 \newcount\@curtabmar
  \@hightab 13 \newcount\@curtab
  \@tabpush 14 \newcount\@hightab
            15 \newcount\@tabpush

        (End of definition for \@nxttabmar and others.)

\@curline
\@curfield 16 \newbox\@curline
\@tabfbox 17 \newbox\@curfield
            18 \newbox\@tabfbox

        (End of definition for \@curline, \@curfield, and \@tabfbox.)

\if@rjfield
            19 \newif\if@rjfield

        (End of definition for \if@rjfield.)

\@startline It is, in some sense, an error if the current margin tab setting is higher than the value of
            \@hightab (which is a local variable). That this is allowed is a fundamental design flaw
            which is not going to be corrected now.
            20 \def\@startline{%
            21     \ifnum \@nxttabmar >\@hightab
            22     \badtab
            23     \global\@nxttabmar \@hightab
            24     \fi
            25     \global\@curtabmar \@nxttabmar
            26     \global\@curtab \@curtabmar
            27     \global\setbox\@curline \hbox {}%
            28     \@startfield
            29     \strut}

        (End of definition for \@startline.)

\@stopline
            30 \def\@stopline{%
            31     \unskip
            32     \@stopfield
            33     \if@rjfield
            34     \global\@rjfieldfalse
            35     \@tempdima\@totalleftmargin
            36     \advance\@tempdima\linewidth
            37     \hb@xt@\@tempdima{%
            38         \@itemfudge\hskip\dimen\@curtabmar
            39         \box\@curline
            40         \hfil
            41         \box\@curfield}%
            42     \else
            43     \@addfield
            44     \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
            45     \fi}

```

```

(End of definition for \@stopline.)

\@startfield
46 \def\@startfield{%
47   \global\setbox\@curfield\hbox\bgroup\color@begingroup}

(End of definition for \@startfield.)

\@stopfield
48 \def\@stopfield{%
49   \color@endgroup\egroup}

(End of definition for \@stopfield.)

\@contfield
50 \def\@contfield{%
51   \global\setbox\@curfield\hbox\bgroup\color@begingroup
52   \unhbox\@curfield}

(End of definition for \@contfield.)

\@addfield
53 \def\@addfield{\global\setbox\@curline\hbox{\unhbox
54   \@curline\unhbox\@curfield}}

(End of definition for \@addfield.)

\@ifatmargin
55 \def\@ifatmargin{\ifdim \wd\@curline =\z@}

(End of definition for \@ifatmargin.)

\@tabcr
56 \protected\def\@tabcr{\@stopline \ifstar{\penalty \@M \@tabcr}\@tabcr}

(End of definition for \@tabcr.)

\@xtabcr
57 \def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}

(End of definition for \@xtabcr.)

\@itabcr
58 \</2ekernel>
59 \< *2ekernel | latexrelease>
60 \< latexrelease>\IncludeInRelease{2020/10/01}%
61 \< latexrelease>          {\@itabcr}{Tabbing calc syntax}%
62 \def\@itabcr[#1]{\@vspace@calcify{#1}\@startline\ignorespaces}
63 \</2ekernel | latexrelease>
64 \< latexrelease>\EndIncludeInRelease
65 \< latexrelease>\IncludeInRelease{0000/00/00}%
66 \< latexrelease>          {\@itabcr}{Tabbing calc syntax}%
67 \< latexrelease>
68 \< latexrelease>\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
69 \< latexrelease>\EndIncludeInRelease
70 \< *2ekernel>

```

tabbing (env.) We use \relax to prevent \item from scanning too far.

```
\tabbing 71 \def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
72 \let\+\@tabplus\let\-\@tabminus\let\'\@tabrj\let\'\@tablab
73 \let\=\@tabcr
74 \@hightab\@firsttab
75 \global\@nxttabmar\@firsttab
76 \dimen\@firsttab\@totalleftmargin
77 \global\@tabpush\z@ \global\@rjfieldfalse
78 \trivlist \item\relax
79 \if@minipage\else\vskip\parskip\fi

80 \setbox\@tabfbox\hbox{%
81 \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
82 \def\@itemfudge{\box\@tabfbox}%
83 \@startline\ignorespaces}
```

```
\endtabbing 84 \def\endtabbing{%
85 \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}
```

Omitted \global added to \@rtab 17 Jun 86

```
\@rtab 86 \def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
87 \global\advance\@curtab \@ne \else\@badtab\fi
88 \@tempdima\dimen\@curtab
89 \advance\@tempdima -\dimen\@curtabmar
90 \advance\@tempdima -\wd\@curline
91 \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
92 \@startfield\ignorespaces}
```

```
\@settab 93 \def\@settab{\@stopfield\@addfield
94 \ifnum \@curtab <\@maxtab
95 \ifnum\@curtab =\@hightab
96 \advance\@hightab \@ne
97 \fi
98 \global\advance\@curtab \@ne
99 \else
100 \@latexerror{Tab overflow}\@ehd
101 \fi
102 \dimen\@curtab \dimen\@curtabmar
103 \advance\dimen\@curtab \wd\@curline
104 \@startfield
105 \ignorespaces}
```

```
\@ltab 106 \def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
107 \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
108 \@badtab\fi\else
109 \@latexerror{\string\<\space in mid line}\@ehd\fi\ignorespaces}
```

```
\@tabplus 110 \def\@tabplus{%
111 \ifnum\@nxttabmar<\@hightab
```

```

112     \global\advance\@nxttabmar\@ne
113 \else
114     \@badtab
115 \fi
116 \ignorespaces}

\@tabminus 117 \def\@tabminus{%
118     \ifnum\@nxttabmar>\@firsttab
119         \global\advance\@nxttabmar\m@ne
120     \else
121         \@badtab
122     \fi
123     \ignorespaces}

\@tabrj 124 \def\@tabrj{%
125     \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab 126 \def\@tablab{%
127     \@stopfield
128     \global\setbox\@curline\hbox{%
129         \box\@curline
130         \hskip-\wd\@curfield \hskip-\tabbingsep
131         \box\@curfield
132         \hskip\tabbingsep}%
133     \@startfield
134     \ignorespaces}

135 \</2ekernel>
136 \<*2ekernel | latexrelease>
137 \<latexrelease>\IncludeInRelease{2019/10/01}%
138 \<latexrelease>                {\pushtabs}{Make commands robust}%

\pushtabs 139 \DeclareRobustCommand\pushtabs{%
140     \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
141         \@contfield}

\poptabs It is, in some sense, an error if, after the endgroup, the current tab setting is higher
than the new value of \@hightab (which is a local variable). That this is allowed is a
fundamental design flaw which is not going to be corrected now.

142 \DeclareRobustCommand\poptabs{\@stopfield\@addfield
143     \ifnum \@tabpush >\z@
144         \endgroup
145         \global\advance\@tabpush \m@ne
146         \ifnum \@curtab >\@hightab
147             \global \@curtab \@hightab
148             \@badtab
149         \fi
150     \else
151         \@badpoptabs
152     \fi
153     \@contfield}

```

```

154 \DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}
(End of definition for \@itabcr and others.)
155 </2ekernel| latexrelease>
156 <latexrelease>\EndIncludeInRelease
157 <latexrelease>\IncludeInRelease{0000/00/00}%
158 <latexrelease>                {\pushtabs}{Make commands robust}%
159 <latexrelease>
160 <latexrelease>\kernel@make@fragile\pushtabs
161 <latexrelease>\kernel@make@fragile\poptabs
162 <latexrelease>\kernel@make@fragile\kill
163 <latexrelease>
164 <latexrelease>\EndIncludeInRelease
165 <*2ekernel>

```

\tabbingsep

```

166 \newdimen\tabbingsep
(End of definition for \tabbingsep.)

```

1.2 array and tabular environments

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

ARRAY PARAMETERS:

`\arraycolsep`
 : half the width separating columns in an array environment
`\tabcolsep`
 : half the width separating columns in a tabular environment
`\arrayrulewidth`
 : width of rules
`\doublerulesep`
 : space between adjacent rules in array or tabular
`\arraystretch`
 : line spacing in array and tabular environments is done by
 placing a strut in every row of height and depth
 `\arraystretch` times the height and depth of the strut
 produced by an ordinary `\strut` command.

PREAMBLE:

The PREAMBLE argument of an array or tabular environment can contain the following:

`l,r,c` : indicate where entry is to be placed.
`|` : for vertical rule
`@{EXP}` : inserts the text EXP in every column.
 `\arraycolsep` or `\tabcolsep` spacing is suppressed.
`*{N}{PRE}` : equivalent to writing N copies of PRE in the preamble.
 PRE may contain `*{N'}{EXP'}` expressions.
`p{LEN}` : makes entry in parbox of width LEN.

SPECIAL ARRAY COMMANDS:

`\multicolumn{N}{FORMAT}{ITEM}` : replaces the next N column items by ITEM, formatted according to FORMAT. FORMAT should contain at most one l,r or c. If it contains none, then ITEM is ignored.

`\vline` : draws a vertical line the height of the current row. May appear in an array element entry.

`\hline` : draws a horizontal line between rows. Must appear either before the first entry (to appear above the first row) or right after a `\\` command. If followed by another `\hline`, then adds a `\vskip` of `\doublerulesep`.

`\cline{i-j}` : draws horizontal lines between rows covering columns i through j, inclusive. Multiple commands may follow one another to provide lines covering several disjoint columns

`\extracolsep{WIDTH}` : for use inside an @ in the preamble. Causes a WIDTH space to be added between columns for the rest of the columns. This is in addition to the ordinary intercolumn space.

```

\array ==
  BEGIN
    \@acol    == \@arrayacol
    \@classz  == \@arrayclassz
    \@classiv == \@arrayclassiv
    \\        == \@arraycr
    \@halignto == NULL
    \@tabarray
  END

\endarray{NAME} == BEGIN \crrc }} END

\tabular ==
  BEGIN
    \@halignto == NULL
    \@tabular
  END

\tabular*{WIDTH} ==
  BEGIN
    \@halignto == to WIDTH
    \@tabular
  END

\@tabular ==
  BEGIN
    \leavevmode
    \hbox { $
      \@acol    == \@tabacol

```



```

\@classz == \@tabclassz
\@classiv == \@tabclassiv
\\ == \@tabularcr
\@tabarray
END

\endtabular == BEGIN \crrc}} $} END

\@tabarray == if next char = [ then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
BEGIN
  define \@arstrutbox to make \@arstrut produce strut of height
    and depth \arraystretch times the height and
    depth of a normal strut.
  \@mkpream{PREAMBLE}
  \@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
    eval{\@preamble}\tabskip = 0pt\cr %%}
  \@startpbox == \@@startpbox
  \@endpbox == \@@endpbox
  if POS = t then \vtop
    else if POS = b then \vbox
      else \vcenter

    fi
  fi
  {
    \par ==L {} % changed 92/09/18
    \@sharp == #
    \protect == \relax
    \lineskip :=L 0pt
    \baselineskip :=L 0pt
    \@preamble
  }
END

\@arraycr ==
BEGIN
  $ %% Prevents extra space at end of row's last entry.
  if next char = [
    then \@argarraycr
    else $ \cr %% Needed to balance $
  }
END

\@argarraycr[LENGTH] ==
BEGIN
  $ %% Needed to balance $ of \@arraycr
  if LENGTH > 0
    then \@tempdima := depth of \@arstrutbox + LENGTH
      \vrule height 0pt width 0pt depth \@tempdima
      \cr
    else \cr \noalign{\vskip LENGTH}
  }

```

END

`\@tabularcr` and `\@argtabularcr` same as `\@arraycr` and `\@argarraycr`
except without the extra `$`'s.

End of historical L^AT_EX 2.09 comments.

`\extracolsep` This command needs to expand during the tabular preamble construction so can't be robust.

```
167 \def\extracolsep#1{\tabskip #1\relax}
```

(End of definition for \extracolsep.)

`\array`

```
168 \</2ekernel>
169 \<latexrelease>\IncludeInRelease{2026/06/01}%
170 \<latexrelease>          {\array}{Use real text vcenter}%
171 \<*2ekernel | latexrelease>
172 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
173   \let\@classiv\@arrayclassiv
174   \let\\\@arraycr\let\@halignto\@empty\m@th\@tabarray}
175 \</2ekernel | latexrelease>
176 \<latexrelease>\EndIncludeInRelease
177 \<latexrelease>\IncludeInRelease{0000/00/00}%
178 \<latexrelease>          {\array}{Use real text vcenter}%
179 \<latexrelease>\def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
180 \<latexrelease> \let\@classiv\@arrayclassiv
181 \<latexrelease> \let\\\@arraycr\let\@halignto\@empty\@tabarray}
182 \<latexrelease>\EndIncludeInRelease
183 \<*2ekernel>
```

(End of definition for \array.)

`\endarray`

`\endtabular`

`\endtabular*`

```
184 \def\endarray{\crrc\egroup\egroup}
185 \</2ekernel>
186 \<latexrelease>\IncludeInRelease{2026/06/01}%
187 \<latexrelease>          {\endtabular}{Use real text vcenter}%
188 \<*2ekernel | latexrelease>
189 \def\endtabular{\crrc\egroup\egroup\egroup}
190 \</2ekernel | latexrelease>
191 \<latexrelease>\EndIncludeInRelease
192 \<latexrelease>\IncludeInRelease{0000/00/00}%
193 \<latexrelease>          {\endtabular}{Use real text vcenter}%
194 \<latexrelease>\def\endtabular{\crrc\egroup\egroup $\egroup}
195 \<latexrelease>\EndIncludeInRelease
196 \<*2ekernel>
197 \expandafter \let \csname endtabular*\endcsname = \endtabular
```

(End of definition for \endarray, \endtabular, and \endtabular.)*

`\tabular`

```
198 \def\tabular{\let\@halignto\@empty\@tabular}
```

(End of definition for \tabular.)

`\tabular*` Note that the change to use `\setlength` slightly alters the timing of the expansion and use of the length in `#1` but this is very unlikely to have any practical effect.

```
199 \namedef\tabular*#1{%
200   \setlength\dimen@{#1}%
201   \edef\@halignto{to\the\dimen@}\@tabular}
```

(End of definition for \tabular.)*

`\@tabular`

```
202 \</2ekernel>
203 \<latexrelease>\IncludeInRelease{2026/06/01}%
204 \<latexrelease>          {\@tabular}{Use real text vcenter}%
205 \<*2ekernel | latexrelease>
206 \def\@tabular{\leavevmode \hbox \bgroup \let\@acol\@tabacol
207   \let\@classz\@tabclassz
208   \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
209 \</2ekernel | latexrelease>
210 \<latexrelease>\EndIncludeInRelease
211 \<latexrelease>\IncludeInRelease{0000/00/00}%
212 \<latexrelease>          {\@tabular}{Use real text vcenter}%
213 \<latexrelease>\def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
214 \<latexrelease>   \let\@classz\@tabclassz
215 \<latexrelease>   \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
216 \<latexrelease>\EndIncludeInRelease
217 \<*2ekernel>
```

(End of definition for \@tabular.)

`\@tabarray` RmS 91/11/04 added `\m@th`.

```
218 \</2ekernel>
219 \<latexrelease>\IncludeInRelease{2026/06/01}%
220 \<latexrelease>          {\@tabarray}{Use real text vcenter}%
221 \<*2ekernel | latexrelease>
222 \def\@tabarray{\ifnextchar[\@array{\@array[c]}}
223 \</2ekernel | latexrelease>
224 \<latexrelease>\EndIncludeInRelease
225 \<latexrelease>\IncludeInRelease{0000/00/00}%
226 \<latexrelease>          {\@tabarray}{Use real text vcenter}%
227 \<latexrelease>\def\@tabarray{\m@th\ifnextchar[\@array{\@array[c]}}
228 \<latexrelease>\EndIncludeInRelease
229 \<*2ekernel>
```

(End of definition for \@tabarray.)

RmS 1993/11/03 changed `\halign` to `\ialign` and removed superfluous `\tabskip` assignment

`\@array`

```
230 \</2ekernel>
231 \<latexrelease>\IncludeInRelease{2026/06/01}%
232 \<latexrelease>          {\@array}{Use real text vcenter}%
233 \<*2ekernel | latexrelease>
234 \def\@array[#1]#2{%
235   \if #1t\top \else \if#1b\box \else\vcenter@text \fi\fi
236   \bgroup
```

This next bit of code sets up the strut and then builds the halign and its preamble according to the specification in the second argument.

This code has been moved inside the box. A side effect of this has been to expose what was a buglet in the previous version: since the `\@arstrut` below is expanded and contains an `\ifmode` then it could produce an unnecessary extra box in every row, thus wasting ‘lots of’ main memory.

```

237 \setbox\@arstrutbox\hbox{%
238 \vrule \@height\arraystretch\ht\strutbox
239 \@depth\arraystretch \dp\strutbox
240 \@width\z@}%
241 \@mkpream{#2}%
242 \edef\@preamble{%
243 \ialign \noexpand\@halignto
244 \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%

```

That is the end of setting up the preamble; now we reset things before executing the halign built-up in `\@preamble`. The restorations could be done by introducing an extra group, thus saving tokens.

```

245 \let\@startpbox\@startpbox \let\@endpbox\@endpbox
246 \let\tabularnewline\\%
247 \def\par{\ifnum\currentgrouptype=6 \else\@par\fi}%
248 \let\@sharp##%
249 \set@typeset@protect
250 \lineskip\z@skip\baselineskip\z@skip

```

If the parsing of the preamble goes wrong there may be some characters left which $\text{\TeX}$ then tries to typeset, i.e., we would be in horizontal mode. That would produce an endless loop because the `\halign` expects vertical mode thus issues a `\par` but that is a no-op at this point. So we better test this case issue some error message and make a crude recovery by ending that horizontal mode with force. A better fix would be to ensure that we never pick up more than a single character token (not done).

```

251 \ifmode \@preamerr\z@ \@@par\fi
252 \@preamble}
253 </2kernel | latexrelease>
254 <latexrelease>\EndIncludeInRelease
255 <latexrelease>\IncludeInRelease{0000/00/00}%
256 <latexrelease> \{\@array\}{Use real text vcenter}%
257 <latexrelease>\def\@array[#1]#2{%
258 <latexrelease> \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
259 <latexrelease> \bgroup
260 <latexrelease> \setbox\@arstrutbox\hbox{%
261 <latexrelease> \vrule \@height\arraystretch\ht\strutbox
262 <latexrelease> \@depth\arraystretch \dp\strutbox
263 <latexrelease> \@width\z@}%
264 <latexrelease> \@mkpream{#2}%
265 <latexrelease> \edef\@preamble{%
266 <latexrelease> \ialign \noexpand\@halignto
267 <latexrelease> \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
268 <latexrelease> \let\@startpbox\@startpbox \let\@endpbox\@endpbox
269 <latexrelease> \let\tabularnewline\\%
270 <latexrelease> \def\par{\ifnum\currentgrouptype=6 \else\@par\fi}%
271 <latexrelease> \let\@sharp##%
272 <latexrelease> \set@typeset@protect

```

```

273 <latexrelease> \lineskip\z@skip\baselineskip\z@skip
274 <latexrelease> \ifhmode \@preamerr\z@ \@@par\fi
275 <latexrelease> \@preamble}
276 <latexrelease>\EndIncludeInRelease
277 <*2ekernel>

```

(End of definition for \@array.)

\@arraycr Array version of \\.

```

278 \protected\def\@arraycr{%
279   ${\ifnum0='}\fi\@ifstar\@xarraycr\@xarraycr}

```

(End of definition for \@arraycr.)

\@arraycr

```

280 \def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0='{ \fi}$\}\cr}}

```

(End of definition for \@arraycr.)

\@argarraycr

```

281 \def\@argarraycr[#1]{%
282   \ifnum0='{ \fi}$\}\ifdim #1>\z@ \@xargarraycr{#1}\else
283   \@yargarraycr{#1}\fi}

```

(End of definition for \@argarraycr.)

\tabularnewline Tabular version of \\.

```

284 \let\tabularnewline\relax

```

(End of definition for \tabularnewline.)

\@tabularcr

```

285 \protected\def\@tabularcr{%
286   {\ifnum0='}\fi\@ifstar\@xtabularcr\@xtabularcr}

```

(End of definition for \@tabularcr.)

\@xtabularcr

```

287 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{ \fi}\cr}}

```

(End of definition for \@xtabularcr.)

\@argtabularcr

```

288 \def\@argtabularcr[#1]{%
289   \ifnum0='{ \fi}%
290   \ifdim #1>\z@
291     \unskip\@xargarraycr{#1}%
292   \else
293     \@yargarraycr{#1}%
294   \fi}

```

(End of definition for \@argtabularcr.)

\@xargarraycr

```

295 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
296   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}

```

(End of definition for \@xargarraycr.)

\@yargarraycr

```

297 </2ekernel>
298 <*2ekernel | latexrelease>
299 <latexrelease>\IncludeInRelease{2020/10/01}%
300 <latexrelease>                {\@yargarraycr}{tabular support calc syntax}%
301 \def\@yargarraycr#1{\cr\noalign{\@vspace@calcify{#1}}}
302 </2ekernel | latexrelease>
303 <latexrelease>\EndIncludeInRelease
304 <latexrelease>\IncludeInRelease{0000/00/00}%
305 <latexrelease>                {\@yargarraycr}{tabular support calc syntax}%
306 <latexrelease>
307 <latexrelease>\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
308 <latexrelease>\EndIncludeInRelease
309 <*2ekernel>

```

(End of definition for \@yargarraycr.)

\multicolumn *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\multicolumn{NUMBER}{FORMAT}{ITEM} ==
BEGIN
\multispan{NUMBER}
\beginngroup
\@addamp == null
\@mkpream{FORMAT}
\@sharp == ITEM
\protect == \relax
\@startpbox == \@@startpbox
\@endpbox == \@@endpbox
\@arstrut
\@preamble
\endgroup
END

```

End of historical L^AT_EX 2.09 comments.

The command \def\@addamp{} was removed from \multicolumn on 6 Dec 86 because it caused embedded array environments not to work. I think that it was included originally to prevent an error message if the 2nd argument to the \multicolumn command had two column specifiers.

8 Feb 89 — \hbox{} added after \@preamble to correct bug that occurred if \multicolumn preceded \[D] with D > 0, caused by \[] command doing an \unskip, which removed \tabcolsep glue inserted by \multicolumn.

This has been made long so that, for example, a p-column can contain multiple paragraphs; maybe the arguments of @-expressions should also be able to contain multiple paragraphs.

```

310 \long\def\multicolumn#1#2#3{\multispan{#1}\beginngroup
311   \@mkpream{#2}%
312   \def\@sharp{#3}\set@typeset@protect
313   \let\@startpbox\@@startpbox\let\@endpbox\@@endpbox
314   \@arstrut \@preamble\hbox{}\endgroup\ignorespaces}

```

(End of definition for `\multicolumn`.)

Historical *LaTeX* 2.09 comments (not necessarily accurate any more):

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
l	1	-
@	2	-
p	3	-
{@-exp}	4	-
{p-arg}	5	-

`\@testpach \foo` : expands `\foo`, which should be an array parameter token, and sets `\@chclass` and `\@chnum` to its class and number. Uses `\@lastchclass` to distinguish 4 and 5

Preamble error codes

0: 'illegal character'
 1: 'Missing @-exp'
 2: 'Missing p-arg'

```
\@addamp ==
  BEGIN if \@firstamp = true then \@firstamp := false
        else &
        fi
  END
```

```
\@mkpream TOKENLIST ==
  BEGIN
    \@firstamp      := T
    \@lastchclass   := 6
    \@preamble      == null
    \@sharp         == \relax
    \@protect       == BEGIN \noexpand\protect\noexpand END
    \@startpbox     == \relax
    \@endpbox       == \relax
    \@expast{TOKENLIST}
    for \@nextchar := expand(\reserved@a)
    do \@testpach{\@nextchar}
      case of \@chclass
        0 -> \@classz
        1 -> \@classi
        ...
        5 -> \@classv
      end case
    \@lastchclass := \@chclass
```

```

od
case of \@lastchclass
  0 -> \hskip \arraycolsep          % lrc
  1 ->                               % l
  2 -> \@preamerr1 % 'Missing @-exp' % @
  3 -> \@preamerr2 % 'Missing p-arg' % p
  4 ->                               % @-exp
  5 -> \hskip \arraycolsep          % p-exp
end case
END

\@arrayclassz ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass
    0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
    1 -> \@addamp \hskip \arraycolsep
    2 -> % impossible
    3 -> % impossible
    4 -> \@addamp
    5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
    6 -> \@addamp \hskip \arraycolsep
  end case
  * case of \@chnum
    0 -> \hfil$\relax\@sharp$\hfil
    1 -> $\relax\@sharp$\hfil
    2 -> \hfil$\relax\@sharp$
  end case
END

\@tabclassz == similar to \@arrayclassz

\@classi ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass
    0 -> \hskip \arraycolsep \@arrayrule
    1 -> \hskip \doublerulesep \@arrayrule
    2 -> % impossible
    3 -> % impossible
    4 -> \@arrayrule
    5 -> \hskip \arraycolsep \@arrayrule
    6 -> \@arrayrule
  end case
END

\@classii ==
BEGIN
  \@preamble := \@preamble *
  case of \@lastchclass

```



```

0      ->
1      -> \hskip .5\arrayrulewidth
2      -> % impossible
else ->
end case

END

\@classiii ==
BEGIN
  \@preamble := \@preamble *
    case of \@lastchclass
      0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
      1 -> \@addamp \hskip \arraycolsep
      2 -> % impossible
      3 -> % impossible
      4 -> \@addamp
      5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
      6 -> \@addamp \hskip \arraycolsep
    end case
END

\@arrayclassiv ==
  BEGIN \@preamble := \@preamble * $ \@nextchar$  END

\@tabclassiv == same as \@arrayclassv except without the $ ... $

\@classv ==
  BEGIN
    \@preamble :=
      \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
                      \@endpbox
  END

\@expast{S}:
  Sets \reserved@a := S with all instances of *{N}{STRING}
  replaced by N copies of STRING, where N > 0. An *
  appearing inside braces is ignored, but *-expressions
  inside STRING are expanded, so nested *-expressions are
  handled properly.

\@expast{S} == BEGIN \@expast S *0x\@@  END

\@expast S1 *{N}{S2} S3 \@@ ==
  BEGIN
    \reserved@a := S1
    \@tempcnta := N
    if \@tempcnta > 0
      then while \@tempcnta > 0 do \reserved@a := \reserved@a S2
                                \@tempcnta := \@tempcnta - 1 od
    \reserved@b == \@expast
  
```

```

        else \reserved@b == \@xexnoop
      fi
      \expandafter \reserved@b \reserved@a S3 \@
    END
  End of historical LATEX 2.09 comments.

\@xexnoop
315 \def\@xexnoop #1\@{ }
  (End of definition for \@xexnoop.)

\@expast
316 \def\@expast#1{\@expast #1*0x\@}
  (End of definition for \@expast.)

\@expast
317 \def\@expast#1*#2#3#4\@{ %
318   \edef\reserved@a{#1}%
319   \@tempcnta#2\relax
320   \ifnum\@tempcnta>\z@
321     \@whilenum\@tempcnta>\z@\do
322       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
323   \let\reserved@b\@expast
324   \else
325     \let\reserved@b\@xexnoop
326   \fi
327   \expandafter\reserved@b\reserved@a #4\@}
  (End of definition for \@expast.)

\if@firstamp
  \@addamp
328 \newif\if@firstamp
329 \def\@addamp{%
330   \if@firstamp
331     \@firstampfalse
332   \else
333     \edef\@preamble{\@preamble &}%
334   \fi}
  (End of definition for \if@firstamp and \@addamp.)

\@arrayacol
  \@tabacol
335 \def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
  \@ampacol
336 \def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\@acolampacol
337 \def\@ampacol{\@addamp \@acol}
338 \def\@acolampacol{\@acol\@addamp\@acol}
  (End of definition for \@arrayacol and others.)

```

\@mkpream

```

339 \def\@mkpream#1{\@firstampttrue\@lastchclass6
340 \let\@preamble\@empty
341 \let\protect\@unexpandable@protect
342 \let\@sharp\relax
343 \let\@startpbox\relax\let\@endpbox\relax
344 \@expast{#1}%
345 \expandafter\@tfor \expandafter
346 \@nextchar \expandafter:\expandafter=\reserved@a\do
347 {\@testpach\@nextchar
348 \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
349 \or \@classiv \or \@classv \fi\@lastchclass\@chclass}%
350 \ifcase \@lastchclass \@acol
351 \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}

```

(End of definition for \@mkpream.)

\@arrayclassz

```

352 \def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
353 \or \or \@addamp \or
354 \@acolampacol \or \@firstampfalse \@acol \fi
355 \edef\@preamble{\@preamble
356 \ifcase \@chnum
357 \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
358 \or \hfil$\relax\@sharp$\fi}}

```

(End of definition for \@arrayclassz.)

\@tabclassz RmS 91/08/14 inserted extra braces around entry for NFSS

```

359 \def\@tabclassz{%
360 \ifcase\@lastchclass
361 \@acolampacol
362 \or
363 \@ampacol
364 \or
365 \or
366 \or
367 \@addamp
368 \or
369 \@acolampacol
370 \or
371 \@firstampfalse\@acol
372 \fi
373 \edef\@preamble{%
374 \@preamble{%
375 \ifcase\@chnum
376 \hfil
377 \hskip1sp%
378 \ignorespaces\@sharp\unskip\hfil
379 \or
380 \hskip1sp\ignorespaces\@sharp\unskip\hfil
381 \or
382 \hfil\hskip1sp\ignorespaces\@sharp\unskip
383 \fi}}

```

(End of definition for \@tabclassz.)

\@classi

```
384 \def\@classi{%
385   \ifcase\@lastchclass
386     \@acol\@arrayrule
387   \or
388     \@addtopreamble{\hskip \doublerulesep}\@arrayrule
389   \or
390   \or
391   \or
392     \@arrayrule
393   \or
394     \@acol\@arrayrule
395   \or
396     \@arrayrule
397   \fi}
```

(End of definition for \@classi.)

\@classii

```
398 \def\@classii{%
399   \ifcase\@lastchclass
400   \or
401     \@addtopreamble{\hskip .5\arrayrulewidth}%
402   \fi}
```

(End of definition for \@classii.)

\@classiii

```
403 \def\@classiii{\ifcase \@lastchclass \@acolampacol \or
404   \@addamp\@acol \or
405   \or \or \@addamp \or
406   \@acolampacol \or \@ampacol \fi}
```

(End of definition for \@classiii.)

\@tabclassiv

```
407 \def\@tabclassiv{\@addtopreamble\@nextchar}
```

(End of definition for \@tabclassiv.)

\@arrayclassiv

```
408 \def\@arrayclassiv{\@addtopreamble{\@nextchar$}}
```

(End of definition for \@arrayclassiv.)

\@classv

```
409 \def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
410   \@sharp\@endpbox}}
```

(End of definition for \@classv.)

\@addtopreamble

```
411 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}
```

(End of definition for \@addtopreamble.)

```
\@chclass
\@lastchclass 412 \newcount\@chclass
\@chnum 413 \newcount\@lastchclass
414 \newcount\@chnum
```

(End of definition for \@chclass, \@lastchclass, and \@chnum.)

```
\arraycolsep
\tabcolsep 415 \newdimen\arraycolsep
\arrayrulewidth 416 \newdimen\tabcolsep
\doublerulesep 417 \newdimen\arrayrulewidth
418 \newdimen\doublerulesep
```

(End of definition for \arraycolsep and others.)

```
\arraystretch
419 \def\arraystretch{1} % Default value.
```

(End of definition for \arraystretch.)

```
\@arstrutbox
\@arstrut 420 \newbox\@arstrutbox
421 \def\@arstrut{%
422 \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}
```

(End of definition for \@arstrutbox and \@arstrut.)

```
\@arrayrule
423 \def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
424 \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

(End of definition for \@arrayrule.)

```
\@testpach
425 \def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
426 \ifnum \@lastchclass=3 5 \else
427 \z@ \if #1c\@chnum \z@ \else
428 \if #1l\@chnum \@ne \else
429 \if #1r\@chnum \tw@ \else
430 \@chclass \if #1|\@ne \else
431 \if #1@\tw@ \else
432 \if #1p3 \else \z@ \@preamerr 0\fi
433 \fi \fi \fi \fi \fi \fi
434 \fi}
```

(End of definition for \@testpach.)

```
\hline
435 \def\hline{%
436 \noalign{\ifnum0='}\fi\hrule \@height \arrayrulewidth \futurelet
437 \reserved@a\@xhline}
```

(End of definition for \hline.)

`\@xhline`

```
438 \def\@xhline{\ifx\reserved@a\hline
439         \vskip\doublerulesep
Measure from the middle of the rules.
440         \vskip-\arrayrulewidth
441         \fi
442         \ifnum0='{ \fi}}
```

(End of definition for \@xhline.)

`\vline`

```
443 \def\vline{\vrule \@width \arrayrulewidth}
```

(End of definition for \vline.)

`\cline` The old L^AT_EX2.09 implementation of `\cline` used up quite a lot of memory and two
`\@cline` precious count registers. This new (1995/09/14) implementation does not use any count registers. It is coded in a way that depends heavily on the definition of `\multispan` so that command has been moved here from the file `ltplain.dtx`.

These counters are no longer declared.

```
\newcount\@cla
```

```
\newcount\@clb
```

```
444 \def\cline#1{\@cline#1\@nil}
```

```
445 \def\@cline#1-#2\@nil{%
```

```
446     \omit
```

Use the counter from `\multispan`.

```
447     \@multicnt#1%
```

```
448     \advance\@multispan\m@ne
```

```
449     \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
```

```
450     \@multicnt#2%
```

```
451     \advance\@multicnt-#1%
```

```
452     \advance\@multispan\@ne
```

The original had `\unskip` at this point, but how could a skip get here ???

```
453     \leaders\hrule\@height\arrayrulewidth\hfill
```

```
454     \cr
```

This is back spacing is fairly horrible, but it is what happened in the old version... An alternative would be to make `\cline` look ahead for a following `\cline` as does `\hline`. This would alter the spacing in existing documents so keep the old version in the kernel. Perhaps a package should do this differently.

```
455     \noalign{\vskip-\arrayrulewidth}}
```

(End of definition for \cline and \@cline.)

`\mscount` The `\mscount` counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

(End of definition for \mscount.)

`\multispan` Modify `\multispan` slightly from its plain T_EX definition to allow more efficient code sharing with `\multicolumn`. Also share a count register with `\multipt`.

`\@multispan`

```

\sp@n 456 \def\multispan{\omit\@multispan}
457 \def\@multispan#1{%
458   \@multicnt#1\relax
459   \loop\ifnum\@multicnt>\@ne \sp@n\repeat}
460 \def\sp@n{\span\omit\advance\@multicnt\m@ne}

```

(End of definition for `\multispan`, `\@multispan`, and `\sp@n`.)

`\@startpbox` Helper macros for ‘p’ columns.

`\@endpbox` `\@startpbox{<width>} text \egroup` is essentially `\parbox{<width>}{<text>}`
`\@endpbox` is essentially `\unskip \strut \par \egroup\hfil` (Changed 14 Jan 89)
(changed again 1994/05/13)

```

461 \def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
462 \def\@endpbox{\@finalstrut\@arstrutbox\par\egroup\hfil}

```

14 Jan 89: Def of `\@endpbox` changed from
`\def\@endpbox{\par\vskip\dp\@arstrutbox\egroup\hfil}`
so vertical spacing works out right if the last line of a ‘p’ entry has a descender.

(End of definition for `\@startpbox` and `\@endpbox`.)

`\@@startpbox`

`\@@endpbox`

```

463 \let\@@startpbox=\@startpbox
464 \let\@@endpbox=\@endpbox

```

(End of definition for `\@@startpbox` and `\@@endpbox`.)

1.3 Hooks for tabulars

Here we declare some hooks that for now are only going to be used in `array` and `longtable`. But the declarations are done in the kernel so that all packages can expect them to be declared and there is no need to check if they are already declared or not depending on package loading.

`tbl/init` (*hook*) Extra initialization at the start of a tabular. It receives the number of columns as its first argument and the full tabular preamble as its second argument so that the code can inspect this data. This hook should not generate any output!

```
465 \NewHookWithArguments{tbl/init}{2}
```

`tbl/row/init` (*hook*) This hook is invoked just before a row is about to start or could start, i.e., just before the first row and then after each row including the last. At this point tagging for the row has not been added and the hook can therefore only contain declarations (which have to be global as everything is executed within a `\noalign`) and it can contain special commands such as `\hline`, `\cline`, or `\rowcolor`, that are allowed after `\`.

```
466 \NewHookWithArguments{tbl/row/init}{1}
```

`tbl/row/begin` (*hook*) Hooks run at the beginning and at the end of each row. The hooks receive the current row number as its argument. This hooks should not generate any output!

In fact, it is not quite clear if they are useful and if we don’t find any good use cases they might vanish in the final release.

```
467 \NewMirroredHookPairWithArguments{tbl/row/begin}{tbl/row/end}{1}
```

`tbl/cell/init` (*hook*) Extra initialization at the start of each cell (horizontal or paragraph) in the tabular, directly in front of the tagging socket setting up cell tagging. It receives the current row and column numbers and the current cell type (`c`, `l`, ...) as arguments. This hook should not generate any output! One important use case for this hook is to manipulate the tagging setup for individual cells.

468 `\NewHookWithArguments{tbl/cell/init}{3}`

`tbl/cell/begin` (*hook*) Hooks run at the beginning and at the end of each cell. The hooks receive the current `tbl/cell/end` (*hook*) row and column number and the cell type as arguments.

469 `\NewMirroredHookPairWithArguments{tbl/cell/begin}{tbl/cell/end}{3}`

470 `\2ekernel`

File 42

ltpictur.dtx

1 Picture Mode

Picture mode commands. In addition to the commands available in L^AT_EX 2.09, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[ $N$ ]( $\langle AX,AY \rangle$ )( $\langle BX,BY \rangle$ )( $\langle CX,CY \rangle$ )` plots a quadratic Bezier curve from ($\langle AX,AY \rangle$) to ($\langle CX,CY \rangle$), with ($\langle BX,BY \rangle$) as the third Bezier point, using $N+1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve—except that at most `\qbeziermax+1` points are drawn. A “point” is a square of side `\@wholewidth`.

`\bezier` In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

<code>\unitlength</code>	= value of dimension argument
<code>\@wholewidth</code>	= current line width
<code>\@halfwidth</code>	= half of current line width
<code>\@linefnt</code>	= font for drawing lines
<code>\@circlefnt</code>	= font for drawing circles

`\linethickness{DIM}` : Sets the width of horizontal and vertical lines in a picture to DIM. Does not change width of slanted lines or circles. Width of all lines reset by `\thinlines` and `\thicklines`

```
\picture(XSIZE,YSIZE)(XORG,YORG)
  BEGIN
    \@picht :=L YSIZE * \unitlength
    box \@picbox :=
      \hb@xt@ XSIZE * \unitlength
      {\hskip -XORG * \unitlength
       \lower YORG * \unitlength
       \hbox{
         \ignorespaces      %% added 13 June 89
       }
    }
  END
```

```
\endpicture ==
  BEGIN
    } \hss }
    height of \@picbox := \@picht
    depth of \@picbox := 0
    \mbox{\box\@picbox}    %% change 26 Aug 91
  END
```

```
\put(X, Y){OBJ} ==
  BEGIN
```

```

\@killglue
\raise Y * \unitlength \hb@xt@ 0pt { \hskip X * \unitlength
                                OBJ \hss }

\ignorespaces
END

\multiput(X,Y)(DELX,DELY){N}{OBJ} ==
BEGIN
\@killglue
\@multicnt := N
\@xdim := X * \unitlength
\@ydim := Y * \unitlength
while \@multicnt > 0
do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
                                OBJ \hss }

\@multicnt := \@multicnt - 1
\@xdim := \@xdim + DELX * \unitlength
\@ydim := \@ydim + DELY * \unitlength
od
\ignorespaces
END

```

`\shortstack[POS]{TEXT}` : Makes a `\vbox` containing TEXT stacked as a one-column array, positioned l, r or c as indicated by POS.

End of historical L^AT_EX 2.09 comments.

The ‘2ekernel’ code ensures that a `\usepackage{autopict}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```
1 <2ekernel>\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

```

\@wholewidth
\@halfwidth
2 <*2ekernel>
3 \newdimen\@wholewidth
4 \newdimen\@halfwidth

(End of definition for \@wholewidth and \@halfwidth.)

```

```

\unitlength
5 \newdimen\unitlength \unitlength =1pt

(End of definition for \unitlength.)

```

```

\@picbox
\@picht
6 \newbox\@picbox
7 \newdimen\@picht

(End of definition for \@picbox and \@picht.)

```

`\@defaultunitsset` Set a length register, #1, accepting number or an etex length expression, #2, with default unit, #3.

The register name in #1 can be prefixed by `\advance` so that the register is incremented by the supplied value.

`\@defaultunitsset{\advance\@vxx}{\textwidth-15pt}\unitlength`

#3 can be a literal unit such as `cm` or a length register such as `\unitlength`.

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\textheight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```

8 \</2ekernel>
9 \<*2ekernel | latexrelease>
10 \<latexrelease>\IncludeInRelease{2020/10/01}%
11 \<latexrelease>{\@defaultunitsset}{default units}%
12 \def\@defaultunitsset#1#2#3{%
13   \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}
14 \</2ekernel | latexrelease>

15 \<latexrelease>\EndIncludeInRelease
16 \<latexrelease>\IncludeInRelease{0000/00/00}%
17 \<latexrelease>{\@defaultunitsset}{default units}%
18 \<latexrelease>\let\@defaultunitsset\undefined
19 \<latexrelease>\EndIncludeInRelease
20 \<*2ekernel>

```

(End of definition for \@defaultunitsset.)

`\picture` #1 should be white space.

#1 should be a ((eating any white space before the bracket),

```

\picture@
21 \</2ekernel>
22 \<*2ekernel | latexrelease>
23 \<latexrelease>\IncludeInRelease{2026/06/01}%
24 \<latexrelease>{\picture@}{optional arg}
25 \newcommand\picture[2] [] {\picture@#2}
26 \</2ekernel | latexrelease>

27 \<latexrelease>\EndIncludeInRelease
28 \<latexrelease>\IncludeInRelease{0000/00/00}%
29 \<latexrelease>{\picture@}{optional arg}%
30 \<latexrelease>\long\def\picture#1{\picture@#1}
31 \<latexrelease>\EndIncludeInRelease
32 \<*2ekernel>
33 \def\picture@(#1){%
34   \ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}

```

(End of definition for \picture and \picture@.)

`\@picture`

```

35 \</2ekernel>
36 \<*2ekernel | latexrelease>
37 \<latexrelease>\IncludeInRelease{2020/10/01}%
38 \<latexrelease>{\@picture}{default units}%
39 \def\@picture(#1,#2)(#3,#4){%
40   \@defaultunitsset\@picht{#2}\unitlength
41   \@defaultunitsset\@tempdimc{#1}\unitlength

```

```

42 \setbox\@picbox\hb@xt@\@tempdimc\bgroup
43 \@defaultunitsset\@tempdimc{#3}\unitlength
44 \hskip -\@tempdimc
45 \@defaultunitsset\@tempdimc{#4}\unitlength
46 \lower\@tempdimc\hbox\bgroup
47 \ignorespaces}
48 </2ekernel | latexrelease>
49 <latexrelease>\EndIncludeInRelease
50 <latexrelease>\IncludeInRelease{0000/00/00}%
51 <latexrelease> {\@picture}{default units}%
52 <latexrelease>\def\@picture(#1,#2)(#3,#4){%
53 <latexrelease> \@picht#2\unitlength
54 <latexrelease> \setbox\@picbox\hb@xt@\#1\unitlength\bgroup
55 <latexrelease> \hskip -#3\unitlength
56 <latexrelease> \lower #4\unitlength\hbox\bgroup
57 <latexrelease> \ignorespaces}
58 <latexrelease>\EndIncludeInRelease
59 <*2ekernel>

```

(End of definition for \@picture.)

\endpicture

```

60 \def\endpicture{%
61 \egroup\hss\egroup
62 \ht\@picbox\@picht\dp\@picbox\z@
63 \mbox{\box\@picbox}}

```

(End of definition for \endpicture.)

In the definitions of \put and \multiput, \hskip was replaced by \kern just in case arg #3 = “plus”. (Bug detected by Don Knuth. changed 20 Jul 87).

```

64 </2ekernel>
65 <*2ekernel | latexrelease>
66 <latexrelease>\IncludeInRelease{2020/10/01}%
67 <latexrelease> {\put}{default units}%
68 <latexrelease>\expandafter\let\csname put \endcsname \@undefined
69 \long\def\put(#1,#2)#3{%
70 \@killglue
71 \@defaultunitsset\@tempdimc{#2}\unitlength
72 \raise\@tempdimc
73 \hb@xt@\z@{%
74 \@defaultunitsset\@tempdimc{#1}\unitlength
75 \kern\@tempdimc
76 #3\hss}%
77 \ignorespaces}
78 </2ekernel | latexrelease>
79 <latexrelease>\EndIncludeInRelease
80 <latexrelease>\IncludeInRelease{0000/00/00}%
81 <latexrelease> {\put}{default units}%
82 <latexrelease>\expandafter\let\csname put \endcsname \@undefined
83 <latexrelease>\long\def\put(#1,#2)#3{%
84 <latexrelease> \@killglue\raise#2\unitlength
85 <latexrelease> \hb@xt@\z@{\kern#1\unitlength #3\hss}%

```

```

86 <latexrelease> \ignorespaces}
87 <latexrelease>\EndIncludeInRelease
88 <*2ekernel>

\multiput #3 had better be a (.
89 </2ekernel>
90 <*2ekernel | latexrelease>
91 <latexrelease>\IncludeInRelease{2020/10/01}%
92 <latexrelease> \multiput{default units}%
93 <latexrelease>\expandafter\let\csname multiput \endcsname\@undefined
94 \def\multiput(#1,#2)#3{%
95   \@defaultunitsset\@xdim{#1}\unitlength
96   \@defaultunitsset\@ydim{#2}\unitlength
97   \@multiput{ }
98 </2ekernel | latexrelease>

99 <latexrelease>\EndIncludeInRelease
100 <latexrelease>\IncludeInRelease{0000/00/00}%
101 <latexrelease> \multiput{default units}%
102 <latexrelease>\expandafter\let\csname multiput \endcsname\@undefined
103 <latexrelease>\def\multiput(#1,#2)#3{%
104 <latexrelease> \@xdim #1\unitlength
105 <latexrelease> \@ydim #2\unitlength
106 <latexrelease> \@multiput{ }
107 <latexrelease>\EndIncludeInRelease
108 <*2ekernel>

(End of definition for \multiput.)

\@multiput
109 </2ekernel>
110 <*2ekernel | latexrelease>
111 <latexrelease>\IncludeInRelease{2020/10/01}%
112 <latexrelease> \@multiput{default units}%
113 \long\def\@multiput(#1,#2)#3#4{%
114   \@killglue\@multicnt #3\relax
115   \@whilenum \@multicnt >\z@ \do
116     {\raise\@ydim\hb@xt@ \z@{\kern\@xdim #4\hss}%
117     \advance\@multicnt\m@ne
118     \@defaultunitsset{\advance\@xdim}{#1}\unitlength
119     \@defaultunitsset{\advance\@ydim}{#2}\unitlength}%
120   \ignorespaces}
121 </2ekernel | latexrelease>

122 <latexrelease>\EndIncludeInRelease
123 <latexrelease>\IncludeInRelease{0000/00/00}%
124 <latexrelease> \@multiput{default units}%
125 <latexrelease>\long\def\@multiput(#1,#2)#3#4{%
126 <latexrelease> \killglue\@multicnt #3\relax
127 <latexrelease> \whilenum \@multicnt >\z@ \do
128 <latexrelease>   {\raise\@ydim\hb@xt@ \z@{\kern\@xdim #4\hss}%
129 <latexrelease>   \advance\@multicnt\m@ne
130 <latexrelease>   \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
131 <latexrelease> \ignorespaces}
132 <latexrelease>\EndIncludeInRelease
133 <*2ekernel>

```

```

(End of definition for \@multiput.)

\@killglue
134 \def\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
(End of definition for \@killglue.)

\thinlines
\thicklines
135 \DeclareRobustCommand\thinlines{\let\@linefnt\tenln
136 \let\@circlefnt\tencirc
137 \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
138 \DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw
139 \let\@circlefnt\tencircw
140 \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
(End of definition for \thinlines and \thicklines.)

\linethickness
141 \DeclareRobustCommand*\linethickness[1]
142 {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}
(End of definition for \linethickness.)

\ishortstack
143 \def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
(End of definition for \ishortstack.)

\@ishortstack
144 \def\@shortstack[#1]{%
145 \leavevmode
146 \vbox\bgroup
147 \baselineskip-\p@\lineskip 3\p@
148 \let\mb@l\hss\let\mb@r\hss
149 \expandafter\let\csname mb@#1\endcsname\relax
150 \let\\ \@stackcr
151 \@ishortstack}
(End of definition for \@ishortstack.)

\@ishortstack
152 \def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\cr}\egroup}
(End of definition for \@ishortstack.)

\@stackcr
\@ixstackcr
153 \protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
154 \def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}
(End of definition for \@stackcr and \@ixstackcr.)

```

\@istackcr

```

155 </2ekernel>
156 <*2ekernel | latexrelease>
157 <latexrelease>\IncludeInRelease{2020/10/01}%
158 <latexrelease>                {\@istackcr}{\shortstack calc support}%
159 \def\@istackcr[#1]{\cr\noalign{\@vspace@calcify{#1}}\ignorespaces}
160 </2ekernel | latexrelease>

161 <latexrelease>\EndIncludeInRelease
162 <latexrelease>\IncludeInRelease{0000/00/00}%
163 <latexrelease>                {\@istackcr}{\shortstack calc support}%
164 <latexrelease>
165 <latexrelease>\def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
166 <latexrelease>\EndIncludeInRelease
167 <*2ekernel>

(End of definition for \@istackcr.)
Historical LATEX 2.09 comments (not necessarily accurate any more):
\line(X,Y){LEN} ==
BEGIN
  \@xarg      := X
  \@yarg      := Y
  \@linelen := LEN * \unitlength
  if \@xarg = 0
    then \vline
  else if \@yarg = 0
    then \hline
    else \sline
      if
        if
          END

\sline ==
BEGIN
  if \@xarg < 0
    then @negarg := T
        \@xarg := -\@xarg
        \@yyarg := -\@yarg
    else @negarg := F
        \@yyarg := \@yarg
  fi
  \@tempcnta := |\@yyarg|
  if \@tempcnta > 6
    then error: 'LATEX ERROR: Illegal \line or \vector argument.'
        \@tempcnta := 0
  fi
  \box\@linechar := \hbox{\@linefont \@getlinechar(\@xarg,\@yyarg) }
  if \@yarg > 0 then \@upordown = \raise
                    \@clnht := 0
                    else \@upordown = \lower
                    \@clnht := height of \box\@linechar

```

```

fi
\@clnwd := width of \box\@linechar
if @negarg
then \hskip - width of \box\@linechar
\reserved@a == \hskip - 2* width of box \@linechar
else \reserved@a == \relax
fi
%% Put out integral number of line segments
while \@clnwd < \@linelen
do \upordown \@clnht \copy\@linechar
\reserved@a
\@clnht := \@clnht + ht of \box\@linechar
\@clnwd := \@clnwd + width of \box\@linechar
od

%% Put out last segment
\@clnht := \@clnht - height of \box\@linechar
\@clnwd := \@clnwd - width of \box\@linechar
\@tempdima := \@linelen - \@clnwd
\@tempdimb := \@tempdima - width of \box\@linechar
if @negarg then \hskip -\@tempdimb
else \hskip \@tempdimb
fi
\@tempdima := 1000 * \@tempdima
\@tempcnta := \@tempdima / width of \box\@linechar
\@tempdima := (\@tempcnta * ht of \box\@linechar)/1000
\@clnht := \@clnht + \@tempdima
if \@linelen < width of box\@linechar
then \hskip width of box\@linechar
else \hbox{\upordown \@clnht \copy\@linechar}
fi
END

\@hline ==
BEGIN
if \@xarg < 0 then \hskip -\@linelen \fi
\vrule height \@halfwidth depth \@halfwidth width \@linelen
if \@xarg < 0 then \hskip -\@linelen \fi
END

\@vline == if \@yarg < 0 \@downline else \@upline fi

\@getlinechar(X,Y) ==
BEGIN
\@tempcnta := 8*X - 9
if Y > 0
then \@tempcnta := \@tempcnta + Y
else \@tempcnta := \@tempcnta - Y + 64
fi

```



```

        \char\@tempcnta
    END

\vector(X,Y){LEN} ==
BEGIN
    \@xarg    := X
    \@yarg    := Y
    \@linelen := LEN * \unitlength
    if \@xarg = 0
        then \@vvector
        else if \@yarg = 0
            then \@hvector
            else \@svector
        if
    if
END

\@hvector ==
BEGIN
    \@hline
    {\@linefmt if \@xarg < 0 then \@getlarrow(1,0)
                                     else \@getrarrow(1,0)
    fi}
END

\@vvector == if \@yarg < 0 \@downvector else \@upvector fi

\@svector ==
BEGIN
    \@sline
    \@tempcnta := |\@yarg|
    if \@tempcnta < 5
        then \hskip - width of \box\@linechar
            \@upordown \@clnht \hbox
            {\@linefmt
            if \@negarg then \@getlarrow(\@xarg,\@yyarg)
                        else \@getrarrow(\@xarg,\@yyarg)
            fi }
        else error: 'LATEX ERROR: Illegal \line or \vector argument.'
    fi
END

\@getlarrow(X,Y) ==
BEGIN
    if Y = 0
        then \@tempcnta := '33
        else \@tempcnta := 16 * X - 9
            \@tempcntb := 2 * Y
            if \@tempcntb > 0
                then \@tempcnta := \@tempcnta + \@tempcntb

```

```

        else \@tempcnta := \@tempcnta - \@tempcntb + 64
      fi
    fi
    \char\@tempcnta
  END

\@gettrarrow(X,Y) ==
BEGIN
  \@tempcntb := |Y|
  case of \@tempcntb
    0 : \@tempcnta := '55
    1 : if X < 3
        then \@tempcnta := 24*X - 6
        else if X = 3
            then \@tempcnta := 49
            else \@tempcnta := 58 fi
        fi
    2 : if X < 3
        then \@tempcnta := 24*X - 3
        else \@tempcnta := 51 % X must = 3
        fi
    3 : \@tempcnta := 16*X - 2
    4 : \@tempcnta := 16*X + 7
  endcase
  if Y < 0
    then \@tempcnta := \@tempcnta + 64
  fi
  \char\@tempcnta
END

```

End of historical L^AT_EX 2.09 comments.

\if@negarg

168 \newif\if@negarg

(End of definition for \if@negarg.)

\line

```

169 </2ekernel>
170 <*2ekernel|latexrelease>
171 <latexrelease>\IncludeInRelease{2020/10/01}%
172 <latexrelease>          {\line}{default units}%
173 <latexrelease>\expandafter\let\csname line \endcsname\@undefined
174 \def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
175   \@defaultunitsset\@linelen{#3}\unitlength
176   \ifdim\@linelen<\z@\@badlinearg\else
177     \ifnum\@xarg =\z@ \@vline
178     \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
179   \fi
180   \fi}
181 </2ekernel|latexrelease>

```

```

182 <latexrelease>\EndIncludeInRelease
183 <latexrelease>\IncludeInRelease{0000/00/00}%
184 <latexrelease>          {\line}{default units}%
185 <latexrelease>\expandafter\let\csname line \endcsname\@undefined
186 <latexrelease>\def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
187 <latexrelease>  \@linelen #3\unitlength
188 <latexrelease>  \ifdim\@linelen<\z@\@badlinearg\else
189 <latexrelease>    \ifnum\@xarg =\z@ \@vline
190 <latexrelease>      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
191 <latexrelease>    \fi
192 <latexrelease>  \fi}
193 <latexrelease>\EndIncludeInRelease
194 <*2ekernel>

```

(End of definition for \line.)

\@sline

```

195 \def\@sline{%
196   \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
197   \else \@negargfalse \@yyarg \@yarg \fi
198   \ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
199   \ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
200   \ifnum\@xarg>6 \@badlinearg\@xarg \@one \fi
201   \setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%

```

If we have something like \line(5,5){30} the \@linechar will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```

202   \ifdim\wd\@linechar=\z@
203     \setbox\@linechar\hbox{.}%
204     \@badlinearg
205   \fi
206   \ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
207   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
208   \@clnwd \wd\@linechar
209   \if@negarg
210     \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
211   \else
212     \let\reserved@a\relax
213   \fi
214   \@whiledim \@clnwd <\@linelen \do
215     {\@upordown\@clnht\copy\@linechar
216     \reserved@a
217     \advance\@clnht \ht\@linechar
218     \advance\@clnwd \wd\@linechar}%
219   \advance\@clnht -\ht\@linechar
220   \advance\@clnwd -\wd\@linechar
221   \@tempdima\@linelen\advance\@tempdima -\@clnwd
222   \@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
223   \if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
224   \multiply\@tempdima \@m
225   \@tempcnta \@tempdima
226   \@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
227   \@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
228   \divide\@tempdima \@m

```

```

229 \advance\@clnht \@tempdima
230 \ifdim \@linelen <\wd\@linechar
231 \hskip \wd\@linechar

```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```

232 \ifdim \@linelen = \z@
233 \else
234 \@picture@warn
235 \fi
236 \else\@upordown\@clnht\copy\@linechar\fi}

```

(End of definition for \@sline.)

\@hline

```

237 \def\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
238 \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
239 \ifnum \@xarg <\z@ \hskip -\@linelen \fi}

```

(End of definition for \@hline.)

\@getlinechar

```

240 \def\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
241 \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
242 \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
243 \char\@tempcnta}

```

(End of definition for \@getlinechar.)

\vector

```

244 \</2ekernel>
245 \< *2ekernel | latexrelease>
246 \< latexrelease>\IncludeInRelease{2020/10/01}%
247 \< latexrelease> \{ \vector\} {default units}%
248 \< latexrelease>\expandafter\let\csname vector \endcsname\@undefined
249 \def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
250 \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
251 \ifnum\@tempcnta<5\relax
252 \@defaultunitsset\@linelen{#3}\unitlength
253 \ifdim\@linelen<\z@\@badlinearg\else
254 \ifnum\@xarg =\z@ \@vvector
255 \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
256 \fi
257 \fi
258 \else\@badlinearg\fi}
259 \</2ekernel | latexrelease>

260 \< latexrelease>\EndIncludeInRelease
261 \< latexrelease>\IncludeInRelease{0000/00/00}%
262 \< latexrelease> \{ \vector\} {default units}%
263 \< latexrelease>\expandafter\let\csname vector \endcsname\@undefined
264 \< latexrelease>\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
265 \< latexrelease> \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
266 \< latexrelease> \ifnum\@tempcnta<5\relax
267 \< latexrelease> \@linelen #3\unitlength
268 \< latexrelease> \ifdim\@linelen<\z@\@badlinearg\else

```

```

269 \latexrelease \ifnum \@xarg =\z@ \@vvector
270 \latexrelease \else \ifnum \@yarg =\z@ \@hvector \else \@svector\fi
271 \latexrelease \fi
272 \latexrelease \fi
273 \latexrelease \else \@badlinearg\fi}
274 \latexrelease \EndIncludeInRelease
275 \*2ekernel)

```

(End of definition for \vector.)

\@hvector

```

276 \def\@hvector{\@hline\hb@xt@\z@{\@linefnt
277 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
278 \hss\@getrarrow(1,0)\fi}}

```

(End of definition for \@hvector.)

\@vvector

```

279 \def\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}

```

(End of definition for \@vvector.)

\@svector

```

280 \def\@svector{\@sline
281 \@tempcnta\@yarg \ifnum \@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
282 \ifnum \@tempcnta <5%
283 \hskip -\wd\@linechar
284 \@upordown\@clnht \hbox{\@linefnt \if@negarg
285 \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
286 \else \@badlinearg\fi}

```

(End of definition for \@svector.)

\@getlarrow

```

287 \def\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
288 \else
289 \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
290 \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
291 \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
292 \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
293 \fi\fi\char\@tempcnta}

```

(End of definition for \@getlarrow.)

\@getrarrow

```

294 \def\@getrarrow(#1,#2){\@tempcntb #2\relax
295 \ifnum \@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
296 \ifcase \@tempcntb\relax \@tempcnta 45 % '55
297 \or
298 \ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
299 24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
300 \else\@tempcnta 58 \fi\fi\or
301 \ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
302 24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
303 \@tempcnta #1\relax\multiply\@tempcnta

```

```

304 \sixt@@n \advance\@tempcnta -\tw@ \else
305 \@tempcnta #1\relax\multiply\@tempcnta
306 \sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
307 \char\@tempcnta}

```

(End of definition for \gettrarrow.)

\@vline

```

308 \def\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}

```

(End of definition for \@vline.)

\@upline

```

309 \def\@upline{%
310   \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
311     \@height \@linelen \@depth \z@\hss}}

```

(End of definition for \@upline.)

\@downline

```

312 \def\@downline{%
313   \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
314     \@height \z@ \@depth \@linelen \hss}}

```

(End of definition for \@downline.)

\@upvector

```

315 \def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}% '66
316   \raise \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}

```

(End of definition for \@upvector.)

\@downvector

```

317 \def\@downvector{\@downline\lower \@linelen
318   \hb@xt@\z@{\@linefnt\char 63 % '77
319   \hss}}

```

(End of definition for \@downvector.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

\dashbox{D}(X,Y) ==
BEGIN
leave vertical mode
\hb@xt@ 0pt {
  \baselineskip := 0pt
  \lineskip      := 0pt
%% HORIZONTAL DASHES
  \@dashdim := X * \unitlength
  \@dashcnt := \@dashdim + 200 % to prevent roundoff error
  \@dashdim := D * \unitlength
  \@dashcnt := \@dashcnt / \@dashdim
  if \@dashcnt is odd
  then \@dashdim := 0pt
       \@dashcnt := (\@dashcnt + 1) / 2
  else \@dashdim := \@dashdim / 2

```

```

\@dashcnt := \@dashcnt / 2 - 1
\box\@dashbox := \hbox{\vrule height \@halfwidth
depth \@halfwidth width \@dashdim}
\put(0,0){\copy\@dashbox}
\put(0,Y){\copy\@dashbox}
\put(X,0){\hskip -\@dashdim\copy\@dashbox}
\put(X,Y){\hskip -\@dashdim\box\@dashbox}
\@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule height \@halfwidth
depth \@halfwidth width D * \unitlength
\hskip D * \unitlength}

\@tempcnta := 0
\put(0,0){\hskip \@dashdim
while \@tempcnta < \@dashcnt
do \copy\@dashbox
\@tempcnta := \@tempcnta + 1
od
}
\@tempcnta := 0
\put(0,Y){\hskip \@dashdim
while \@tempcnta < \@dashcnt
do \copy\@dashbox
\@tempcnta := \@tempcnta + 1
od
}

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
then \@dashdim := 0pt
\@dashcnt := (\@dashcnt + 1) / 2
else \@dashdim := \@dashdim / 2
\@dashcnt := \@dashcnt / 2 - 1
\box\@dashbox := \hbox{\hskip -\@halfwidth
\vrule width \@wholewidth
height \@dashdim }

\put(0,0){\copy\@dashbox}
\put(X,0){\copy\@dashbox}
\put(0,Y){\lower\@dashdim\copy\@dashbox}
\put(X,Y){\lower\@dashdim\copy\@dashbox}
\@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
height D * \unitlength }

\@tempcnta := 0
\put(0,0){\hskip -\halfwidth

```

```

        \vbox{while \@tempcnta < \@dashcnt
            do \vskip D*\unitlength
              \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
            od
            \vskip \@dashdim
          } }
    \@tempcnta := 0
    put(X,0){\hskip -\halfwidth
      \vbox{while \@tempcnta < \@dashcnt
          do \vskip D*\unitlength
            \copy\@dashbox
            \@tempcnta := \@tempcnta + 1
          od
          \vskip \@dashdim
        }
      }
  }      % END DASHES

  \@imakepicbox(X,Y)
END
End of historical LATEX 2.09 comments.

```

\dashbox

```

320 </2ekernel>
321 <*2ekernel | latexrelease>
322 <[latexrelease]\IncludeInRelease{2020/10/01}%
323 <[latexrelease]                {\dashbox}{default units}%
324 <[latexrelease]\expandafter\let\csname dashbox \endcsname\@undefined
325 \def\dashbox#1(#2,#3){\leavevmode\hb@xt@ \z@{\baselineskip \z@skip
326 \lineskip \z@skip
327 \@defaultunitsset\@dashdim{#2}\unitlength
328 \@dashcnt \@dashdim \advance\@dashcnt 200
329 \@defaultunitsset\@dashdim{#1}\unitlength
330 \divide\@dashcnt \@dashdim
331 \ifodd\@dashcnt\@dashdim \z@
332 \advance\@dashcnt \@one \divide\@dashcnt \tw@
333 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
334 \advance\@dashcnt \m@ne
335 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
336 \@width \@dashdim}\put(0,0){\copy\@dashbox}%
337 \put(0,#3){\copy\@dashbox}%
338 \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
339 \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
340 \multiply\@dashdim \thr@@
341 \fi
342 \setbox\@dashbox \hbox{%
343   \@defaultunitsset\@tempdimc{#1}\unitlength
344   \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@tempdimc
345   \hskip\@tempdimc}%
346 \@tempcnta\z@
347 \put(0,0){\hskip\@dashdim \@whilenum \@tempcnta < \@dashcnt

```



```

348 \do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
349 \put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
350 \do{\copy\@dashbox\advance\@tempcnta \@ne }}%
351 \@defaultunitsset\@dashdim{#3}\unitlength
352 \@dashcnt \@dashdim \advance\@dashcnt 200
353 \@defaultunitsset\@dashdim{#1}\unitlength
354 \divide\@dashcnt \@dashdim
355 \ifodd\@dashcnt \@dashdim \z@
356 \advance\@dashcnt \@ne \divide\@dashcnt \tw@
357 \else
358 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
359 \advance\@dashcnt \m@ne
360 \setbox\@dashbox\hbox{\hskip -\@halfwidth
361 \vrule \@width \@wholewidth
362 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
363 \put(#2,0){\copy\@dashbox}%
364 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
365 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
366 \multiply\@dashdim \thr@@
367 \fi
368 \@defaultunitsset\@tempdimb{#1}\unitlength
369 \setbox\@dashbox\hbox{%
370 \vrule \@width \@wholewidth \@height\@tempdimb}%
371 \@tempcnta\z@
372 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
373 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }}%
374 \vskip\@dashdim}}\@tempcnta\z@
375 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
376 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }}%
377 \vskip\@dashdim}}\@makepicbox(#2,#3)}
378 /2ekernel|latexrelease)

379 \<latexrelease>\EndIncludeInRelease
380 \<latexrelease>\IncludeInRelease{0000/00/00}%
381 \<latexrelease> \<\dashbox>\{default units}%
382 \<latexrelease>\expandafter\let\csname dashbox \endcsname\@undefind
383 \<latexrelease>\def\dashbox#1(#2,#3){%
384 \<latexrelease>\leavevmode\hb@xt@\z@{\baselineskip \z@skip
385 \<latexrelease>\lineskip \z@skip
386 \<latexrelease>\@dashdim #2\unitlength
387 \<latexrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
388 \<latexrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
389 \<latexrelease>\ifodd\@dashcnt\@dashdim \z@
390 \<latexrelease>\advance\@dashcnt \@ne \divide\@dashcnt \tw@
391 \<latexrelease>\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
392 \<latexrelease>\advance\@dashcnt \m@ne
393 \<latexrelease>\setbox\@dashbox \hbox{%
394 \<latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
395 \<latexrelease> \@width \@dashdim}\put(0,0){\copy\@dashbox}%
396 \<latexrelease>\put(0,#3){\copy\@dashbox}%
397 \<latexrelease>\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
398 \<latexrelease>\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
399 \<latexrelease>\multiply\@dashdim \thr@@
400 \<latexrelease>\fi
401 \<latexrelease>\setbox\@dashbox \hbox{%

```

```

402 <latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
403 <latexrelease> \@width #1\unitlength\hskip #1\unitlength\@tempcnta\z@
404 <latexrelease>\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
405 <latexrelease>\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
406 <latexrelease>\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
407 <latexrelease>\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
408 <latexrelease>\@dashdim #3\unitlength
409 <latexrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
410 <latexrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
411 <latexrelease>\ifodd\@dashcnt \@dashdim \z@
412 <latexrelease>\advance\@dashcnt \@ne \divide\@dashcnt \tw@
413 <latexrelease>\else
414 <latexrelease>\divide\@dashdim \tw@ \divide\@dashcnt \tw@
415 <latexrelease>\advance\@dashcnt \m@ne
416 <latexrelease>\setbox\@dashbox\hbox{\hskip -\@halfwidth
417 <latexrelease>\vrule \@width \@wholewidth
418 <latexrelease>\@height \@dashdim}\put(0,0){\copy\@dashbox}%
419 <latexrelease>\put(#2,0){\copy\@dashbox}%
420 <latexrelease>\put(0,#3){\lower\@dashdim\copy\@dashbox}%
421 <latexrelease>\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
422 <latexrelease>\multiply\@dashdim \thr@@
423 <latexrelease>\fi
424 <latexrelease>\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
425 <latexrelease>\@height #1\unitlength\@tempcnta\z@
426 <latexrelease>\put(0,0){%
427 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
428 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
429 <latexrelease> \advance\@tempcnta \@ne }}%
430 <latexrelease> \vskip\@dashdim}}\@tempcnta\z@
431 <latexrelease>\put(#2,0){%
432 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
433 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
434 <latexrelease> \advance\@tempcnta \@ne }}%
435 <latexrelease> \vskip\@dashdim}}\@makepicbox(#2,#3)}
436 <latexrelease>\EndIncludeInRelease
437 <*2ekernel>

```

(End of definition for \dashbox.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to `D * \unitlength`. `\put(X,Y){\circle{D}}` puts the circle with its center at (X,Y).

`\oval(X,Y)` : Makes an oval as round as possible that fits in the rectangle of width `X * \unitlength` and height `Y * \unitlength`. The reference point is the center.

`\oval(X,Y)[POS]` : Same as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS.

E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical rule. The width of the box will be `\@tempdima`. DELTA1 and DELTA2 are added to the character number in `\@tempcnta` to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the top or the bottom of the oval being constructed. The baseline will coincide with bottom edge of the rule; the left side of the box will coincide with the left side of the oval. The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number of the top-right quarter circle with the largest diameter less than or equal to DIAM. Sets `\@tempboxa` to an hbox containing that character. Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance from the circle's left outside edge to its right inside edge. (These characters are like those described in the TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
BEGIN
  \@tempcnta      := integer coercion of (DIAM + 2pt)
                                     + 2pt added 1 Nov 88
  \@tempcnta      := \@tempcnta / integer coercion of 4pt
  if \@tempcnta > 10
    then \@tempcnta := 10 fi
  if \@tempcnta > 0
    then \@tempcnta := \@tempcnta-1
    else LaTeX Warning: Oval too small.
  fi
  \@tempcnta      := 4 * \@tempcnta
  \@tempboxa      := \hbox{\@circlefnt \char \@tempcnta}
  \@tempdima      := \wd \@tempboxa
END
```

```
\@put{X}{Y}{OBJ} ==
BEGIN
  \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
END
```

```

\@oval(X,Y)[POS] ==
BEGIN
  \begingroup
    \boxmaxdepth := \maxdimen
    @ovt := @ovb := @ovl := @ovr := true
    for all E in POS
      do @ovE := false od
    \@ovxx      := X * \unitlength
    \@ovyy      := Y * \unitlength
    \@tempdimb := min(\@ovxx,\@ovyy)
    \getcirc{\@tempdimb-2pt} %% "-2pt" added 7 Dec 89
    \@ovro      := \ht \@tempboxa
    \@ovri      := \dp \@tempboxa
    \@ovdx      := \@ovxx - \@tempdima
    \@ovdx      := \@ovdx/2
    \@ovdy      := \@ovyy - \@tempdima
    \@ovdy      := \@ovdy/2
    \@circlefnt
    \@tempboxa :=
      \hbox{
        if @ovr
          then \@ovvert{3}{2} \kern -\@tempdima
        fi
        if @ovl
          then \kern \@ovxx \@ovvert{0}{1} \kern -\@tempdima
            \kern -\@ovxx
          fi
        if @ovt
          then \@ovhorz \kern -\@ovxx
        fi
        if @ovb
          then \raise \@ovyy \@ovhorz
        fi
      }
    \@ovdx := \@ovdx + \@ovro
    \@ovdy := \@ovdy + \@ovro
    \ht\@tempboxa := \dp\@tempboxa := 0
    \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
  \endgroup
END

\@ovvert {DELTA1} {DELTA2} ==
BEGIN
  \vbox to \@ovyy {
    if @ovb
      then \@tempcntb := \@tempcnta + DELTA1
        \kern -\@ovro
        \hbox { \char \@tempcntb }
        \nointerlineskip
      else \kern \@ovri \kern \@ovdy
    fi
  }

```

```

        fi
        \leaders \vrule width \@wholewidth \vfil
        \nointerlineskip
        if @ovt
            then \@tempcntb := \@tempcnta + DELTA2
                \hbox { \char \@tempcntb }
            else \kern \@ovdy \kern \@ovro
        fi
    }
END

\@ovhorz ==
BEGIN
    \hb@xt@ \@ovxxx{
        \kern \@ovro
        if @ovr
            then
            else \kern \@ovdx
        fi
        \leaders \hrule height \@wholewidth \hfil
        if @ovl
            then
            else \kern \@ovdx
        fi
        \kern \@ovri
    }
END

\circle{DIAM} ==
BEGIN
    \begingroup
    \boxmaxdepth := maxdimen
    \@tempdimb := DIAM *\unitlength
    if \@tempdimb > 15.5pt
        then \@getcirc{\@tempdimb}
            \@ovro := \ht \@tempboxa
            \@tempboxa := \hbox{
                \@circlefnt
                \@tempcnta := \@tempcnta + 2
                \char \@tempcnta
                \@tempcnta := \@tempcnta - 1
                \char \@tempcnta
                \kern -2\@tempdima
                \@tempcnta := \@tempcnta + 2
                \raise \@tempdima \hbox { \char \@tempcnta }
                \raise \@tempdima \box\@tempboxa
            }
            \ht\@tempboxa := \dp\@tempboxa := 0
            \@put{-\@ovro}{-\@ovro}{\@tempboxa}
        else

```

```

\@circ{\@tempdimb}{96}
fi
\endgroup
END

\circle*{DIAM} == \@dot{DIAM} == \@circ{DIAM*\unitlength}{112}

\@circ{DIAM}{CHAR} ==
BEGIN
\@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
if \@tempcnta > 15 then \@tempcnta := 15 fi
if \@tempcnta > 1 then \@tempcnta := \@tempcnta - 1 fi
\@tempcnta := \@tempcnta + CHAR
\@circlefnt
\char \@tempcnta
END
End of historical LATEX 2.09 comments.

\if@ovt If producing the Top Bottom Left or Right of an oval.
\if@ovb 438 \newif\if@ovt
\if@ovl 439 \newif\if@ovb
\if@ovr 440 \newif\if@ovl
441 \newif\if@ovr

(End of definition for \if@ovt and others.)

\@ovxx
\@ovyy 442 \newdimen\@ovxx
\@ovdx 443 \newdimen\@ovyy
\@ovdy 444 \newdimen\@ovdx
\@ovro 445 \newdimen\@ovdy
\@ovri 446 \newdimen\@ovro
447 \newdimen\@ovri

(End of definition for \@ovxx and others.)

\advance\@tempdima 2pt\relax added 1 Nov 88 to fix bug in which size of drawn
circle not monotonic function of argument of \circle, caused by different rounding for
dimensions of large and small circles.

\@getcirc
448 \def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
449 \@tempcnta\@tempdima
450 \@tempdima 4\p@ \divide\@tempcnta\@tempdima
451 \ifnum \@tempcnta >10\relax
452 \@picture@warn
453 \@tempcnta 10\relax
454 \fi
455 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
Warn if requirements for oval or circle can't be met.
456 \else \@picture@warn \fi
457 \multiply\@tempcnta 4\relax
458 \setbox \@tempboxa \hbox{\@circlefnt
459 \char \@tempcnta}\@tempdima \wd \@tempboxa}

```

(End of definition for \@getcirc.)

\@picture@warn Generic warning for lines, vectors (used in \@sline) and oval or circle (used in \@getcirc) are not available at right size.

```
460 \def\@picture@warn{\@latex@warning{%  
461     \string\oval, \string\circle, or \string\line\space  
462     size unavailable}}
```

(End of definition for \@picture@warn.)

\@put

```
463 \def\@put#1#2#3{\raise #2\hb@xt@{#3}{\hskip #1#3\hss}}
```

(End of definition for \@put.)

\oval

```
464 \def\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2) []}}
```

(End of definition for \oval.)

```
465 </2ekernel>  
466 <latexrelease>\IncludeInRelease{2016/03/31}%  
467 <latexrelease>                {\@ovhlinetrue}%  
468 <latexrelease>                {Avoid almost zero length leaders}%  
469 <*2ekernel | latexrelease>
```

\if@ovvline Tests whether horizontal or vertical lines are needed.

```
\if@ovhline 470 \newif\if@ovvline \@ovvlinetrue  
471 \newif\if@ovhline \@ovhlinetrue  
472 </2ekernel | latexrelease>  
473 <latexrelease>\EndIncludeInRelease  
474 <latexrelease>\IncludeInRelease{0000/00/00}%  
475 <latexrelease>                {\@ovhlinetrue}%  
476 <latexrelease>                {Avoid almost zero length leaders}%  
477 <latexrelease>\let\if@ovvline\@undefined  
478 <latexrelease>\let\if@ovhline\@undefined  
479 <latexrelease>\EndIncludeInRelease  
480 <*2ekernel>
```

(End of definition for \if@ovvline and \if@ovhline.)

\@oval

```
481 </2ekernel>  
482 <*2ekernel | latexrelease>  
483 <latexrelease>\IncludeInRelease{2020/10/01}%  
484 <latexrelease>                {\@oval}{default units}%  
485 \def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen  
486     \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue  
  
487     \@ovvlinefalse \@ovhlinefalse  
  
488     \@tfor\reserved@a :=#3\do{%  
489         \csname @ov\reserved@a false\endcsname}%  
490     \@defaultunitsset\@ovxx{#1}\unitlength  
491     \@defaultunitsset\@ovyy{#2}\unitlength
```

```

492 \tempdimb \ifdim \ovyy >\ovxx \ovxx \ovvlinetrue
493 \else \ovyy \ifdim \ovyy =\ovxx \else \ovhlinetrue \fi\fi
494 \advance \tempdimb -2\p@
495 \getcirc \tempdimb
496 \ovro \ht\tempboxa \ovri \dp\tempboxa
497 \ovdx\ovxx \advance\ovdx -\tempdima \divide\ovdx \tw@
498 \ovdy\ovyy \advance\ovdy -\tempdima \divide\ovdy \tw@

499 \ifdim \ovdx >\z@ \ovhlinetrue \fi
500 \ifdim \ovdy >\z@ \ovvlinetrue \fi

501 \circlefnt \setbox\tempboxa
502 \hbox{\if@ovr \ovvert32\kern -\tempdima \fi
503 \if@ovl \kern \ovxx \ovvert01\kern -\tempdima \kern -\ovxx \fi
504 \if@ovt \ovhorz \kern -\ovxx \fi
505 \if@ovb \raise \ovyy \ovhorz \fi}\advance\ovdx\ovro
506 \advance\ovdy\ovro \ht\tempboxa\z@ \dp\tempboxa\z@
507 \put{-\ovdx}{-\ovdy}{\box\tempboxa}%
508 \endgroup}
509 </2ekernel|latexrelease>

510 <latexrelease>\EndIncludeInRelease
511 <latexrelease>\IncludeInRelease{2016/03/31}%
512 <latexrelease> {\oval}{default units}%
513 <latexrelease>\def\oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
514 <latexrelease> \ovttrue \ovbtrue \ovltrue \ovrtrue
515 <latexrelease> \ovvlinefalse \ovhlinefalse
516 <latexrelease> \tfor\reserved@a :=#3\do{%
517 <latexrelease> \csname @ov\reserved@a false\endcsname}%
518 <latexrelease> \ovxx #1\unitlength
519 <latexrelease> \ovyy #2\unitlength
520 <latexrelease> \tempdimb \ifdim \ovyy >\ovxx \ovxx \ovvlinetrue
521 <latexrelease> \else \ovyy \ifdim \ovyy =\ovxx \else \ovhlinetrue
522 <latexrelease> \fi\fi
523 <latexrelease> \advance \tempdimb -2\p@
524 <latexrelease> \getcirc \tempdimb
525 <latexrelease> \ovro \ht\tempboxa \ovri \dp\tempboxa
526 <latexrelease> \ovdx\ovxx \advance\ovdx -\tempdima \divide\ovdx \tw@
527 <latexrelease> \ovdy\ovyy \advance\ovdy -\tempdima \divide\ovdy \tw@
528 <latexrelease> \ifdim \ovdx >\z@ \ovhlinetrue \fi
529 <latexrelease> \ifdim \ovdy >\z@ \ovvlinetrue \fi
530 <latexrelease> \circlefnt \setbox\tempboxa
531 <latexrelease> \hbox{\if@ovr \ovvert32\kern -\tempdima \fi
532 <latexrelease> \if@ovl
533 <latexrelease> \kern \ovxx \ovvert01\kern -\tempdima \kern -\ovxx
534 <latexrelease> \fi
535 <latexrelease> \if@ovt \ovhorz \kern -\ovxx \fi
536 <latexrelease> \if@ovb \raise \ovyy \ovhorz \fi}\advance\ovdx\ovro
537 <latexrelease> \advance\ovdy\ovro \ht\tempboxa\z@ \dp\tempboxa\z@
538 <latexrelease> \put{-\ovdx}{-\ovdy}{\box\tempboxa}%
539 <latexrelease> \endgroup}
540 <latexrelease>\EndIncludeInRelease

541 <latexrelease>\IncludeInRelease{0000/00/00}%
542 <latexrelease> {\oval}{default units}%
543 <latexrelease>\def\oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen

```



```

544 <latexrelease> \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
545 <latexrelease> \tfor\reserved@a :=#3\do
546 <latexrelease> {\csname @ov\reserved@a false\endcsname}%
547 <latexrelease> \@ovxx #1\unitlength
548 <latexrelease> \@ovyy #2\unitlength
549 <latexrelease> \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovxx\else \@ovyy \fi
550 <latexrelease> \advance \@tempdimb -2\p@
551 <latexrelease> \@getcirc \@tempdimb
552 <latexrelease> \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
553 <latexrelease> \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
554 <latexrelease> \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
555 <latexrelease> \@circlefnt \setbox\@tempboxa
556 <latexrelease> \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
557 <latexrelease> \if@ovl
558 <latexrelease> \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
559 <latexrelease> \fi
560 <latexrelease> \if@ovt \@ovhorz \kern -\@ovxx \fi
561 <latexrelease> \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
562 <latexrelease> \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
563 <latexrelease> \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
564 <latexrelease> \endgroup}
565 <latexrelease>\EndIncludeInRelease
566 <*2ekernel>

```

(End of definition for \@oval.)

\@ovvert

```

567 </2ekernel>
568 <latexrelease>\IncludeInRelease{2016/03/31}%
569 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
570 <*2ekernel | latexrelease>
571 \def\@ovvert#1#2{\vbox to\@ovyy{%
572 \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
573 \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
574 \else \kern \@ovri \kern \@ovdy \fi
575
576 \if@ovvline \leaders\vrule \@width \@wholewidth \fi
577
578 \vfil \nointerlineskip
579 \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
580 \hbox{\char \@tempcntb}%
581 \else \kern \@ovdy \kern \@ovro \fi}}
580 </2ekernel | latexrelease>
581 <latexrelease>\EndIncludeInRelease
582 <latexrelease>\IncludeInRelease{0000/00/00}%
583 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
584 <latexrelease>\def\@ovvert#1#2{\vbox to\@ovyy{%
585 <latexrelease> \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
586 <latexrelease> \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
587 <latexrelease> \else \kern \@ovri \kern \@ovdy \fi
588 <latexrelease> \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
589 <latexrelease> \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
590 <latexrelease> \hbox{\char \@tempcntb}%
591 <latexrelease> \else \kern \@ovdy \kern \@ovro \fi}}
592 <latexrelease>\EndIncludeInRelease

```

```

593 <*2ekernel>

(End of definition for \@ovvert.)

\@ovhorz

594 </2ekernel>
595 <latexrelease>\IncludeInRelease{2016/03/31}%
596 <latexrelease>          {\@ovhorz}{Avoid almost zero length leaders}%
597 <*2ekernel | latexrelease>
598 \def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
599   \if@ovr \else \kern \@ovdx \fi

600   \if@ovhline \leaders \hrule \@height \@wholewidth \fi

601   \hfil
602   \if@ovl \else \kern \@ovdx \fi
603   \kern \@ovri}}
604 </2ekernel | latexrelease>

605 <latexrelease>\EndIncludeInRelease
606 <latexrelease>\IncludeInRelease{0000/00/00}%
607 <latexrelease>          {\@ovhorz}{Avoid almost zero length leaders}%
608 <latexrelease>\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
609 <latexrelease>   \if@ovr \else \kern \@ovdx \fi
610 <latexrelease>   \leaders \hrule \@height \@wholewidth \hfil
611 <latexrelease>   \if@ovl \else \kern \@ovdx \fi
612 <latexrelease>   \kern \@ovri}}
613 <latexrelease>\EndIncludeInRelease
614 <*2ekernel>

(End of definition for \@ovhorz.)

\circle

615 \def\circle{\@inmatherr\circle\@ifstar\@dot\@circle}

(End of definition for \circle.)

\@circle

616 </2ekernel>
617 <*2ekernel | latexrelease>
618 <latexrelease>\IncludeInRelease{2020/10/01}%
619 <latexrelease>          {\@circle}{default units}%
620 \def\@circle#1{%
621   \begingroup \boxmaxdepth \maxdimen
622   \@defaultunitsset\@tempdimb{#1}\unitlength
623   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
624     \@ovro\ht\@tempboxa
625     \setbox\@tempboxa\hbox{\@circlefnt
626     \advance\@tempcnta\tw@ \char \@tempcnta
627     \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
628     \advance\@tempcnta\tw@
629     \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
630     \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
631     \put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
632   \else \@circ\@tempdimb{96}\fi\endgroup}
633 </2ekernel | latexrelease>

```

```

634 <latexrelease>\EndIncludeInRelease
635 <latexrelease>\IncludeInRelease{0000/00/00}%
636 <latexrelease>          {\@circle}{default units}%
637 <latexrelease>\def\@circle#1{%
638 <latexrelease>  \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
639 <latexrelease>  \ifdim \@tempdimb >15.5\p@ \getcirc\@tempdimb
640 <latexrelease>  \@ovro\ht\@tempboxa
641 <latexrelease>  \setbox\@tempboxa\hbox{\@circlefnt
642 <latexrelease>  \advance\@tempcnta\tw@ \char \@tempcnta
643 <latexrelease>  \advance\@tempcnta\m@ne \char \@tempcnta
644 <latexrelease>  \kern -2\@tempdima
645 <latexrelease>  \advance\@tempcnta\tw@
646 <latexrelease>  \raise \@tempdima \hbox{\char\@tempcnta}%
647 <latexrelease>  \raise \@tempdima
648 <latexrelease>  \box\@tempboxa\ht\@tempboxa\z@ \dp\@tempboxa\z@
649 <latexrelease>  \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
650 <latexrelease>  \else \@circ\@tempdimb{96}\fi\endgroup}
651 <latexrelease>\EndIncludeInRelease
652 <*2ekernel>

```

(End of definition for \@circle.)

\@dot Internal form of \circle*.

```

653 </2ekernel>
654 <*2ekernel | latexrelease>
655 <latexrelease>\IncludeInRelease{2020/10/01}%
656 <latexrelease>          {\@dot}{default units}%
657 \def\@dot#1{%
658   \@defaultunitsset\@tempdimb{#1}\unitlength
659   \@circ\@tempdimb{112}}
660 </2ekernel | latexrelease>
661 <latexrelease>\EndIncludeInRelease
662 <latexrelease>\IncludeInRelease{0000/00/00}%
663 <latexrelease>          {\@dot}{default units}%
664 <latexrelease>\def\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
665 <latexrelease>\EndIncludeInRelease
666 <*2ekernel>

```

(End of definition for \@dot.)

\@circ

```

667 \def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
668   \@tempcnta\@tempdima \@tempdima \p@
669   \divide\@tempcnta\@tempdima
670   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
671   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
672   \advance\@tempcnta #2\relax
673   \@circlefnt \char\@tempcnta}

```

(End of definition for \@circ.)

\@xarg Counters used for manipulating the ‘slope’ arguments.

```

\@yarg 674 \newcount\@xarg
\@yyarg 675 \newcount\@yarg
        676 \newcount\@yyarg

```

(End of definition for \@xarg, \@yarg, and \@yyarg.)

\@multicnt Counter used in \multiput, and also \multicolumn.
677 \newcount\@multicnt

(End of definition for \@multicnt.)

\@xdim Length registers.
\@ydim 678 \newdimen\@xdim
679 \newdimen\@ydim

(End of definition for \@xdim and \@ydim.)

\@linechar Box for holding a line segment character, for sloping lines.
680 \newbox\@linechar

(End of definition for \@linechar.)

\@linelen Length of the line currently being built.
681 \newdimen\@linelen

(End of definition for \@linelen.)

\@clnwd Height and width of current line segment.
\@clnht 682 \newdimen\@clnwd
683 \newdimen\@clnht

(End of definition for \@clnwd and \@clnht.)

\@dashdim \dashbox internal registers.
\@dashbox 684 \newdimen\@dashdim
\@dashcnt 685 \newbox\@dashbox
686 \newcount\@dashcnt

(End of definition for \@dashdim, \@dashbox, and \@dashcnt.)

Initialization: “\thinlines”

687 \let\@linefnt\tenln
688 \let\@circlefnt\tencirc
689 \@wholewidth\fontdimen8\tenln
690 \@halfwidth .5\@wholewidth

1.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
  IF N = 0
    THEN \@xdima := |BX - AX|
        \@xb := |CX - BX|
        \@xa := Max(\@xa, \@xb)
        \@ya := |BY - AY|
        \@yb := |CY - BY|
        \@ya := Max(\@ya, \@yb)
        @sc := Max(\@xa, \@ya)
        %% The coefficient .5 below is the degree of overlap of
        %% successive points, where 1 is no overlap and 0 is
        %% complete overlap. A coefficient of C multiplies
        %% the number of points plotted by 1/C.
        %%
        \@xa := .5 * \@halfwidth
        @sc := @sc / \@halfwidth
        @sc := Max(@sc, qbeziermax)
    ELSE @sc := N
  @scp := @sc+1
  \@xb := 2 * (BX - AX) * \unitlength
  \@xa := ((CX-AX)*\unitlength - \@xb)/@sc
  \@yb := 2 * (BY - AY) * \unitlength
  \@ya := ((CY-AY)*\unitlength - \@yb)/@sc
  \@pictdot := square rule of width \@wholewidth
  \count@ := 0
  WHILE \count@ < @scp
    DO \@xdim := ((\count@*\@xa + @xb) / @sc) * \count@
        \@ydim := ((\count@*\@ya + @yb) / @sc) * \count@
        plot pt with relative coords (\@xdim,\@ydim)
        \count@ := \count@+1
  OD
```

End of historical L^AT_EX 2.09 comments.

```
\qbeziermax The maximum number of points to plot.
691 \def\qbeziermax{500}
```

(End of definition for \qbeziermax.)

In the code below, to save registers \@a ... are not used. Instead other registers are reused.

```
\newcounter{@sc} -> \c@multicnt
\newcounter{@scp} -> \@tempcnta
\newdimen@xa -> \@ovxx
\newdimen@xb -> \@ovdx
\newdimen@ya -> \@ovyy
\newdimen@yb -> \@ovdy
\newsavebox{@pictdot} -> \@tempboxa
```

\qbezier Main user-level command to plot quadratic bezier curves. #2 should be (.

```
692 \newcommand\qbezier[2][0]{\bezier{#1}#2}
```

(End of definition for \qbezier.)

\bezier Form of \bezier compatible with 2.09 bezier.sty, but modified to ignore spaces between its arguments. #2 should be white space, and #4 should be (.

```
693 \def\bezier#1#2(#3)#4({\@bezier#1)(#3)(}
```

\@bezier

```
694 {/2kernel}
695 {*2kernel|latexrelease}
696 {latexrelease}\IncludeInRelease{2020/10/01}%
697 {latexrelease} {\@bezier}{default units}%
698 \def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
699   \ifnum #1=\z@
700     \@defaultunitsset\@ovxx{#4}\unitlength
701     \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
702     \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
703     \@defaultunitsset\@ovdx{#6}\unitlength
704     \@defaultunitsset{\advance\@ovdx}{-#4}\unitlength
705     \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
706     \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
707     \@defaultunitsset\@ovyy{#5}\unitlength
708     \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
709     \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
710     \@defaultunitsset\@ovdy{#7}\unitlength
711     \@defaultunitsset{\advance\@ovdy}{-#5}\unitlength
712     \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
713     \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
714     \@multicnt
715     \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
716     \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
717     \ifnum \qbeziermax<\@multicnt
718       \@multicnt\qbeziermax\relax
719     \fi
720   \else \@multicnt#1\relax \fi
721   \@tempcnta\@multicnt \advance\@tempcnta\@one
722   \@defaultunitsset\@ovdx{#4}\unitlength
723   \@defaultunitsset{\advance\@ovdx}{-#2}\unitlength
724   \multiply\@ovdx \tw@
725   \@defaultunitsset\@ovxx{#6}\unitlength
```

```

726 \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
727 \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
728 \@defaultunitsset\@ovdy{#5}\unitlength
729 \@defaultunitsset{\advance\@ovdy}{-#3}\unitlength
730 \multiply\@ovdy \tw@
731 \@defaultunitsset\@ovyy{#7}\unitlength
732 \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
733 \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt

734 \setbox\@tempboxa\hbox{%
735 \hskip -\@halfwidth
736 \vrule \@height\@halfwidth
737 \@depth \@halfwidth
738 \@width \@wholewidth}%
739 \put(#2,#3){%
740 \count@\z@
741 \@whilenum{\count@<\@tempcnta}\do
742 {\@xdim\count@\@ovxx
743 \advance\@xdim\@ovdx
744 \divide\@xdim\@multicnt
745 \multiply\@xdim\count@
746 \@ydim\count@\@ovyy
747 \advance\@ydim\@ovdy
748 \divide\@ydim\@multicnt
749 \multiply\@ydim\count@
750 \raise \@ydim
751 \hb@xt@\z@{\kern\@xdim
752 \unhcopy\@tempboxa\hss}%
753 \advance\count@\@ne}}
754 \endkernel | latexrelease)

755 \latexrelease\EndIncludeInRelease
756 \latexrelease\IncludeInRelease{0000/00/00}%
757 \latexrelease{\@bezier}{default units}%
758 \latexrelease\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
759 \latexrelease \ifnum #1=\z@
760 \latexrelease \@ovxx #4\unitlength
761 \latexrelease \advance\@ovxx -#2\unitlength
762 \latexrelease \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
763 \latexrelease \@ovdx #6\unitlength
764 \latexrelease \advance\@ovdx -#4\unitlength
765 \latexrelease \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
766 \latexrelease \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
767 \latexrelease \@ovyy #5\unitlength
768 \latexrelease \advance\@ovyy -#3\unitlength
769 \latexrelease \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
770 \latexrelease \@ovdy #7\unitlength
771 \latexrelease \advance\@ovdy -#5\unitlength
772 \latexrelease \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
773 \latexrelease \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
774 \latexrelease \@multicnt
775 \latexrelease \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
776 \latexrelease \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
777 \latexrelease \ifnum
778 \latexrelease \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax

```

```

779 <latexrelease> \fi
780 <latexrelease> \else \@multicnt#1\relax \fi
781 <latexrelease> \@tempcnta\@multicnt \advance\@tempcnta\@ne
782 <latexrelease> \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
783 <latexrelease> \multiply\@ovdx \tw@
784 <latexrelease> \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
785 <latexrelease> \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
786 <latexrelease> \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
787 <latexrelease> \multiply\@ovdy \tw@
788 <latexrelease> \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
789 <latexrelease> \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
790 <latexrelease> \setbox\@tempboxa\hbox{%
791 <latexrelease> \hskip -\@halfwidth
792 <latexrelease> \vrule \@height\@halfwidth
793 <latexrelease> \@depth \@halfwidth
794 <latexrelease> \@width \@wholewidth}%
795 <latexrelease> \put(#2,#3){%
796 <latexrelease> \count@\z@
797 <latexrelease> \@whilenum{\count@<\@tempcnta}\do
798 <latexrelease> {\@xdim\count@\@ovxx
799 <latexrelease> \advance\@xdim\@ovdx
800 <latexrelease> \divide\@xdim\@multicnt
801 <latexrelease> \multiply\@xdim\count@
802 <latexrelease> \@ydim\count@\@ovyy
803 <latexrelease> \advance\@ydim\@ovdy
804 <latexrelease> \divide\@ydim\@multicnt
805 <latexrelease> \multiply\@ydim\count@
806 <latexrelease> \raise \@ydim
807 <latexrelease> \hb@xt@\z@{\kern\@xdim
808 <latexrelease> \unhcopy\@tempboxa\hss}%
809 <latexrelease> \advance\count@\@ne}}
810 <latexrelease>\EndIncludeInRelease
811 <*2kernel>

```

(End of definition for \bezier and \@bezier.)

As the commands above all use “picture” interface we couldn’t define them with \DeclareRobustCommand so we do that now.

```

812 </2kernel>
813 <*2kernel | latexrelease>
814 <latexrelease>\IncludeInRelease{2019/10/01}%
815 <latexrelease> {\bezier}{Make commands robust}%
816 \MakeRobust\bezier
817 \MakeRobust\circle
818 \MakeRobust\dashbox
819 \MakeRobust\line
820 \MakeRobust\linethickness
821 \MakeRobust\multiput
822 \MakeRobust\oval
823 \MakeRobust\put
824 \MakeRobust\qbezier
825 \MakeRobust\shortstack
826 \MakeRobust\thinline
827 \MakeRobust\vector
828 </2kernel | latexrelease>

```



```

829 <latexrelease>\EndIncludeInRelease
830 <latexrelease>\IncludeInRelease{0000/00/00}%
831 <latexrelease>          {\bezier}{Make commands robust}%
832 <latexrelease>
833 <latexrelease>\kernel@make@fragile\bezier
834 <latexrelease>\kernel@make@fragile\circle
835 <latexrelease>\kernel@make@fragile\dashbox
836 <latexrelease>\kernel@make@fragile\line
837 <latexrelease>\kernel@make@fragile\linethickness
838 <latexrelease>\kernel@make@fragile\multiput
839 <latexrelease>\kernel@make@fragile\oval
840 <latexrelease>\kernel@make@fragile\put
841 <latexrelease>\kernel@make@fragile\qbezier
842 <latexrelease>\kernel@make@fragile\shortstack
843 <latexrelease>\kernel@make@fragile\thinline
844 <latexrelease>\kernel@make@fragile\vector
845 <latexrelease>
846 <latexrelease>\EndIncludeInRelease
847 <*2ekernel>
848 </2ekernel>

```

File 43

ltthm.dtx

1 Theorem Environments

The user creates his own theorem-like environments with the command

`\newtheorem{<name>}{<text>}[<counter>]` or

`\newtheorem{<name>}[<oldname>]{<text>}`

This defines the environment `<name>` to be just as one would expect a theorem environment to be, except that it prints `<text>` instead of “Theorem”.

If `<oldname>` is given, then environments `<name>` and `<oldname>` use the same counter, so using a `<name>` environment advances the number of the next `<name>` environment, and vice-versa.

If `<counter>` is given, then environment `<name>` is numbered within `<counter>`.

E.g., if `<counter> = subsection`, then the first `<name>` in subsection 7.2 is numbered `<text> 7.2.1`.

The way `<name>` environments are numbered can be changed by redefining `\the<name>`.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

DOCUMENT STYLE PARAMETERS

`\@thmcounter{COUNTER}` : A command such that

`\edef\theCOUNTER{\@thmcounter{COUNTER}}`

defines `\theCOUNTER` to produce a number for a theorem environment.

The default is:

`BEGIN \noexpand\arabic{COUNTER} END`

`\@thmcountersep` : A separator placed between a theorem number and the number of the counter within which it is numbered.

E.g., to make the third theorem of section 7.2 be numbered 7.2-3, `\@thmcountersep` should be `\def`'ed to `'-'`. Its default is `''`.

`\@begintheorem{NAME}{NUMBER}` : A command that begins a theorem environment for a 'theorem' named 'NAME NUMBER' –

e.g., `\@begintheorem{Lemma}{3.7}` starts Lemma 3.7.

`\@opargbegintheorem{NAME}{NUMBER}{OPARG}` :

A command that begins a theorem environment for a 'theorem' named 'NAME NUMBER' with optional argument OPARG – e.g., `\@begintheorem{Lemma}{3.7}{Jones}` starts 'Lemma 3.7 (Jones)':.

`\@endtheorem` : A command that ends a theorem environment.

`\newtheorem{NAME}{TEXT}[COUNTER] ==`

`BEGIN`

if `\NAME` is definable

```

then \@definecounter{NAME}
  if COUNTER present
    then \@newctr{NAME}[COUNTER] fi
    \theNAME == BEGIN \theCOUNTER \@thmcountersep
                                eval\@thmcounter{NAME} END
    else \theNAME == BEGIN eval\@thmcounter{NAME} END
    \NAME == \@thm{NAME}{TEXT}
    \endNAME == \@endtheorem
  else error
fi
END

\newtheorem{NAME}[OLDNAME]{TEXT}==
BEGIN
  if counter OLDNAME nonexistent
  then ERROR
  else
    if \NAME is definable
    then BEGIN
      \theNAME == \theOLDNAME
      \NAME == \@thm{OLDNAME}{TEXT}
      \endNAME == \@endtheorem
      END
    else error
    fi
  fi
END

\@thm{NAME}{TEXT} ==
BEGIN
  \refstepcounter{NAME}
  if next char = [
    then \@ythm{NAME}{TEXT}
    else \@xthm{NAME}{TEXT}
  fi
END

\@xthm{NAME}{TEXT} ==
BEGIN
  \@begintheorem{TEXT}{\theNAME}
  \ignorespaces
END

\@ythm{NAME}{TEXT}[OPARG] ==
BEGIN
  \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
  \ignorespaces
END

```

End of historical L^AT_EX 2.09 comments.

`\newtheorem` `\newtheorem` ought really be allowed only in the preamble Which would be good document style, and allow some main memory to be saved by declaring these commands to be `\onlypreamble`. Unfortunately the L^AT_EX book indicates that `\newtheorem` may be used anywhere in the document...

```

1 <*2ekernel>
2 \def\newtheorem#1{%
3   \ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

(End of definition for `\newtheorem`.)

`\@nthm`

```

4 \def\@nthm#1#2{%
5   \ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

(End of definition for `\@nthm`.)

`\@xnthm` 92/09/18 RmS: Changed `\@addtoreset` to `\@newctr` to produce error message if counter #3 does not exist (to be consistent with behaviour of `\newcounter`)

```

6 \def\@xnthm#1#2[#3]{%
7   \expandafter\@ifdefinable\csname #1\endcsname
8     {\@definecounter{#1}\@newctr{#1}[#3]%
9     \expandafter\xdef\csname the#1\endcsname{%
10      \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
11      \@thmcounter{#1}}}%
12   \global\@namedef{#1}{\@thm{#1}{#2}}%
13   \global\@namedef{end#1}{\@endtheorem}}}
```

(End of definition for `\@xnthm`.)

`\@ynthm`

```

14 \def\@ynthm#1#2{%
15   \expandafter\@ifdefinable\csname #1\endcsname
16     {\@definecounter{#1}%
17     \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}}%
18   \global\@namedef{#1}{\@thm{#1}{#2}}%
19   \global\@namedef{end#1}{\@endtheorem}}}
```

(End of definition for `\@ynthm`.)

`\@othm`

```

21 <*2ekernel | latexrelease>
22 <latexrelease>\IncludeInRelease{2026/06/01}%
23 <latexrelease>          {\@othm}{use alias counter}%
24 \def\@othm#1[#2]#3{%
25   \ifundefined{c@#2}{\@nocounterr{#2}}%
26   {\expandafter\@ifdefinable\csname #1\endcsname
27     {\newcounteralias{#1}{#2}%
28     \global\@namedef{#1}{\@thm{#1}{#3}}%
29     \global\@namedef{end#1}{\@endtheorem}}}}
30 <latexrelease>\EndIncludeInRelease
31 <latexrelease>\IncludeInRelease{0000/00/00}%
32 <latexrelease>          {\@othm}{use alias counter}%
33 <latexrelease>\def\@othm#1[#2]#3{%
```

```

34 <latexrelease> \ifundefined{c@#2}{\@nocounterr{#2}}%
35 <latexrelease> {\expandafter\@ifdefinable\csname #1\endcsname
36 <latexrelease> {\global\@namedef{the#1}{\@nameuse{the#2}}}%
37 <latexrelease> \global\@namedef{#1}{\@thm{#2}{#3}}%
38 <latexrelease> \global\@namedef{end#1}{\@endtheorem}}}%
39 <latexrelease>\EndIncludeInRelease
40 </2ekernel | latexrelease>

```

(End of definition for \@othm.)

\@thm

```

41 <*2ekernel | latexrelease>
42 <latexrelease>\IncludeInRelease{2024/03/18}%
43 <latexrelease> {\@thm}{no link target}%
44 \def\@thm#1#2{%
45 \kernel@refstepcounter{#1}%
46 \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
47 <latexrelease>\EndIncludeInRelease
48 <latexrelease>\IncludeInRelease{0000/00/00}%
49 <latexrelease> {\@thm}{no link target}%
50 <latexrelease>\def\@thm#1#2{%
51 <latexrelease> \refstepcounter{#1}%
52 <latexrelease> \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
53 <latexrelease>\EndIncludeInRelease
54 </2ekernel | latexrelease>

```

(End of definition for \@thm.)

\@xthm

\@ythm

```

55 <*2ekernel>
56 \def\@xthm#1#2{%
57 \@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
58 \def\@ythm#1#2[#3]{%
59 \@opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}

```

(End of definition for \@xthm and \@ythm.)

Default values

\@thmcounter

\@thmcountersep

```

60 \def\@thmcounter#1{\noexpand\arabic{#1}}
61 \def\@thmcountersep{.}

```

(End of definition for \@thmcounter and \@thmcountersep.)

\@begintheorem

Providing theorem defaults.

\@opargbegintheorem

\@endtheorem

```

62 </2ekernel>
63 <*2ekernel | latexrelease>
64 <latexrelease>\IncludeInRelease{2024/03/18}%
65 <latexrelease> {\@begintheorem}{add link targets}%
66 \def\@begintheorem#1#2{\trivlist
67 \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2}]\itshape}
68 \def\@opargbegintheorem#1#2#3{\trivlist
69 \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
70 <latexrelease>\EndIncludeInRelease
71 <latexrelease>\IncludeInRelease{0000/00/00}%

```

```

72 <latexrelease>                {\@begintheorem}{add link targets}%
73 <latexrelease>\def\@begintheorem#1#2{\trivlist
74 <latexrelease>    \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
75 <latexrelease>\def\@opargbegintheorem#1#2#3{\trivlist
76 <latexrelease>    \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
77 <latexrelease>\EndIncludeInRelease
78 </2ekernel | latexrelease>
79 <*2ekernel>
80 \def\@endtheorem{\endtrivlist}
81 </2ekernel>

```

(End of definition for \@begintheorem, \@opargbegintheorem, and \@endtheorem.)

File 44

ltsect.dtx

1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the L^AT_EX kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
1 <*2ekernel>
2 \message{title,}
```

1.1 The Title

`\title` The user defines the title and author by the declarations `\title{<name>}`, `\author{<name>}`
`\author` Similarly the date is declared with `\date{<date>}`.
`\date` Inside these, the `\thanks{<footnote text>}` command may be used to make acknowl-
`\thanks` edgements, notice of address, etc. in a footnote. If there are multiple authors, they have
`\and` to be separated with the `\and` command.
`\maketitle` And finally, the `\maketitle` command produces the actual title, using the informa-
tion previously saved with the other commands.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease>                {\title}{Make commands robust}%
```

\title `\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.

```
7 \DeclareRobustCommand\title[1]{\gdef\@title{#1}}
```

(End of definition for \title. This function is documented on page 854.)

`\author` `\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.

```
8 \DeclareRobustCommand*\author[1]{\gdef\@author{#1}}
```

(End of definition for \author.)

`\date` `\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
9 \DeclareRobustCommand*\date[1]{\gdef\@date{#1}}
```

(End of definition for \date.)

`\thanks`

```
10 \DeclareRobustCommand\thanks[1]{\footnotemark
11     \protected@xdef\@thanks{\@thanks
12         \protect\footnotetext[\the\c@footnote]{#1}}%
13 }
```

(End of definition for \thanks.)

`\and`

```
14 \DeclareRobustCommand\and{%    % \begin{tabular}
15   \end{tabular}}%
16   \hskip 1em \@plus.17fil%
17   \begin{tabular}[t]{c}}%    % \end{tabular}
```

(End of definition for \and.)

```
18 </2kernel | latexrelease>
19 <latexrelease>\EndIncludeInRelease
20 <latexrelease>\IncludeInRelease{0000/00/00}%
21 <latexrelease>          {\title}{Make commands robust}%
22 <latexrelease>
23 <latexrelease>\kernel@make@fragile\title
24 <latexrelease>\kernel@make@fragile\author
25 <latexrelease>\kernel@make@fragile\date
26 <latexrelease>\kernel@make@fragile\thanks
27 <latexrelease>\kernel@make@fragile\and
28 <latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <*2kernel>
```

`\@title`

```
31 \def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

(End of definition for \@title.)

`\@author`

```
32 \def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

(End of definition for \@author.)

`\@date`

```
33 \gdef\@date{\today}
```

(End of definition for \@date.)

`\@thanks`

```
34 \let\@thanks\@empty
```

(End of definition for \@thanks.)

```
35 \message{sectioning,}
```

1.2 Sectioning

`\@secpenalty`

```
36 \newcount\@secpenalty
37 \@secpenalty = -300
```

(End of definition for \@secpenalty.)

`\if@noskipsec` Way back in 1991 (08/26) FMi & RmS set the `@noskipsec` switch to true for the preamble
`\@noskipsectrue` and to false in `\document`. This was done to trap lists and related text in the preamble
but it does not catch everything.

```
38 \newif\if@noskipsec \@noskipsectrue
```


(End of definition for `\ifnoskipsec` and `\@noskipsectrue`.)

`\@startsection` The `\@startsection{<name>}{<level>}{<indent>}{<before skip>}{<after skip>}{<style>}`*[`<altheading>`] [`<heading>`] command is the mother of all the user level sectioning commands. The part after the *, including the * is optional.

name: e.g., 'subsection'

level: a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

indent: Indentation of heading from left margin

before skip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

after skip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as `\MakeUppercase` that takes an argument but doesn't start a paragraph (so `\fbox` or `\textit`, for example, wouldn't work without breaking the formatting). The section heading will be supplied as the argument to this command. So setting #6 to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If '*' is missing, then increment the counter. If it is present, then there should be no [`<altheading>`] argument. The command uses the counter 'secnumdepth'. It contains a pointer to the highest section level that is to be numbered.

Warning: The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```
\def\foo{ \begingroup ...
           \paragraph{...}
           \endgroup}
```

Pseudocode for the `\@startsection` command *Historical L^AT_EX 2.09 comments* (not necessarily accurate any more):

```
\@startsection
{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE} ==
BEGIN
  IF  @noskipsec = T  THEN  \leavevmode FI
                                % true if previous section had no body.

  \par
  \@tempskipa := BEFORESKIP
  @afterindent := T
  IF \@tempskipa < 0  THEN  \@tempskipa := -\@tempskipa
                                @afterindent := F
  FI
  IF @nobreak = true
    THEN \everypar == null
    ELSE \addpenalty{\@secpenalty}
          \addvspace{\@tempskipa}
```

```

FI
IF * next
  THEN \@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}
  ELSE \@dblarg{\@sect
    {NAME}{LEVEL}{INDENT}
    {BEFORESKIP}{AFTERSKIP}{STYLE}}
FI
END

```

End of historical L^AT_EX 2.09 comments.

```

39 \def\@startsection#1#2#3#4#5#6{%
40   \if@noskipsec \leavevmode \fi
41   \par
42   \@tempskipa #4\relax
43   \@afterindenttrue
44   \ifdim \@tempskipa <\z@
45     \@tempskipa -\@tempskipa \@afterindentfalse
46   \fi
47   \if@nobreak
48     \everypar{}%
49   \else
50     \addpenalty\@secpenalty\addvspace\@tempskipa
51   \fi
52   \ifstar
53     {\@ssect{#3}{#4}{#5}{#6}}%
54     {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}

```

(End of definition for \@startsection.)

\@sect Pseudocode for the \@sect command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\@sect{NAME}{LEVEL}
  {INDENT}{BEFORESKIP}{AFTERSKIP}
  {STYLE}[ARG1]{ARG2}
  ==
BEGIN
  IF LEVEL > \c@secnumdepth
  THEN \@svsec :=L null
  ELSE \refstepcounter{NAME}
    \@svsec :=L BEGIN \@seccntformat{#1}\relax END
  FI
  IF AFTERSKIP > 0
  THEN \begingroup
    STYLE
    \@hangfrom{\hskip INDENT\@svsec}
    {\interlinepenalty 10000 ARG2\par}
  \endgroup
  \NAMEmark{ARG1}
  \addcontentsline{toc}{NAME}
  { IF LEVEL > \c@secnumdepth
    ELSE \protect\numberline{\theNAME} FI
  ARG1 }

```

```

ELSE \@svsechd == BEGIN STYLE
    \hskip INDENT\@svsec
    ARG2
    \NAMEmark{ARG1}
    \addcontentsline{toc}{NAME}
    { IF LEVEL > \c@secnumdepth
      ELSE
        \protect\numberline{\theNAME}
      FI
    ARG1 }
END

FI
\@xsect{AFTERSKIP}
END

```

End of historical L^AT_EX 2.09 comments.

```

55 \def\@sect#1#2#3#4#5#6[#7]#8{%
56   \ifnum #2>\c@secnumdepth
57     \let\@svsec\@empty
58   \else
59     \refstepcounter{#1}%

```

Since \@secntformat might end with an improper \hskip which is scanning forward for plus or minus we end the definition of \@svsec with \relax as a precaution.

```

60   \protected@edef\@svsec{\@secntformat{#1}\relax}%
61   \fi
62   \@tempskipa #5\relax
63   \ifdim \@tempskipa>\z@
64     \begingroup

```

This { used to be after the argument to \@hangfrom but was moved here to allow commands such as \MakeUppercase to be used at the end of #6.

```

65     #6{%
66       \@hangfrom{\hskip #3\relax\@svsec}%
67       \interlinepenalty \@M #8\@par}%
68   \endgroup
69   \csname #1mark\endcsname{#7}%
70   \addcontentsline{toc}{#1}{%
71     \ifnum #2>\c@secnumdepth \else
72       \protect\numberline{\csname the#1\endcsname}%
73     \fi
74     #7}%
75   \else

```

\relax added 2 May 90

```

76   \def\@svsechd{%
77     #6{\hskip #3\relax
78       \@svsec #8}%
79     \csname #1mark\endcsname{#7}%
80     \addcontentsline{toc}{#1}{%
81       \ifnum #2>\c@secnumdepth \else
82         \protect\numberline{\csname the#1\endcsname}%
83       \fi
84       #7}}%

```

```

85   \fi
86   \@xsect{#5}}

```

(End of definition for \@sect.)

`\@xsect` Pseudocode for the `\@xsect` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\@xsect{AFTERSKIP} ==
BEGIN
  IF AFTERSKIP > 0
  THEN \par \nobreak
       \vskip AFTERSKIP
       \@afterheading
  ELSE @nobreak :=G F
       @noskipsec :=G T
       \everypar{ IF @noskipsec = T
                   THEN @noskipsec :=G F
                       \clubpenalty := 10000 % local
                       \hskip -\parindent
                       \begingroup
                       \@svsechd
                       \endgroup
                       \unskip
                       \hskip -AFTERSKIP \relax
                                   %% relax added 14 Jan 91
                   ELSE \clubpenalty := \@clubpenalty % local
                       \everypar := NULL
                   FI
               }
  FI
FI

END

```

End of historical L^AT_EX 2.09 comments.

```

87 \def\@xsect#1{%
88   \@tempskipa #1\relax
89   \ifdim \@tempskipa>\z@

```

Why not combine `\@sect` and `\@xsect` and save doing the same test twice? It is not possible to change this now as these have become hooks!

This `\par` seems unnecessary.

```

90   \par \nobreak
91   \vskip \@tempskipa
92   \@afterheading
93 \else

```

```

94     \@nobreakfalse
95     \global\@noskipsectrue
96     \everypar{%
97         \if@noskipsec
98             \global\@noskipsecfalse
99             {\setbox\z@\lastbox}%
100             \clubpenalty\@M
101             \begingroup \@svsechd \endgroup
102             \unskip
103             \@tempskipa #1\relax
104             \hskip -\@tempskipa
105         \else
106             \clubpenalty \@clubpenalty
107             \everypar{}%
108         \fi}%
109     \fi
110     \ignorespaces}

```

(End of definition for \@xsect.)

\@secntformat This command formats the section number including the space following it.

```

111 \def\@secntformat#1{\csname the#1\endcsname\quad}

```

(End of definition for \@secntformat.)

Pseudocode for the \@ssect command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}{ARG} ==
BEGIN
  IF AFTERSKIP > 0
  THEN \begingroup
      STYLE
      \@hangfrom{\hskip INDENT}
          {\interlinepenalty 10000 ARG\par}
      \endgroup
  ELSE \@svsechd == BEGIN STYLE
      \hskip INDENT
      ARG
      END
  FI
  \@xsect{AFTERSKIP}
END

```

End of historical L^AT_EX 2.09 comments.

Pseudocode for the \@afterheading command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\@afterheading ==
BEGIN
  \@nobreak :=G true
  \everypar := BEGIN IF \@nobreak = T
      THEN \@nobreak :=G false
      \clubpenalty := 10000 % local
      IF \@afterindent = F
      THEN remove \lastbox

```

```

FI
ELSE \clubpenalty := \@clubpenalty % local
\everypar := NULL
FI
END
END

```

End of historical L^AT_EX 2.09 comments.

`\@ssect`

```

112 \def\@ssect#1#2#3#4#5{%
113   \@tempskipa #3\relax
114   \ifdim \@tempskipa>\z@
115     \begingroup
This { used to be after the argument to \@hangfrom but was moved here to allow com-
mands such as \MakeUppercase to be used at the end of #4.
116     #4{%
117       \@hangfrom{\hskip #1}%
118       \interlinepenalty \@M #5\@par}%
119     \endgroup
120   \else
121     \def\@svsechd{#4{\hskip #1\relax #5}}%
122   \fi
123   \@xsect{#3}}

```

(End of definition for \@ssect.)

`\if@afterindent`
`\@afterindenttrue`

```

124 \newif\if@afterindent \@afterindenttrue

```

(End of definition for \if@afterindent and \@afterindenttrue.)

`\@afterheading` This hook is used in setting up custom-built headings in classes.dtx.

```

125 \def\@afterheading{%
126   \@nobreaktrue
127   \everypar{%
128     \if@nobreak
129       \@nobreakfalse
130       \clubpenalty \@M
131     \if@afterindent \else
132       {\setbox\z@\lastbox}%
133     \fi
134   \else
135     \clubpenalty \@clubpenalty
136     \everypar{}%
137   \fi}}

```

(End of definition for \@afterheading.)

`\@hangfrom` `\@hangfrom{<text>}` : Puts <text> in a box, and makes a hanging indentation of the following material up to the first \par. Should be used in vertical mode.

```

138 \def\@hangfrom#1{\setbox\@tempboxa\hbox{#1}}%
139   \hangindent \wd\@tempboxa\noindent\box\@tempboxa}

```

(End of definition for \@hangfrom.)

```
\c@secnumdepth
\c@tocdepth 140 \newcount\c@secnumdepth
141 \newcount\c@tocdepth
```

(End of definition for \c@secnumdepth and \c@tocdepth.)

```
\secdef \secdef{\unstarcmds}{\unstarcmds}{\starcmds}
```

When defining a `\chapter` or `\section` command without using `\@startsection`, you can use `\secdef` as follows:

1. `\def\chapter{... \secdef \langle starcmd \rangle \langle unstarcmd \rangle }`
2. `\def\langle starcmd \rangle[#1]#2{...} % Command to define \chapter[...]{...}`
3. `\def\langle unstarcmd \rangle#1{...} % Command to define \chapter*{...}`

```
142 \def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}
```

(End of definition for `\secdef`.)

1.2.1 Initializations

```
\sectionmark
\subsectionmark 143 \let\sectionmark\@gobble
\subsubsectionmark 144 \let\subsectionmark\@gobble
\paragraphmark 145 \let\subsubsectionmark\@gobble
\subparagraphmark 146 \let\paragraphmark\@gobble
147 \let\subparagraphmark\@gobble
```

(End of definition for `\sectionmark` and others.)

```
148 \message{contents,}
```

1.3 Table of Contents etc.

1.3.1 Convention

`\tf@{foo}` = file number for output for table foo. The file is opened only if `@files` = true.

1.3.2 Commands

A `\l@{type}{\langle entry \rangle}{\langle page \rangle}` Macro needs to be defined by document style for making an entry of type `\langle type \rangle` in a table of contents, etc. E.g., the document style should define `\l@chapter`, `\l@section`, etc.

Note: When the `\protect` command is used in the `\langle entry \rangle` or `\langle text \rangle` of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an `\addcontentsline` or `\addtocontents` command argument, the commands: `\index`, `\glossary`, and `\label` are no-ops. This could cause a problem if the user puts an `\index` or `\label` into one of the commands he writes, or into the optional ‘short version’ argument of a `\section` or `\caption` command.

`\@starttoc` The `\@starttoc{<ext>}` command is used to define the commands:
`\tableofcontents`, `\listoffigures`, etc.

For example: `\@starttoc{lof}` is used in `\listoffigures`. This command reads the `.<ext>` file and sets up to write the new `.<ext>` file.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@starttoc{EXT} ==
BEGIN
  \begingroup
    \makeatletter
    read file \jobname.EXT
    IF @filesw = true
      THEN open \jobname.EXT as file \tf@EXT
    FI
    @nobreak :=G FALSE %% added 24 May 89
  \endgroup
END
```

End of historical L^AT_EX 2.09 comments.

```
149 \def\@starttoc#1{%
150   \begingroup
151     \makeatletter
152     \@input{\jobname.#1}%
153     \if@filesw
154       \expandafter\newwrite\csname tf@#1\endcsname
155       \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
156     \fi
157     \@nobreakfalse
158   \endgroup}
```

(End of definition for \@starttoc.)

`\addcontentsline` The `\addcontentsline{<table>}{<type>}{<entry>}` command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry `\contentsline{<type>}{<entry>}{<page>}{}` to the `.<table>` file.

This macro is implemented as an application of `\addtocontents`. Note that `\thepage` is not expandable during `\protected@write` therefore one gets the page number at the time of the `\shipout`.

```
159 </2kernel>
160 <*2kernel | latexrelease>
161 <latexrelease>\IncludeInRelease{2020/10/01}%
162 <latexrelease>          {\addcontentsline}{fourth argument}%
163 \def\addcontentsline#1#2#3{%
```

We add an empty brace pair at the end of `\contentsline` so that the number of argument is identical in documents with and without `hyperref`.

```
164   \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}{}}%
165   \protected@file@percent}%
166 </2kernel | latexrelease>
167 <latexrelease>\EndIncludeInRelease
168 <latexrelease>\IncludeInRelease{2018/12/01}%
169 <latexrelease>          {\addcontentsline}{Mask line endings}%
170 <latexrelease> \def\addcontentsline#1#2#3{%
171 <latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}%
```


We add `\protected@file@percent` at the end which is turned inside `\@writefile` into a percent character to mask the newline after the closing argument brace.

```

172 <latexrelease> \protected@file@percent}}
173 <latexrelease>\EndIncludeInRelease
174 <latexrelease>\IncludeInRelease{0000/00/00}%
175 <latexrelease> {&addcontentsline}{Mask line endings}%
176 <latexrelease>\def&addcontentsline#1#2#3{%
177 <latexrelease> \addtocontents{#1}{\protect&contentsline{#2}{#3}{\thepage}}}
178 <latexrelease>\EndIncludeInRelease
179 <*2ekernel>

```

(End of definition for &addcontentsline.)

`\@gobble@om` Each of these four commands accepts an optional and a mandatory argument, possibly preceded by a star. In all cases the expansion is empty or just manipulates the spaces around the command. Used to disable commands such as `\index` or `\label` in certain situations.

```

180 </2ekernel>
181 <*2ekernel | latexrelease>
182 <latexrelease>\IncludeInRelease{2025/06/01}%
183 <latexrelease> {\@gobble@som}{Extended index/label}%

```

Just getting rid of the input in an expandable way is done by these two.

```

184 \DeclareExpandableDocumentCommand&gobble@om{+o+m}{&}
185 \DeclareExpandableDocumentCommand&gobble@som{s+o+m}{&}

```

When something needs to be suppressed during typesetting, one often also wants to ensure that this doesn't end up with two spaces (in case there is one before and one after). This can't be done expandably.

```

186 \DeclareDocumentCommand&gobble@with@sphack@om{+o+m}{\@bsphack&esphack}
187 \DeclareDocumentCommand&gobble@with@sphack@som{s+o+m}{\@bsphack&esphack}

```

(End of definition for &gobble@om and others.)

`\addtocontents` The `\addtocontents{<table>}{<text>}` command adds `<text>` to the `.<table>` file, with no page number.

```

188 \long&def&addtocontents#1#2{%
189   \protected&write&auxout
190     {\let&label&gobble@om
191      \let&index&gobble@som
192      \let&glossary&gobble@om}%
193   {\string&writefile{#1}{#2}}}
194 </2ekernel | latexrelease>
195 <latexrelease>\EndIncludeInRelease
196 <latexrelease>\IncludeInRelease{0000/00/00}%
197 <latexrelease> {\@gobble@som}{Extended index/label}%
198 <latexrelease>\long&def&addtocontents#1#2{%
199 <latexrelease> \protected&write&auxout
200 <latexrelease> {\let&label&gobble \let&index&gobble \let&glossary&gobble}%
201 <latexrelease> {\string&writefile{#1}{#2}}}
202 <latexrelease>
203 <latexrelease>\let&gobble@om&@undefined
204 <latexrelease>\let&gobble@som&@undefined
205 <latexrelease>\let&gobble@with@sphack@om&@undefined

```

```

206 <latexrelease>\let\@gobble@with@sphack@som\@undefined
207 <latexrelease>\EndIncludeInRelease
208 <*2ekernel>

```

(End of definition for \addtocontents.)

\contentsline The `\contentsline{<type>}{<entry>}{<page>}{}` macro produces a `<type>` entry in a table of contents, etc. It will appear in the .toc or other file. For example, the entry for subsection 1.4.3 in the table of contents, might be produced by:

```

\contentsline{subsection}
{\numberline{1.4.3}Gnats and Gnus}{22}{ }

```

The `\protect` command causes command sequences to be written without expanding them.

```

209 </2ekernel>
210 <*2ekernel | latexrelease>
211 <latexrelease>\IncludeInRelease{2021/11/15}%
212 <latexrelease>{\contentsline}{Four arguments}%

```

In the toc file `\contentsline` is followed by 4 arguments these days, but only the first 3 are used in the old interface. The fourth was by default empty and only used when `hyperref` was loaded. We now pick up all 4 arguments, save the last one away in `\@contentsline@destination` and then call the old interface. This is done to simplify the interface to `hyperref` and to prepare for future changes.

```

213 \def\contentsline#1#2#3#4{\gdef\@contentsline@destination{#4}%
214 \csname l@#1\endcsname{#2}{#3}}

```

Default definition.

```

215 \let\@contentsline@destination\@empty
216 </2ekernel | latexrelease>
217 <latexrelease>\EndIncludeInRelease
218 <latexrelease>\IncludeInRelease{0000/00/00}%
219 <latexrelease>{\contentsline}{Four arguments}%
220 <latexrelease>
221 <latexrelease>\def\contentsline#1{\csname l@#1\endcsname}
222 <latexrelease>\let\@contentsline@destination\@undefined
223 <latexrelease>\EndIncludeInRelease
224 <*2ekernel>

```

(End of definition for \contentsline.)

`\@dottedtocline{<level>}{<indent>}{<numwidth>}{<title>}{<page>}`: Macro to produce a table of contents line with the following parameters:

level If `<level>` > `\c@tocdepth`, then no line produced.

indent Total indentation from the left margin.

numwidth Width of box for number if the `<title>` has a `\numberline` command. As of 25 Jan 1988, this is also the amount of extra indentation added to second and later lines of a multiple line entry.

title Contents of entry.

page Page number.

Uses the following parameters, which must be set by the document style. They should be defined with `\def`'s.

`pnumwidth` Width of box in which page number is set.

`tocrmarg` Right margin indentation for all but last line of multiple-line entries.

`dotsep` Separation between dots, in mu units. Should be `\def`'d to a number like 2 or 1.7

`\@dottedtocline`

```

225 </2ekernel>
226 <*2ekernel | latexrelease>
227 <latexrelease>\IncludeInRelease{2018/12/01}%
228 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
229 \def\@dottedtocline#1#2#3#4#5{%
230   \ifnum #1>\c@tocdepth \else
231     \vskip \z@ \@plus.2\p@
232     {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
233      \parindent #2\relax\@afterindenttrue
234      \interlinepenalty\@M
235      \leavevmode
236      \@tempdima #3\relax
237
238      \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
239      {#4}\nobreak
240      \leaders\hbox{$\m@th

```

If a document uses fonts other than computer modern, the use of a dot from math can be very disturbing despite the fact that this might be the only place in a document that then uses computer modern. Therefore we surround the dot with an `\hbox` to escape to the surrounding text font.

```

240       \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
241       mu$}\hfill
242       \nobreak
243       \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5%

```

We finish off by preventing any protrusion if that is enabled. If protrusion happens the number may shift to the right and as a result you may end up with an additional dot in the toc line in some situations.

```

244       \kern-\p@\kern\p@}%
245       \par}%
246       \fi}

```

(End of definition for \@dottedtocline.)

`\noptrusion` This command, if placed directly to the right (or left) of a word, will prevent protrusion of that word into the margin. It is used in the toc entry lines as they shouldn't protrude. It is implemented as to kerns that cancel each other but being there hide the word so that protrusion is not added. Note that a zero kern or an empty box would not work as the protrusion mechanism will skip over those.

```

247 \DeclareRobustCommand\noptrusion{\leavevmode\kern-\p@\kern\p@}

```

(End of definition for \noprotrusion.)

```

248 </2ekernel | latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <latexrelease>\IncludeInRelease{0000/00/00}%
251 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
252 <latexrelease>\def\@dottedtocline#1#2#3#4#5{%
253 <latexrelease>  \ifnum #1>\c@tocdepth \else
254 <latexrelease>    \vskip \z@ \@plus.2\p@
255 <latexrelease>    {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
256 <latexrelease>    \parindent #2\relax\@afterindenttrue
257 <latexrelease>    \interlinepenalty\@M
258 <latexrelease>    \leavevmode
259 <latexrelease>    \@tempdima #3\relax
260 <latexrelease>    \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
261 <latexrelease>    {#4}\nobreak
262 <latexrelease>    \leaders\hbox{$\m@th
263 <latexrelease>      \mkern \dotsep mu\hbox{.}\mkern \dotsep
264 <latexrelease>      mu$}\hfill
265 <latexrelease>    \nobreak
266 <latexrelease>    \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
267 <latexrelease>    \par}%
268 <latexrelease>  \fi}
269 <latexrelease>
270 <latexrelease>\let\noprotrusion\@undefined
271 <latexrelease>\EndIncludeInRelease
272 <*2ekernel>

```

Note: \nobreak's added 7 Jan 86 to prevent bad line break that left the page number dangling by itself at left edge of a new line.

Changed 25 Jan 88 to use \leftskip instead of \hangindent so leaders of multiple-line contents entries would line up properly.

\numberline **\numberline{<number>}**: For use in a \contentsline command. It puts <number> flush-left in a box of width \@tempdima (Before 25 Jan 88 change, it also added \@tempdima to the hanging indentation.)

```

273 \def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
274 </2ekernel>

```

(End of definition for \numberline.)

File 45

lfloat.dtx

1 Floats

The different types of floats are identified by a $\langle type \rangle$ name, which is the name of the counter for that kind of float. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each $\langle type \rangle$ has associated a positive $\langle type\ number \rangle$, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a $\langle placement\ specifier \rangle$, which is a list of the possible locations, each denoted by a letter as follows:

h : here	— at the current location in the text.
t : top	— at the top of a text page.
b : bottom	— at the bottom of a text page.
p : page	— on a separate float page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on ‘p’, float page positioning.) For example, ‘pht’ specifies that the float can appear in any of three locations: page, here or top.

1.1 Floating Environments

```
1  $\langle *2kernel \rangle$ 
2  $\backslash message\{floats,\}$ 
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

$\backslash c@topnumber$	: Number of floats allowed at the top of a column.
$\backslash topfraction$	: Fraction of column that can be devoted to floats.
$\backslash c@dbltopnumber$, $\backslash dbltopfraction$	: Same as above, but for double-column floats.
$\backslash c@bottomnumber$, $\backslash bottomfraction$	: Same as above for bottom of page.
$\backslash c@totalnumber$	: Number of floats allowed in a single column, including in-text floats.
$\backslash textfraction$	: Minimum fraction of column that must contain text.
$\backslash floatpagefraction$	: Minimum fraction of page that must be taken up by float page.
$\backslash dblfloatpagefraction$	: Same as above, for double-column floats.

The document style must define the following.

`\fps@TYPE` : The default placement specifier for floats of type TYPE.

`\ftype@TYPE` : The type number for floats of type TYPE.

`\ext@TYPE` : The file extension indicating the file on which the contents list for float type TYPE is stored.
For example, `\ext@figure = 'lof'`.

`\fnum@TYPE` : A macro to generate the figure number for a caption.
For example, `\fnum@TYPE == Figure \thefigure`.

`\@makecaption{NUM}{TEXT}` :
A macro to make a caption, with NUM the value produced by `\fnum@...` and TEXT the text of the caption. It can assume it's in a `\parbox` of the appropriate width.

`\@float{TYPE}[PLACEMENT]` : This macro begins a float environment for a single-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is defined by `\fps@TYPE`. The environment is ended by `\end@float`.
E.g., `\figure == \@float{figure}, \endfigure == \end@float`.

```
\@float{TYPE}[PLACEMENT] ==
BEGIN
  if hmode then \@bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  \@captype ==L TYPE
  \@dblflset
  \@fps ==L PLACEMENT
  \@onelevel@sanitize \@fps
  add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist nonempty
    then \@currbox :=L head of \@freelist
    \@freelist :=G tail of \@freelist
    \count\@currbox :=G 32*\ftype@TYPE +
                                bits determined by PLACEMENT
    else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
  fi
fi
\@currbox :=G \color@vbox
```

```

\normalcolor
\ vbox{
  %% 15 Dec 87 -
  %% removed \boxmaxdepth :=L 0pt
  %% that made box 0 depth because it screwed
  %% things up. Instead, added \vskip0pt at end
  \hsize = \columnwidth
  \@parboxrestore
  \@floatboxreset
END

```

```

\caption ==
BEGIN
  \refstepcounter{\@capttype}
  \@dblarg{\@caption{\@capttype}}
END

```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```

\@caption{TYPE}[STEXT]{TEXT} ==
BEGIN
  \par
  \addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
  \begingroup
    \@parboxrestore
    \@normalsize
    \makecaption{\fnum@TYPE}{TEXT}
  \par
  \endgroup
END

```

`\@dblfloat{TYPE}[PLACEMENT]` : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'

The environment is ended by `\end@dblfloat`.

E.g., `\figure* == \@dblfloat{figure}`,

`\endfigure* == \end@dblfloat`.

```

\@dblfloat{TYPE}[PLACEMENT] ==
  Identical to \@float{TYPE}[PLACEMENT] except \hsize and \linewidth
  are set to \textwidth.

```

End of historical L^AT_EX 2.09 comments.

`\@floatpenalty`

```

3 \newcount\@floatpenalty

```

(End of definition for \@floatpenalty.)

`\caption` This is set to be an error message outside a float since no `captype` is defined there; this may need to be changed by some classes.

```

4 \def\caption{%
5   \ifx\@captype\@undefined
6     \@latexerror{\noexpand\caption outside float}\@ehd
7     \expandafter\@gobble
8   \else
9     \refstepcounter\@captype
10    \expandafter\@firstofone
11  \fi
12  {\@dblarg{\@caption\@captype}}%
13 }

```

(End of definition for \caption.)

`\@caption`

```

14 \long\def\@caption#1[#2]#3{%
15   \par
16   \addcontentsline{\csname ext@#1\endcsname}{#1}%
17   {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
18   \begingroup

```

The paragraph setting parameters are normalized at this point, however `\@parboxrestore` resets `\everypar` which is not correct in this context so `\@setminipage` is called if needed.

The float mechanism, like `minipage`, sets the flag `@minipage` true before executing the user-supplied text. Many L^AT_EX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates T_EX's ‘top of page’ behaviour. The flag must be set false at the start of the first paragraph. This is achieved by a redefinition of `\everypar`, but the call to `\@parboxrestore` removes that redefinition, so it is re-inserted if needed. If the flag is already false then the `\caption` was not the first entry in the float, and so some other paragraph has already activated the special `\everypar`. In this case no further action is needed.

```

19   \@parboxrestore
20   \if@minipage
21     \@setminipage
22   \fi
23   \normalsize
24   \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
25   \endgroup}

```

(End of definition for \@caption.)

`\@float`
`\@dblflset`

```

26 \def\@float#1{%
27   \@ifnextchar[%
28     {\@xfloat{#1}}%
29     {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}}%
30     \reserved@a}

```

(End of definition for \@float and \@dblflset.)

`\@dblfloat`

```

31 \def\@dblfloat{%
32   \if@twocolumn\let\reserved@a\@dblft\else\let\reserved@a\@float\fi
33   \reserved@a}

```


(End of definition for \@dblfloat.)

`\fps@dbl` Note that all double floats have default fps ‘tp’.

(End of definition for \fps@dbl.)

`\@setfps` This sets the fps, dealing with error conditions by adding the default.

(End of definition for \@setfps.)

`\@xfloat` The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```

34 </2ekernel>
35 <latexrelease>\IncludeInRelease{2015/01/01}%
36 <latexrelease>                {\@xfloat}{Check float options}%
37 <*2ekernel | latexrelease>
38 \def\@xfloat #1[#2]{%
39   \@nodocument
40   \def \@capytype {#1}%
41   \def \@fps {#2}%
42   \@onelevel@sanitize \@fps
43   \def \reserved@a {!}%
44   \ifx \reserved@a \@fps
45     \fpsadddefault
46   \else
47     \ifx \@fps \@empty
48       \fpsadddefault
49     \fi
50   \fi
51   \ifhmode
52     \bsphack
53     \@floatpenalty -\@Mii
54   \else
55     \@floatpenalty-\@Miii
56   \fi
57   \ifinner
58     \@parmoderr\@floatpenalty\z@
59   \else
60     \@next\@currbox\@freelist
61     {%
62       \@tempcnta \sixt@@n
63       \expandafter \@tfor \expandafter \reserved@a
64       \expandafter : \expandafter =\@fps
65       \do

```

Start of changes, use a nested if structure, ending in an error.

```

66       {%
67       \if \reserved@a h%
68       \ifodd \@tempcnta
69       \else
70       \advance \@tempcnta \@ne
71       \fi

```

```

72         \else\if \reserved@a t%
73             \@setfpsbit \tw@
74         \else\if \reserved@a b%
75             \@setfpsbit 4%
76         \else\if \reserved@a p%
77             \@setfpsbit 8%
78         \else\if \reserved@a !%
79             \ifnum \@tempcnta>15
80                 \advance\@tempcnta -\sixt@@n\relax
81             \fi
82         \else
83             \@latex@error{Unknown float option ‘\reserved@a’}%
84             {Option ‘\reserved@a’ ignored and ‘p’ used.}%
85             \@setfpsbit 8%
86         \fi\fi\fi\fi\fi
87     }%

```

End of changes

```

88         \@tempcntb \csname ftype@\@capttype \endcsname
89         \multiply \@tempcntb \@xxxii
90         \advance \@tempcnta \@tempcntb
91         \global \count\@currbox \@tempcnta
92     }%
93     \@fltovf
94 \fi

```

The remainder sets up the box in which the float is typeset, and the typesetting environment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```

95     \global \setbox\@currbox
96         \color@vbox
97         \normalcolor
98         \vbox \bgroup
99             \hsize\columnwidth
100             \@parboxrestore
101             \@floatboxreset
102     }%
103 </2ekernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease>          {\@xfloat}{Check float options}%
107 <latexrelease>\def\@xfloat #1[#2]{%
108 <latexrelease>    \@nocument
109 <latexrelease>    \def \@capttype {#1}%
110 <latexrelease>    \def \@fps {#2}%
111 <latexrelease>    \@onelevel@sanitize \@fps
112 <latexrelease>    \def \reserved@b {!}%
113 <latexrelease>    \ifx \reserved@b \@fps
114 <latexrelease>        \@fpsadddefault
115 <latexrelease>    \else
116 <latexrelease>        \ifx \@fps \@empty
117 <latexrelease>            \@fpsadddefault

```

```

118 <latexrelease> \fi
119 <latexrelease> \fi
120 <latexrelease> \ifhmode
121 <latexrelease> \@bsphack
122 <latexrelease> \@floatpenalty -\@Mii
123 <latexrelease> \else
124 <latexrelease> \@floatpenalty-\@Miii
125 <latexrelease> \fi
126 <latexrelease> \ifinner
127 <latexrelease> \@parmoderr\@floatpenalty\z@
128 <latexrelease> \else
129 <latexrelease> \@next\@currbox\@freelist
130 <latexrelease> {%
131 <latexrelease> \@tempcnta \sixt@@n
132 <latexrelease> \expandafter \@tfor \expandafter \reserved@a
133 <latexrelease> \expandafter :\expandafter =\@fps
134 <latexrelease> \do
135 <latexrelease> {%
136 <latexrelease> \if \reserved@a h%
137 <latexrelease> \ifodd \@tempcnta
138 <latexrelease> \else
139 <latexrelease> \advance \@tempcnta \@ne
140 <latexrelease> \fi
141 <latexrelease> \fi
142 <latexrelease> \if \reserved@a t%
143 <latexrelease> \@setfpsbit \tw@
144 <latexrelease> \fi
145 <latexrelease> \if \reserved@a b%
146 <latexrelease> \@setfpsbit 4%
147 <latexrelease> \fi
148 <latexrelease> \if \reserved@a p%
149 <latexrelease> \@setfpsbit 8%
150 <latexrelease> \fi
151 <latexrelease> \if \reserved@a !%
152 <latexrelease> \ifnum \@tempcnta>15
153 <latexrelease> \advance\@tempcnta -\sixt@@n\relax
154 <latexrelease> \fi
155 <latexrelease> \fi
156 <latexrelease> }%
157 <latexrelease> \@tempcntb \csname ftype@\@capttype \endcsname
158 <latexrelease> \multiply \@tempcntb \@xxxii
159 <latexrelease> \advance \@tempcnta \@tempcntb
160 <latexrelease> \global \count\@currbox \@tempcnta
161 <latexrelease> }%
162 <latexrelease> \@fltovf
163 <latexrelease> \fi
164 <latexrelease> \global \setbox\@currbox
165 <latexrelease> \color@vbox
166 <latexrelease> \normalcolor
167 <latexrelease> \vbox \bgroup
168 <latexrelease> \hsize\columnwidth
169 <latexrelease> \@parboxrestore
170 <latexrelease> \@floatboxreset
171 <latexrelease> }%

```

```

172 <latexrelease>\EndIncludeInRelease
173 <*2ekernel>

```

(End of definition for \@xfloat.)

\@floatboxreset The rationale for allowing these normally global flags to be set locally here, via **\@parboxrestore**, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in **\@setnobreak**; otherwise this command will be redundant.

```

174 \def \@floatboxreset {%
175     \reset@font
176     \normalsize
177     \@setminipage
178 }

```

(End of definition for \@floatboxreset.)

\@setnobreak

```

179 \def \@setnobreak{%
180     \if@nobreak
181         \let\outer@nobreak\@nobreaktrue
182         \@nobreakfalse
183     \fi
184 }

```

(End of definition for \@setnobreak.)

\@setminipage

```

185 \def \@setminipage{%
186     \@minipagetrue
187     \everypar{\@minipagefalse\everypar{}}%
188 }

```

(End of definition for \@setminipage.)

\end@float

```

189 \def\end@float{%
190     \@endfloatbox
191     \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed **\textheight**, otherwise float will never get typeset (91/03/15 FMi).

```

192     \@largefloatcheck
193     \@cons\@currlist\@currbox
194     \ifnum\@floatpenalty <-\@Mii
195         \penalty -\@Miv

```

Saving and restoring **\prevdepth** added 26 May 87 to prevent extra vertical space when used in vertical mode.

```

196     \@tempdima\prevdepth
197     \vbox{}%
198     \prevdepth\@tempdima

```

```

199     \penalty\@floatpenalty

200     \else
201     \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
202     \fi
203     \fi
204 }

```

(End of definition for \end@float.)

\end@dblfloat

```

205 </2ekernel>
206 <latexrelease>\IncludeInRelease{2015/01/01}%
207 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
208 <*2ekernel | latexrelease>
209 \def\end@dblfloat{%
210     \if@twocolumn
211     \endfloatbox
212     \ifnum\@floatpenalty <\z@
213     \@largefloatcheck

```

Force the depth of two column float boxes.

```

214     \global\dp\@currbox1sp %

```

What follows is essentially \end@float without a starting \endfloatbox.

```

215     \@cons\@currlist\@currbox
216     \ifnum\@floatpenalty <-\@Mii
217     \penalty -\@Miv
218     \@tempdima\prevdepth
219     \vbox{}%
220     \prevdepth\@tempdima
221     \penalty\@floatpenalty
222     \else
223     \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
224     \fi

225     \fi
226     \else
227     \end@float
228     \fi
229 }%

```

```

230 </2ekernel | latexrelease>
231 <latexrelease>\EndIncludeInRelease
232 <latexrelease>\IncludeInRelease{0000/00/00}%
233 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
234 <latexrelease>\def\end@dblfloat{%
235 <latexrelease>\if@twocolumn
236 <latexrelease> \endfloatbox
237 <latexrelease> \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMi).

```

238 <latexrelease> \@largefloatcheck
239 <latexrelease> \@cons\@dbldeferlist\@currbox
240 <latexrelease> \fi

```

RmS 92/03/18 changed \@esphack to \@Esphack.

```
241 \<latexrelease> \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
242 \<latexrelease>\else
243 \<latexrelease> \end@float
244 \<latexrelease>\fi
245 \<latexrelease>\}%
246 \<latexrelease>\EndIncludeInRelease
247 \<*2ekernel>
```

(End of definition for \end@dblfloat.)

\@endfloatbox This macro is not intended to be a hook; it is designed to help maintain the integrity of this code, which is used twice and, as can be seen, is subject to frequent changes.

```
248 \def \@endfloatbox{%
249     \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

250     \@minipagefalse
251     \outer@nobreak
252     \egroup                %% end of vbox
253     \color@endbox
254 }
```

(End of definition for \@endfloatbox.)

\outer@nobreak

```
255 \let\outer@nobreak\@empty
```

(End of definition for \outer@nobreak.)

\@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
256 \def \@largefloatcheck{%
257     \ifdim \ht\@currbox>\textheight
258         \@tempdima -\textheight
259         \advance \@tempdima \ht\@currbox

260         \@latex@warning {Float too large for page by \the\@tempdima}%
261         \ht\@currbox \textheight
262     \fi
263 }
```

(End of definition for \@largefloatcheck.)

\@dbflt

\@xdblfloat

```
264 \def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
265 \def\@xdblfloat#1[#2]{%
266     \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}
```

(End of definition for \@dbflt and \@xdblfloat.)

Moved to ltoutput 93/12/16

```
267 %\newcount\c@topnumber
268 %\newcount\c@dbltopnumber
269 %\newcount\c@bottomnumber
270 %\newcount\c@totalnumber
```

`\@floatplacement` An analysis of `\@floatplacement`:
This should be called whenever `\@colht` has been set.

```

271 \def\@floatplacement{\global\@topnum\c@topnumber
272   % Textpage bit, global:
273   \global\@toproom \topfraction\@colht
274   \global\@botnum \c@bottomnumber
275   \global\@botroom \bottomfraction\@colht
276   \global\@colnum \c@totalnumber
277   % Floatpage bit, local:
278   \@fpmin \floatpagefraction\@colht}
279 \endkernel

```

(End of definition for \@floatplacement.)

`\@dblfloatplacement` This should be called only within a group. Now changed to provide extra checks in `\@addtodblcol`, needed when processing a BANG float.

```

280 \latexrelease\IncludeInRelease{2015/01/01}%
281 \latexrelease \def\@dblfloatplacement{\float order in 2-column}%
282 \endkernel | latexrelease

```

When making two column float area, look for floats with 1sp depth.

```

283 \def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
284   \global\@dbltoproom \dbltopfraction\@colht
285   \@textmin \@colht
286   \advance \@textmin -\@dbltoproom
287   \@fpmin \dblfloatpagefraction\textheight
288   \fptop \@dblfpsep
289   \fpsep \@dblfpsep
290   \fpbot \@dblfpbot

```

`\f@depth` is used in `\@testwrongwidth` to look for either column or dbl-column floats. A value of 1sp signals the latter. Because of this setting here, `\@dblfloatplacement` needs to be called inside a group which is a questionable design.

```

291 \def\f@depth{1sp}%
292 \endkernel | latexrelease
293 \latexrelease\EndIncludeInRelease
294 \latexrelease\IncludeInRelease{0000/00/00}%
295 \latexrelease \def\@dblfloatplacement{\float order in 2-column}%
296 \latexrelease\def \@dblfloatplacement {%

```

Textpage bit: global, but need not be.

```

297 \latexrelease \global \@dbltopnum \c@dbltopnumber
298 \latexrelease \global \@dbltoproom \dbltopfraction\@colht

```

This new bit uses `\@textmin` to locally store the amount of extra room in the column.

```

299 \latexrelease \@textmin \@colht
300 \latexrelease \advance \@textmin -\@dbltoproom

```

Floatpage bit: must be local.

```

301 \latexrelease \@fpmin \dblfloatpagefraction\textheight
302 \latexrelease \fptop \@dblfpsep
303 \latexrelease \fpsep \@dblfpsep
304 \latexrelease \fpbot \@dblfpbot
305 \latexrelease}%
306 \latexrelease\EndIncludeInRelease
307 \endkernel

```

(End of definition for `\@dblfloatplacement`.)

Historical *LaTeX* 2.09 comments (not necessarily accurate any more):

MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate with the `\output` routine. Marginal notes are distinguished from floats by having a negative placement specification. The command `\marginpar [LTEXT]{RTEXT}` generates a marginal note in a parbox, using LTEXT if it's on the left and RTEXT if it's on the right. (Default is RTEXT = LTEXT.) It uses the following parameters.

`\marginparwidth` : Width of marginal notes.
`\marginparsep` : Distance between marginal note and text.
the page layout to determine how to move the marginal
note into the margin. E.g., `\@leftmargin skip ==`
`\hskip -\marginparwidth \hskip -\marginparsep` .
`\marginparpush` : Minimum vertical separation between `\marginpar`'s

Marginal notes are normally put on the outside of the page if `@mparswitch = true`, and on the right if `@mparswitch = false`. The command `\reversemarginpar` reverses the side where they are put. `\normalmarginpar` undoes `\reversemarginpar`. These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of text, then the second one could appear on the next page, in a funny position.

```
\marginpar [LTEXT]{RTEXT} ==
BEGIN
  if hmode then \bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist has two elements:
    then get \@marbox, \@currbox from \@freelist
    \count\@marbox :=G -1
  else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
    \@currbox, \@marbox := \@tempboxa %%use \def
  fi
fi
if optional argument
  then %% \@xmpar ==
    \savemarbox\@marbox{LTEXT}
    \savemarbox\@currbox{RTEXT}
```



```

        else %% \@ympar ==
            \@savemarbox\@marbox{RTEXT}
            \box\@currbox :=G \box\@marbox
        fi
    \@xympar
END

\reversemarginpar == BEGIN \@mparbottom :=G 0
                        @reversemargin :=G true
                        END

\normalmarginpar == BEGIN \@mparbottom :=G 0
                        @reversemargin :=G false
                        END

```

End of historical L^AT_EX 2.09 comments.

\marginpar

```

308 \def\marginpar{%
309   \ifhmode
310     \@bsphack
311     \@floatpenalty -\@Mii
312   \else
313     \@floatpenalty-\@Miii
314   \fi
315   \ifinner
316     \@parmoderr
317     \@floatpenalty\z@
318   \else
319     \@next\@currbox\@freelist{}\{}%
320     \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
321     {\@floatpenalty\z@
322       \@fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
323   \fi
324   \@ifnextchar [\@xmpar\@ympar}

```

(End of definition for \marginpar.)

\@xmpar

```

325 \long\def\@xmpar[#1]#2{%
326   \@savemarbox\@marbox{#1}%
327   \@savemarbox\@currbox{#2}%
328   \@xympar}

```

(End of definition for \@xmpar.)

\@ympar

```

329 \long\def\@ympar#1{%
330   \@savemarbox\@marbox{#1}%
331   \global\setbox\@currbox\copy\@marbox
332   \@xympar}

```

(End of definition for \@ympar.)

`\@savemarbox`

```

333 </2ekernel>
334 <*2ekernel | latexrelease>
335 <latexrelease>\IncludeInRelease{2021/06/01}%
336 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
337 \long\def \@savemarbox #1#2{%
338   \global\setbox #1%
339     \color@vbox
340     \vtop{%
341       \hsize\marginparwidth
342       \@parboxrestore
343       \@marginparreset
344       #2\par
345       \@minipagefalse
346       \outer@nobreak
347     }%
348   \color@endbox
349 }
350 </2ekernel | latexrelease>
351 <latexrelease>\EndIncludeInRelease
352 <latexrelease>\IncludeInRelease{0000/00/00}%
353 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
354 <latexrelease>
355 <latexrelease>\long\def \@savemarbox #1#2{%
356 <latexrelease>  \global\setbox #1%
357 <latexrelease>    \color@vbox
358 <latexrelease>    \vtop{%
359 <latexrelease>      \hsize\marginparwidth
360 <latexrelease>      \@parboxrestore
361 <latexrelease>      \@marginparreset
362 <latexrelease>      #2%
363 <latexrelease>      \@minipagefalse
364 <latexrelease>      \outer@nobreak
365 <latexrelease>      }%
366 <latexrelease>    \color@endbox
367 <latexrelease>}
368 <latexrelease>\EndIncludeInRelease
369 <*2ekernel>

```

(End of definition for \@savemarbox.)

`\@marginparreset` The rationale for allowing these normally global flags to be set locally here, via `\@parboxrestore` was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```

370 \def \@marginparreset {%
371   \reset@font
372   \normalsize
373 %   \let@if@nobreak\iffalse
374 %   \let@if@noskipsec\iffalse

```

```

375 %          \@setnobreak
376          \@setminipage
377 }

```

(End of definition for \@marginparreset.)

\@xympar

Setting the box here is done only because the code uses \end@float; it will be empty and gets discarded.

```

378 \def \@xympar{%
379   \ifnum \@floatpenalty < \z@ \@cons \@currlist \@marbox \fi
380   \setbox \@tempboxa
381     \color@vbox
382     \vbox \bgroup
383   \end@float
384   \@ignorefalse
385   \@esphack
386 }

```

(End of definition for \@xympar.)

\reversemarginpar

\normalmarginpar

```

387 \def \reversemarginpar {\global \@parbottom \z@ \@reversemargintrue}
388 \def \normalmarginpar {\global \@parbottom \z@ \@reversemarginfalse}

```

(End of definition for \reversemarginpar and \normalmarginpar.)

```

389 \message{footnotes,}

```

1.2 Footnotes

Historical $\text{\LaTeX}$ 2.09 comments (not necessarily accurate any more):

\footnote{NOTE} : User command to insert a footnote.

\footnote[*NUM*]{NOTE}: User command to insert a footnote numbered *NUM*, where *NUM* is a number – 1, 2, etc. For example, if footnotes are numbered *, **, etc. within pages, then \footnote[2]{...} produces footnote '**'. This command does not step the footnote counter.

\footnotemark[*NUM*] : Command to produce just the footnote mark in the text, but no footnote. With no argument, it steps the footnote counter before generating the mark.

\footnotetext[*NUM*]{TEXT} : Command to produce the footnote but no mark. \footnote is equivalent to \footnotemark \footnotetext .

As in PLAIN, footnotes use \insert\footins, and the following parameters:

`\footnotesize` : Size-changing command for footnotes.

`\footnotesep` : The height of a strut placed at the beginning of every footnote.

`\skip\footins` : Space between main text and footnotes. The rule separating footnotes from text occurs in this space. This space lies above the strut of height `\footnotesep` which is at the beginning of the first footnote.

`\footnoterule` : Macro to draw the rule separating footnotes from text. It is executed right after a `\vspace` of `\skip\footins`. It should take zero vertical space—i.e., it should to a negative skip to compensate for any positive space it occupies. (See PLAIN.TEX.)

`\interfootnotelinepenalty` : Interline penalty for footnotes.

`\thefootnote` : In usual LaTeX style, produces the footnote number. If footnotes are to be numbered within pages, then the document style file must include an `\@addtoreset` command to cause the footnote counter to be reset when the page counter is stepped. This is not a good idea, though, because the counter will not always be reset in time to ensure that the first footnote on a page is footnote number one.

`\@thefnmark` : Holds the current footnote’s mark—e.g., `\dag` or `'1'` or `'a'`.

`\@mpfnnumber` : A macro that generates the numbers for `\footnote` and `\footnotemark` commands. It == `\thefootnote` outside a minipage environment, but can be changed inside to generate numbers for `\footnote`’s.

`\@makefnmark` : A macro to generate the footnote marker from `\@thefnmark`. The default definition was `\hbox{$\@thefnmark$}`.

This is now replaced by
`\textsuperscript{\@thefnmark}`

`\@makefntext{NOTE}` :
 Must produce the actual footnote, using `\@thefnmark` as the mark of the footnote and `NOTE` as the text. It is called when effectively inside a `\parbox`, with `\hsize = \columnwidth`.
 For example, it might be as simple as
`$\@thefnmark$ NOTE`

In a minipage environment, `\footnote` and `\footnotetext` are redefined so that

- (a) they use the counter `mpfootnote`
- (b) the footnotes they produce go at the bottom of the minipage.

The switch is accomplished by letting `\@mpfn == footnote` or `mpfootnote` and `\thempfn == \thefootnote` or `\thempfootnote`, and by redefining `\@footnotetext` to be `\@mpfootnotetext` in the minipage.

```

\footnote{NOTE} ==
BEGIN
  \stepcounter{\@mpfn}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnote[NUM]{NOTE} ==
BEGIN
  begingroup
    \protect == \noexpand
    counter \@mpfn :=L NUM
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnotemark ==
BEGIN \stepcounter{footnote}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

\footnotemark[NUM] ==
BEGIN
  begingroup
    footnote counter :=L NUM
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

```

```

\@footnotemark ==
  BEGIN
    \leavevmode
    IF hmode THEN \@x@sf := \the\spacefactor FI
    \@makefnmark      % put number in main text
    IF hmode THEN \spacefactor := \@x@sf FI
  END

\footnotetext      ==
  BEGIN begingroup \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

\footnotetext[ NUM ] ==
  BEGIN begingroup counter \@mpfn :=L NUM
    \protect == \noexpand
    \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

```

End of historical L^AT_EX 2.09 comments.

\footins L^AT_EX does use the same insert for footnotes as PLAIN.

```

390 \newinsert\footins

```

L^AT_EX leaves these initializations for the \footins insert.

```

391 \skip\footins=\bigskipamount % space added when footnote is present
392 \count\footins=1000 % footnote magnification factor (1 to 1)
393 \dimen\footins=8in % maximum footnotes per page

```

(End of definition for \footins.)

\footnoterule L^AT_EX keeps PLAIN T_EX's \footnoterule as the default.

```

394 \def\footnoterule{\kern-3\p@
395   \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high

```

(End of definition for \footnoterule.)

\thefootnote

```

396 \@definecounter{footnote}
397 \def\thefootnote{\@arabic\c@footnote}

```

(End of definition for \thefootnote.)

\thempfootnote The default display for the footnote counter in minipages is to use italic letters. We use \itshape not \textit as the latter would add an italic correction.

```

398 \@definecounter{mpfootnote}
399 \def\thempfootnote{{\itshape\@alph\c@mpfootnote}}

```

(End of definition for \thempfootnote.)

`\@makefnmark` Default definition.

```
400 %\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
401 \def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

(End of definition for \@makefnmark.)

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
402 \DeclareRobustCommand*\textsuperscript[1]{%
403   \@textsuperscript{\selectfont#1}}
```

(End of definition for \textsuperscript.)

`\@textsuperscript` This command should not be used directly, but may be used to define other commands
`\textsuperscript@offset` `\textsuperscript`, `\@makefnmark`. #1 should always start with a font selection com-
`\textsuperscript@space` mand, to activate the font size switch. Doing everything in text mode using the classical dimensions means using the settings from math mode: to allow the user to deviate from those, we use a one-level indirection. Notice that the script space is applied without using a kern to keep the tracing output as similar as possible to the previous math-based approach. The rather involved default for `\textsuperscript@offset` arises so we get the same result as from the math mode predecessor. The one for `\textsuperscript@space` is needed as in math mode, `\scriptspace` is not dropped at a line break (as it's a box adjustment). LuaTeX has rather easier to understand names for the various parameters, and these will be correct whether classical or OpenType fonts are in use for math, so we set up with these if available.

```
404 \if2ekernel
405 \iflatexrelease\IncludeInRelease{2026/06/01}%
406 \iflatexrelease\@textsuperscript{real text superscript}%
407 \if*2ekernel\iflatexrelease
408 \protected\def\@textsuperscript#1%
409   {%
410     \leavevmode
411     \begingroup
412       \check@mathfonts
413       \sbox\z@{\fontsize\sf@size\sf@size#1}%
414       \raise\dimexpr\textsuperscript@offset\relax\box\z@
415       \textsuperscript@space
416     \endgroup
417   }
418 \ifdefined\directlua
419   \def\textsuperscript@offset{%
420     \ifdim\Umathsupshiftup\textstyle<\dimexpr\dp\z@+\Umathsupbottommin\textstyle\relax
421       \dp\z@+\Umathsupbottommin\textstyle
422     \else
423       \Umathsupshiftup\textstyle
424     \fi
425   }
426 \else
427   \def\textsuperscript@offset{%
428     \ifdim\fontdimen14\textfont\tw@<\dimexpr\dp\z@+0.25\fontdimen5\textfont\tw@\relax
429       \dp\z@+0.25\fontdimen5\textfont\tw@
430     \else
431       \fontdimen14\textfont\tw@
```

```

432 \fi
433 }
434 \fi
435 \def\textsuperscript@space{\nobreak\hskip\scriptspace\kern\z@}
436 \</2ekernel | latexrelease>
437 \<latexrelease>\EndIncludeInRelease

438 \<latexrelease>\IncludeInRelease{2020/10/01}%
439 \<latexrelease> \<@textsuperscript>\{real text superscript\}%
440 \<latexrelease>\def\<@textsuperscript#1{%
441 \<latexrelease> \<m@th\ensuremath{\sim\{\mbox{\fontsize\sf@size\sf@size#1}\}}\}%
442 \<latexrelease>\EndIncludeInRelease

443 \<latexrelease>\IncludeInRelease{0000/00/00}%
444 \<latexrelease> \<@textsuperscript>\{real text superscript\}%
445 \<latexrelease>
446 \<latexrelease>\def\<@textsuperscript#1{%
447 \<latexrelease> \<m@th\ensuremath{\sim\{\mbox{\fontsize\sf@size\z@#1}\}}\}%
448 \<latexrelease>\EndIncludeInRelease
449 \<*2ekernel>

```

(End of definition for \<@textsuperscript, \<textsuperscript@offset, and \<textsuperscript@space.)

\<textsubscript

```

450 \</2ekernel>
451 \<latexrelease>\IncludeInRelease{2015/01/01}%
452 \<latexrelease> \<\textsubscript>\{\textsubscript\}%
453 \<*2ekernel | latexrelease>

454 \DeclareRobustCommand*\<textsubscript[1]{%
455 \<\textsubscript{\selectfont#1}\}%

456 \</2ekernel | latexrelease>
457 \<latexrelease>\EndIncludeInRelease
458 \<latexrelease>\IncludeInRelease{0000/00/00}%
459 \<latexrelease> \<\textsubscript>\{\textsubscript\}%
460 \<latexrelease>\let\<textsubscript\<undefined
461 \<latexrelease>\EndIncludeInRelease
462 \<*2ekernel>

```

(End of definition for \<textsubscript.)

\<@textsubscript

\<textsubscript@offset
\<textsubscript@space

```

463 \</2ekernel>
464 \<latexrelease>\IncludeInRelease{2026/06/01}%
465 \<latexrelease> \<\<@textsubscript>\{real text subscript\}%
466 \<*2ekernel | latexrelease>
467 \protected\def\<@textsubscript#1%
468 {%
469 \leavevmode
470 \begingroup
471 \check@mathfonts
472 \sbox\z@{\fontsize\sf@size\sf@size#1}%
473 \lower\dimexpr\<textsubscript@offset\relax\box\z@
474 \<textsubscript@space
475 \endgroup
476 }

```



```

477 \ifdefined\directlua
478   \def\textsubscript@offset{%
479     \ifdim\Umathsubshiftdown\textstyle<\dimexpr\ht\z@-\Umathsubtopmax\textstyle\relax
480       \ht\z@-\Umathsubtopmax\textstyle
481     \else
482       \Umathsubshiftdown\textstyle
483     \fi
484   }
485 \else
486   \def\textsubscript@offset{%
487     \ifdim\fontdimen16\textfont\tw@<\dimexpr\ht\z@-0.8\fontdimen5\textfont\tw@\relax
488       \ht\z@-0.8\fontdimen5\textfont\tw@
489     \else
490       \fontdimen16\textfont\tw@
491     \fi
492   }
493 \fi
494 \def\textsubscript@space{\nobreak\hskip\scriptspace\kern\z@}
495 \if2ekernel | latexrelease)
496 \latexrelease)\EndIncludeInRelease
497 \latexrelease)\IncludeInRelease{2020/10/01}%
498 \latexrelease)      {\@textsubscript}{real text subscript}%
499 \latexrelease)\def\@textsubscript#1{%
500 \latexrelease)  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\sf@size#1}}}}
501 \latexrelease)\EndIncludeInRelease
502 \latexrelease)\IncludeInRelease{2015/01/01}%
503 \latexrelease)      {\@textsubscript}{real text subscript}%
504 \latexrelease)
505 \latexrelease)\def\@textsubscript#1{%
506 \latexrelease)  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}
507 \latexrelease)\EndIncludeInRelease
508 \latexrelease)\IncludeInRelease{0000/00/00}%
509 \latexrelease)      {\@textsubscript}{real text subscript}%
510 \latexrelease)\let\@textsubscript\@undefined
511 \latexrelease)\EndIncludeInRelease
512 \if2ekernel)

```

(End of definition for \@textsubscript, \textsubscript@offset, and \textsubscript@space.)

\footnotesep

```

513 \newdimen\footnotesep

```

(End of definition for \footnotesep.)

\footnote

```

514 \def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
515   \protected@xdef\@thefnmark{\thempfn}%
516   \@footnotemark\@footnotetext}}

```

(End of definition for \footnote.)

\@xfootnote

```
517 \def\@xfootnote[#1]{%
518   \begingroup
519     \csname c@\mpfn\endcsname #1\relax
520     \unrestored@protected@xdef\@thefnmark{\thempfn}%
521   \endgroup
522   \@footnotemark\@footnotetext}
```

(End of definition for \@xfootnote.)

\@footnotetext

```
523 </2ekernel>
524 <*2ekernel | latexrelease>
525 <latexrelease>\IncludeInRelease{2021/11/15}%
526 <latexrelease>          {\@footnotetext}{footnotetext tagging}%
527 \long\def\@footnotetext#1{\insert\footins{%
528   \reset@font\footnotesize
529   \interlinepenalty\interfootnotelinepenalty
530   \splittopskip\footnotesep
531   \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
532   \hsize\columnwidth \@parboxrestore
533   \def\@currentcounter{footnote}%
534   \protected@edef\@currentlabel{%
535     \csname p@footnote\endcsname\@thefnmark
536   }%
537   \color@begingroup
538     \@makefntext{%
539       \rule{\z@}{\footnotesep}\ignorespaces#1\@finalstrut\strutbox}%
540   \par
541   \color@endgroup}}}%
542 </2ekernel | latexrelease>
543 <latexrelease>\EndIncludeInRelease
544 <latexrelease>\IncludeInRelease{2021/06/01}%
545 <latexrelease>          {\@footnotetext}{footnotetext tagging}%
546 <latexrelease>\long\def\@footnotetext#1{\insert\footins{%
547 <latexrelease>    \reset@font\footnotesize
548 <latexrelease>    \interlinepenalty\interfootnotelinepenalty
549 <latexrelease>    \splittopskip\footnotesep
550 <latexrelease>    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
551 <latexrelease>    \hsize\columnwidth \@parboxrestore
552 <latexrelease>    \protected@edef\@currentlabel{%
553 <latexrelease>      \csname p@footnote\endcsname\@thefnmark
554 <latexrelease>    }%
555 <latexrelease>    \color@begingroup
556 <latexrelease>      \@makefntext{%
557 <latexrelease>        \rule{\z@}{\footnotesep}\ignorespaces#1\@finalstrut\strutbox}%
558 <latexrelease>    \par
559 <latexrelease>    \color@endgroup}}}%
560 <latexrelease>\EndIncludeInRelease
561 <latexrelease>\IncludeInRelease{0000/00/00}%
562 <latexrelease>          {\@footnotetext}{footnotetext tagging}%
563 <latexrelease>
564 <latexrelease>\long\def\@footnotetext#1{\insert\footins{%
```

```

565 <latexrelease> \reset@font\footnotesize
566 <latexrelease> \interlinepenalty\interfootnotelinepenalty
567 <latexrelease> \splittopskip\footnotesep
568 <latexrelease> \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
569 <latexrelease> \hsize\columnwidth \@parboxrestore
570 <latexrelease> \protected@edef\@currentlabel{%
571 <latexrelease> \csname p@footnote\endcsname\@thefnmark
572 <latexrelease> }%
573 <latexrelease> \color@begingroup
574 <latexrelease> \@makefnmark{%
575 <latexrelease> \rule{z@footnotesep\ignorespaces#1\@finalstrut\strutbox}%
576 <latexrelease> \color@endgroup}}%
577 <latexrelease>
578 <latexrelease>\EndIncludeInRelease
579 <*2ekernel>

```

(End of definition for \footnotetext.)

\footnotemark

```

580 \def\footnotemark{%
581 \ifnextchar[\@xfootnotemark
582 {\stepcounter{footnote}}%
583 \protected@xdef\@thefnmark{\thefootnote}%
584 \@footnotemark}}

```

(End of definition for \footnotemark.)

\@xfootnotemark

```

585 \def\@xfootnotemark[#1]{%
586 \begingroup
587 \c@footnote #1\relax
588 \unrestored@protected@xdef\@thefnmark{\thefootnote}%
589 \endgroup
590 \@footnotemark}

```

(End of definition for \@xfootnotemark.)

\@footnotemark

```

591 \def\@footnotemark{%
592 \leavevmode
593 \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
594 \@makefnmark
595 \ifhmode\spacefactor\@x@sf\fi
596 \relax}

```

(End of definition for \@footnotemark.)

\footnotetext

```

597 \def\footnotetext{%
598 \ifnextchar [\@xfootnotenext
599 {\protected@xdef\@thefnmark{\thempfn}%
600 \@footnotetext}}

```

(End of definition for \footnotetext.)

`\@xfootnotenext`

```

601 \def\@xfootnotenext[#1]{%
602   \begingroup
603     \csname c@\@mpfn\endcsname #1\relax
604     \unrestored@protected@xdef\@thefnmark{\thempfn}%
605   \endgroup
606   \@footnotetext}

```

(End of definition for \@xfootnotenext.)

`\thempfn`

`\@mpfn`

```

607 \def\@mpfn{footnote}
608 \def\thempfn{\thefootnote}

```

(End of definition for \thempfn and \@mpfn.)

`\footref` This command generates a footnote mark. The value is produced by referencing a `\label` placed into a `\footnote` elsewhere (can be one in the main galley or in a minipage).

```

609 \</2ekernel>
610 \< *2ekernel | latexrelease>
611 \< latexrelease> \IncludeInRelease{2021/06/01}%
612 \< latexrelease> {\footref}{Add footref}%
613 \def\footref#1{%
614   \begingroup
615     \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
616   \endgroup
617   \@footnotemark
618 }
619 \< /2ekernel | latexrelease>
620 \< latexrelease> \EndIncludeInRelease

```

We don't remove it when rolling back so that packages offered it in the past do not need to alter their behavior in a rollback situation.

```

621 \< latexrelease> \IncludeInRelease{0000/00/00}%
622 \< latexrelease> {\footref}{Add footref}%
623 \< latexrelease>
624 \< latexrelease> % \let\footref\@undefined
625 \< latexrelease>
626 \< latexrelease> \EndIncludeInRelease
627 \< *2ekernel>

```

(End of definition for \footref.)

```

628 \< /2ekernel>

```

File 46

ltxglo.dtx

1 Index and Glossary Generation

Index and Glossary commands.

```
\makeindex      A preamble command to turn on indexing.
\makeglossary    A preamble command to turn on making glossary entries.
  \index         Make an index entry for #1.
  \glossary      Make a glossary entry for #1.
Historical LATEX 2.09 comments (not necessarily accurate any more):
\makeindex ==
  BEGIN
    \index == BEGIN \@bsphack
              \begingroup
              \protect{X} == \string X\space
              %% added 3 Feb 87 for \index commands
              %% in \footnotes
              re-\catcode special characters
              to 'other'
              \@wrindex

  END

\@wrindex{ITEM} ==
  BEGIN
    write of {\indexentry{ITEM}{page number}}
  \endgroup
  \@esphack
  END

INITIALIZATION:

\index == BEGIN \@bsphack
          \begingroup
          re-\catcode special characters (in case '%' there)
          \@index

  END

\@index{ITEM} == BEGIN \endgroup \@esphack END
```

Changes made 14 Apr 89 to write \glossaryentry's instead of
\indexentry's on the .glo file.

End of historical L^AT_EX 2.09 comments.

```
1 <*2ekernel>
2 \message{index,}
```

`\makeindex`

```
3 \def\makeindex{%  
4   \newwrite\@indexfile  
5   \immediate\openout\@indexfile=\jobname.idx  
6   \def\index{\@bsphack\beginingroup  
7     \@sanitize  
8     \@wrindex}\typeout  
9     {Writing index file \jobname.idx}}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
10   \let\makeindex\@empty  
11 }  
12 \@onlypreamble\makeindex
```

(End of definition for \makeindex.)

`\@wrindex`

```
13 \def\@wrindex#1{%  
14   \protected@write\@indexfile{}%  
15     {\string\indexentry{#1}{\thepage}}%  
16   \endgroup  
17   \@esphack}
```

(End of definition for \@wrindex.)

`\index`

```
18 \def\index{\@bsphack\beginingroup \@sanitize\@index}
```

(End of definition for \index.)

`\@index`

```
19 \def\@index#1{\endgroup\@esphack}
```

(End of definition for \@index.)

`\makeglossary`

```
20 \def\makeglossary{%  
21   \newwrite\@glossaryfile  
22   \immediate\openout\@glossaryfile=\jobname.glo  
23   \def\glossary{\@bsphack\beginingroup  
24     \@sanitize  
25     \@wrglossary}\typeout  
26     {Writing glossary file \jobname.glo }}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
27   \let\makeglossary\@empty  
28 }  
29 \@onlypreamble\makeglossary
```

(End of definition for \makeglossary.)

`\@wrglossary`

```
30 \def\@wrglossary#1{%  
31   \protected@write\@glossaryfile{%  
32     {\string\glossaryentry{#1}{\thepage}}}%  
33   \endgroup  
34   \@esphack}
```

(End of definition for \@wrglossary.)

`\glossary`

```
35 \def\glossary{\@bsphack\begin@group\@sanitizel\@index}
```

(End of definition for \glossary.)

```
36 \</2kernel>
```

File 47

ltbibl.dtx

1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as “References”, and a list of entries. The `BIBTEX` program will create a file containing such an environment, which will be read in by the `\bibliography` command. With `BIBTEX`, the following commands will be used.

`\bibliography` `\bibliography{<file1,file2, ...,filen>}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle` `\bibliographystyle{<style>}` : Writes a `\bibstyle` entry on the `.aux` file.

`thebibliography (env.)` The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:

`\bibitem{<name>}` : Produces a numbered entry cited as `<name>`.

`\bibitem[<label>]{<name>}` : Produces an entry labeled by `<Label>` and cited by `<name>`.

 The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

 The document class must define the `thebibliography` environment. This environment has a single argument, which is the widest bibliography label— e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{<name>}`.

`\nocite` `\nocite{<citations>}` puts information on the `.aux` file that causes `BIBTEX` to include the `{<citations>}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells `BIBTEX` to put the whole of a collection of references into the bibliography.

```
1 <*2ekernel>
2 \message{bibliography,}
```

Historical $\LaTeX$ 2.09 comments (not necessarily accurate any more):

PARAMETERS

`@cite` : A macro such that `\@cite{LABEL1,LABEL2}{NOTE}` produces the output for a `\cite[NOTE]{FOO1,FOO2}` command, where entry `FOOi` is defined by `\bibitem[LABELi]{FOOi}`. The switch `@tempswa` is true if the optional `NOTE` argument is present. The default definition is :

```
\@cite{LABELS}{NOTE} ==
  BEGIN [LABELS
    IF @tempswa = T THEN , NOTE FI
  ]
END
```

`@biblabel` : A macro to produce the label in the bibliography entry. For `\bibitem[LABEL]{NAME}`, the label is

generated by \@biblabel{LABEL}. It has the default definition \@biblabel{LABEL} -> [LABEL].

CONVENTION

\b@F00 : The name or number of the reference created by \cite{FOO}
E.g., if \cite{FOO} -> [17] , then \b@F00 -> 17.

End of historical L^AT_EX 2.09 comments.

\bibitem

```
3 \def\bibitem{\@ifnextchar[\@lbibitem\bibitem}
```

(End of definition for \bibitem.)

\@lbibitem

```
4 \def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
5     {\let\protect\noexpand
6      \immediate
7      \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

(End of definition for \@lbibitem.)

\@bibitem

```
8 \def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
9     {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

(End of definition for \@bibitem.)

\bibcite

```
10 \def\bibcite{\@newlabel\b@b}
```

(End of definition for \bibcite.)

\citation

```
11 \let\citation\@gobble
```

(End of definition for \citation.)

\cite

```
12 \if2ekernel
13 \if*2ekernel\latexrelease
14 \if\latexrelease\IncludeInRelease{2022/06/01}%
15 \if\latexrelease{\cite}{check for blank}%
16 \DeclareRobustCommand\cite{%
17     \@ifnextchar [{\@tempswatrue\@citex@checkblank}{\@tempswafalse\@citex@checkblank[]}]}
```

Due to the way \@for as used in \@citex behaves an empty argument to \cite did not produce any warning for a missing citation. So we now inject a command before calling \@citex that does the checking for us. It is not done in \@citex directly, because that command is altered by a number of packages/classes and this way it is more likely that the check survives.

```
18 \def\@citex@checkblank[#1]#2{%
19     \IfBlankTF {#2}%
20     {\@citex[#1]{\space}}%
21     {\@citex[#1]{#2}}%
22 }
23 \if2ekernel\latexrelease
```

```

24 <latexrelease>\EndIncludeInRelease
25 <latexrelease>\IncludeInRelease{0000/00/00}%
26 <latexrelease>          {\cite}{check for blank}%
27 <latexrelease>
28 <latexrelease>\DeclareRobustCommand\cite{%
29 <latexrelease> \ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}}
30 <latexrelease>\let\@citex@checkblank\@undefined
31 <latexrelease>
32 <latexrelease>\EndIncludeInRelease
33 <*2ekernel>

```

(End of definition for \cite.)

\@citex \penalty\@m added to definition of \@citex to allow a line break after the ‘,’ in citations like [Jones80,Smith77] (Added 23 Oct 86)
space added after the ‘,’ (21 Nov 87)

```

34 \def\@citex[#1]#2{\leavevmode
35   \let\@citea\@empty
36   \@cite{\@for\@citeb:=#2\do
37     {\@citea\def\@citea{,\penalty\@m\ }%
38     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}}%
39   \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi

```

Using \hbox instead of \mbox is fine because of the \leavevmode above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the \@cite hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

40   \ifundefined{b@\@citeb}{\hbox{\reset@font\bfseries ?}}%
41   \G@refundefinedtrue
42   \@latex@warning
43     {Citation ‘\@citeb’ on page \thepage \space undefined}}%
44   {\@cite@ofmt{\csname b@\@citeb\endcsname}}}{#1}}

```

(End of definition for \@citex.)

```

\@bibdata
\@bibstyle
45 \let\@bibdata=\@gobble
46 \let\@bibstyle=\@gobble

```

(End of definition for \bibdata and \bibstyle.)

```

\bibliography
47 \def\bibliography#1{%
48   \if@filesw
49     \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
50   \fi
51   \@input{\jobname.bbl}}

```

(End of definition for \bibliographystyle.)

\bibliographystyle

```
52 \def\bibliographystyle#1{%
53   \ifx\@begindocumenthook\@undefined\else
54     \expandafter\AtBeginDocument
55   \fi
56   {\if@filesw
57     \immediate\write\@auxout{\string\bibstyle{#1}}%
58   \fi}}
```

(End of definition for \bibliographystyle.)

\nocite (Added 14 Jun 85)

This puts information on the .aux file that causes BibTeX to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for \nocite like that for \citex, to get rid of leading spaces.

```
59 \</2ekernel>
60 \<2ekernel | latexrelease>
61 \<latexrelease>\IncludeInRelease{2021/06/01}%
62 \<latexrelease>          {\nocite}{Allow nocite in preamble}%
63 \def\nocite#1{\@bsphack
```

With the implementation designed already in L^AT_EX 2.09 the \nocite command will not work before \begin{document} since it tries to write to the .aux file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of L^AT_EX, missing all citations done with \nocite. Thus we do only generate an error message and leave the fix for a L^AT_EX 2_ε successor.]

Given that we are now a quarter century into using L^AT_EX 2_ε there is no good reason any more do limit ourself to 2.09 considerations. So we now simply delay the \nocite if it is issued in the preamble.

```
64   \ifx\@onlypreamble\document
```

Since we are after \begin{document} we can do the citations:

```
65   \@for\@citeb:=#1\do{%
66     \edef\@citeb{\expandafter\@firstofone\@citeb}%
67     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
68     \@ifundefined{b@\@citeb}{\G@refundefinedtrue
69       \@latex@warning{Citation ‘\@citeb’ undefined}}{}%
70   \else
```

But before \begin{document} we raised an error message in the past but as of 2021/05 not any longer.

```
71 %   \@latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the .aux file is open for writing and after \@preamblecmds was executed to change the above test. Therefore \AtBeginDocument would still be too early.

```
72   \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
73   \fi
```

```

74 \esphack}
75 </2ekernel | latexrelease>
76 <latexrelease>\EndIncludeInRelease

77 <latexrelease>\IncludeInRelease{0000/00/00}%
78 <latexrelease> \nocite}{Allow nocite in preamble}%
79 <latexrelease>
80 <latexrelease>\def\nocite#1{\@bsphack
81 <latexrelease> \ifx\@onlypreamble\document
82 <latexrelease> \for\citeb:=#1\do{%
83 <latexrelease> \edef\citeb{\expandafter\@firstofone\citeb}%
84 <latexrelease> \if@filesw\immediate\write\@auxout{\string\citation{\citeb}}\fi
85 <latexrelease> \ifundefined{b@\citeb}{\Grefundefinedtrue
86 <latexrelease> \@latex@warning{Citation ‘\citeb’ undefined}}{}}%
87 <latexrelease> \else
88 <latexrelease> \@latex@error{Cannot be used in preamble}\@eha
89 <latexrelease> \fi
90 <latexrelease> \esphack}
91 <latexrelease>
92 <latexrelease>\EndIncludeInRelease
93 <*2ekernel>

```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence `‘b@*’` to something other than `\relax`. As a result `\cite{*}` will not warn either (but that never worked with BibT_EX in the first place).

```

94 \expandafter\let\csname b@*\endcsname\@empty

```

(End of definition for \nocite.)

1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

```

\@cite
95 \def\@cite#1#2{[#1\if@tempswa , #2\fi]}

```

(End of definition for \@cite.)

`\@cite@ofmt` This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@\@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```

96 \let\@cite@ofmt\hbox

```

(End of definition for \@cite@ofmt.)

```

\@biblabel
97 \def\@biblabel#1{[#1]}
98 </2ekernel>

```

(End of definition for \@biblabel.)

File 48

ltmarks.dtx

Abstract

Marks are used to communicate information about the content of a page to the output routine. For example, in order to construct running headers, the output routine needs information about which section names are present on a page, and this information is passed to it through the mark system. However, marks may also be used for other purposes. This module provides a generalized mechanism for marks of independent classes.

1 Introduction

The $\text{T}_{\text{E}}\text{X}$ engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document. When the material for the current page is assembled in box 255, $\text{T}_{\text{E}}\text{X}$ scans for such marks and sets the commands `\topmark`, `\firstmark` and `\botmark`. The `\firstmark` receives the content of the first `\mark` seen in box 255 and `\botmark` the content of the last mark seen. The `\topmark` holds the content of the last mark seen on the previous page or more exactly the value of `\botmark` from the previous page. If there are no marks on the current page then all three are made equal to the `\botmark` from the previous page.

This mechanism works well for simple formats (such as plain $\text{T}_{\text{E}}\text{X}$) whose output routines are only called to generate pages. It fails, however, in $\text{\LaTeX}$ (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. In that case the output routine is called, determines where to place the float, alters the goal for assembling text material (if the float was added to the top or bottom region) and then it resumes collecting textual material.

As a result the `\botmark` gets updated and so `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Another problem for $\text{\LaTeX}$ was that it wanted to use several “independent” marks and in the early implementations of $\text{T}_{\text{E}}\text{X}$ there was only a single `\mark` command available. For that reason $\text{\LaTeX}$ implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

However, this extended mechanism (while supporting scenarios such as chapter/section marks) was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism overcomes both issues:

- It provides arbitrarily many, fully independent named marks, that can be allocated and, from that point onwards, used.
- It offers access for each such marks to retrieve its top, first, and bottom values separately.
- Furthermore, the mechanism is augmented to give access to marks in different “regions” which may not be just full pages.

2 Design-level and code-level interfaces

The interfaces are mainly meant for package developers, but they are usable (with appropriate care) also in the document preamble, for example, when setting up special running headers with `fancyhdr`, etc. They are therefore available both as CamelCase commands as well as commands for use in the L3 programming layer. Both are described together below.

<code>\NewMarkClass</code>	<code>\NewMarkClass {<class>}</code>
<code>\mark_new_class:n</code>	<code>\mark_new_class:n {<class>}</code>

Declares a new `<class>` of marks to be tracked by L^AT_EX. Each `<class>` must be declared before it is used.

Mark classes can only be declared before `\begin{document}`.

<code>\InsertMark</code>	<code>\InsertMark {<class>} {<text>}</code>
<code>\mark_insert:nn</code>	<code>\mark_insert:nn {<class>} {<text>}</code>

Adds a mark to the current galley for the `<class>`, containing the `<text>`.

It has no effect in places in which you can't place floats, e.g., a mark inside a box or inside a footnote never shows up anywhere.

If used in vertical mode it obeys L^AT_EX's internal `@nobreak` switch, i.e., it does not introduce a breakpoint if used after a heading. If used in horizontal mode it doesn't handle spacing (like, for example, `\index` or `\label` does, so it should be attached to material that is typeset.

<code>insertmark</code>	<code>\AddToHook {insertmark} {<code>}</code>
-------------------------	---

When marks are inserted, the mark content may need some special treatment, e.g., by default `\label`, `\index`, and `\glossary` do not expand at this time (but only later if and when the mark content is actually used. In order to allow packages to augment or alter this setup there is a public hook `insertmark` that is executed at this point. It runs in a group so local modification to commands are only applied to the `<text>` argument of `\InsertMark` or `\mark_insert:nn`.

<code>\TopMark</code>	<code>* \TopMark</code>	<code>[\langle region \rangle] {\langle class \rangle}</code>
<code>\FirstMark</code>	<code>* \FirstMark</code>	<code>[\langle region \rangle] {\langle class \rangle}</code>
<code>\LastMark</code>	<code>* \LastMark</code>	<code>[\langle region \rangle] {\langle class \rangle}</code>
<code>\mark_use_top:nn</code>	<code>* \mark_use_top:nn</code>	<code>{\langle region \rangle} {\langle class \rangle}</code>
<code>\mark_use_first:nn</code>	<code>* \mark_use_first:nn</code>	<code>{\langle region \rangle} {\langle class \rangle}</code>
<code>\mark_use_last:nn</code>	<code>* \mark_use_last:nn</code>	<code>{\langle region \rangle} {\langle class \rangle}</code>

These functions expand to the appropriate mark $\langle text \rangle$ for the given $\langle class \rangle$ in the specified $\langle region \rangle$. The default $\langle region \rangle$ in the design-level commands is `page`. Note that with the L3 layer commands there are no optional arguments, i.e., both arguments have to be provided.

T_EXhackers note: The result is returned within the `\unexpanded` primitive (`\exp_not:n`), which means that the $\langle text \rangle$ does not expand further when appearing in an `x`-type or `e`-type argument expansion.

The “first” and “last” marks are those seen first and last in the current region/page, respectively. The “top” mark is the last mark of the $\langle class \rangle$ seen in an earlier region, i.e., the $\langle text \rangle$ what would be “current” at the very top of the region.

Important!

The commands are only meaningful inside the output routine, in other places their result is (while not random) unpredictable due to the way L^AT_EX cuts text material into pages. There is, however, one exception: if you produce multiple columns using the `multicol` package, it is possible to retrieve mark values from the regions `first-column`, `last-column`, `mcol-1`, `mcol-2`,... directly after the environment has ended. This can, for example, be useful if a `multicols` has been used inside a box.

Currently, $\langle region \rangle$ is one of `page`, `previous-page`, `column`, `previous-column`, `first-column`, `last-column`, and `mcol-1` (first column in a `multicols`), `mcol-2` (second column in a `multicols`), up to `mcol-20` (twentieth column in a `multicols`). See section 2.2 for discussion of how these regions behave and how one can make use of them.

<code>\IfMarksEqualTF</code>	<code>* \IfMarksEqualTF</code>	<code>[\langle region \rangle] {\langle class \rangle} {\langle pos_1 \rangle} {\langle pos_2 \rangle} {\langle true \rangle} {\langle false \rangle}</code>
<code>\IfMarksEqualT</code>	<code>* \mark_if_eq:nnnnTF</code>	<code>{\langle region \rangle} {\langle class \rangle} {\langle pos_1 \rangle} {\langle pos_2 \rangle} {\langle true \rangle} {\langle false \rangle}</code>
<code>\IfMarksEqualF</code>	<code>* \mark_if_eq:nnnnnnTF</code>	<code>{\langle region_1 \rangle} {\langle class_1 \rangle} {\langle pos_1 \rangle}</code>
<code>\mark_if_eq:nnnnTF</code>	<code>*</code>	<code>{\langle region_2 \rangle} {\langle class_2 \rangle} {\langle pos_2 \rangle} {\langle true \rangle} {\langle false \rangle}</code>
<code>\mark_if_eq:nnnnnnTF</code>	<code>*</code>	

These conditionals allow you to compare the content of two marks and act based on the result. The commands work in an expansion context, if necessary.

It is quite common when programming with marks to need to interrogate conditions such as whether marks have appeared on a previous page, or if there are multiple marks present on the current page, and so on. The tests above allow for the construction of a variety of typical test scenarios, with three examples presented below.

The first two conditionals cover only the common scenarios. Both marks are picked up from the same $\langle region \rangle$ (by default `page`) and they have to be of the same $\langle class \rangle$.⁴¹ The $\langle pos_i \rangle$ argument can be either `top`, `first`, or `last`.

Important to note is that the comparison is not with respect to the textual content of the marks but whether or not they originated from the same `\InsertMark` command (or the L3 layer version `\mark_insert:nn`).

If you wish to compare marks across different regions or across different classes, you have to do it using the generic test only available in the L3 programming layer or do it manually, i.e., get the marks and then compare the values yourself.⁴²

⁴¹If an undeclared mark class is used the tests return `true` (not an error).

⁴²If two undeclared mark classes are compared the result is always `true`; if a declared and an undeclared

2.1 Use cases for conditionals

However, the basic version is enough for the following typical use cases:

Test for at most one mark of class `myclass` on current page: If the first and last mark in a region are the same then either there was no mark at all, or there was at most one. To test this on the current page:

```
\NewMarkClass{myclass}
\IfMarksEqualTF{myclass}{first}{last}
{ <zero or one mark> }{ <two or more marks> }
```

Test for no mark of class `myclass` in the previous page: If the top mark is the same as the first mark, there is no mark in the region at all. If we wanted to do this test for the previous page:

```
\IfMarksEqualTF[previous-page]{myclass}{top}{first}
{ <no marks> }{ <at least one mark> }
```

Comparing top and last would give you the same result.

Test for zero, one, or more than one: Combining the two tests from above you can test for zero, one or more than one mark.

```
\IfMarksEqualTF{myclass}{top}{first}
{ <no marks> }
{\IfMarksEqualTF{myclass}{first}{last}
{ <exactly one mark> }{ <more than one mark> }}
```

If you need one of such tests more often (or if you want a separate command for it for readability), then consider defining:

```
\providecommand\IfNoMarkTF[2][page]{\IfMarksEqualTF[#1]{#2}{first}{last}}
```

2.2 Understanding regions

If a page has just been finished then the region `page` refers to the current page and `previous-page`, as the name indicates, refers to the page before the current page. This means you are able to access mark information for the current page as well as for the page before (as long as you are inside the output routine) without the need to explicitly save that information beforehand. The `page` region is the region that is most often queried, which is why commands like `\FirstMark` use that region by default.

In single column documents the `column` is the same as the `page` region, but in two-column documents (if not produced by `multicols`), `column` refers to the current column that just got finished and `previous-column` to the one previously finished. Code for running headers is (in standard L^AT_EX) evaluated only after both columns have been assembled, which is another way of saying that in that case `previous-column` refers to the left column and `column` to the right column. However, to make these somewhat easier to use, there are also aliased names for these two regions: `first-column` and `last-column`.⁴³

mark class is used it is always *false*.

⁴³The region is called “last-column” not “second-column” in anticipation of extending the mechanism to multiple columns, where first and last would still make sense. There aren’t any `previous-first-column` and `previous-last-column` regions to access the corresponding columns from the previous page.

Note that you can only look backwards at already processed regions, e.g., in a `twoside` document finishing a recto (odd, right-hand) page you can access the data from the facing verso (left-hand) page, but if you are finishing a left-hand page you can't integrate data from the upcoming right-hand page. If such a scenario needs to be realized then it is necessary to save the left-hand page temporarily instead of finalizing it, process material for the right-hand page and once both are ready, attach running headers and footers and shipout out both in one go.⁴⁴

The situation starts getting rather complex if you allow for multiple columns in the way they are supported by the `multicol` package. In this case you might have a variable number of “columns” on a single page to be shipped out. And even if not, then a `multicols` might start or end in the middle of the page; in either case, the regions `column` and `previous-column` become rather meaningless and you should therefore not use them.⁴⁵ Instead, the algorithm offers `mcol-1`, `mcol-2`, `mcol-3`, etc., to represent the columns in the `multicols` on the current page to be shipped out. If there is more than one `multicols` on the current page then in the output routine only the columns of the last one will be accessible.

These provisions cover, out of the box, a number of layouts and use cases, but obviously not all. However, more cases can be supported by storing away mark information during the processing. Here is the full algorithm:

- The `column` region is used by the “current column” that is being built (moving through all columns with `previous-column` trailing behind (to handle top marks properly).
- When the `multicols` starts, the `column` region is cleared, i.e., from that point on it looks as if there have not been any marks so far. This will make sure that the top mark in the first column is always empty.
- If the `multicols` extends beyond the current page, then the material designated for the current page is split into columns. The `column` region is used to represent each column in turn.
 - First we copy the current data from `column` to `previous-column`. Then the mark data from the current column is placed into the `column` region. Then we alias `column` to `mcol-1`.
 - These steps are repeated for all columns of the `multicols` environment.
 - Finally, the first and the last column of that page is also made available as `first-column` and `last-column`, respectively.
- All those marks inside any of the columns are also available in the `page` region. Thus, if you are interested in the top, first, or last mark of a specific class on the whole page you simply need to query for it in the `page` region.
- If the `multicols` continues across several pages then this algorithm above is repeated for each page, except that the `column` region is not cleared again. This means that the top mark of the first column of the next page will be the last mark of the last column from the previous page.

⁴⁴As of now that scenario is not (yet) officially supported but it would be possible to achieve this using the shipout hooks to store the verso page and then on the next shipout use the hook to shipout both with running headers and footers attached.

⁴⁵They return something, because they represent the last two columns of the `multicols` when you are inside the output routine, but that is obviously of little use.

- When the `multicols` finishes the remaining material for the current page is balanced to produce columns of roughly equal height.
- Again `column` and `previous-column` are used while this balancing happens and `mcol-1`, `mcol-2`, etc., are used to represent the column regions and `first-column` and `last-column` are set appropriately.
- Then the balanced set of columns is returned back to the page (since there may be space for further material). In addition, all marks inside that material are reinserted so that they become available in the `page` region.
- As a side effect, it is possible (and useful in certain circumstances) to query for mark classes directly after the `multicols` has ended without the need to be inside the output routine. The regions that can be queried this way are `mcol-1`, `mcol-2`, etc. (up to the number of columns the multicol had) and `first-column` and `last-column`.

2.3 Debugging mark code

<code>\DebugMarksOn</code>	<code>\DebugMarksOn ... \DebugMarksOff</code>
<code>\DebugMarksOff</code>	
<code>\mark_debug_on:</code>	Commands to turn the debugging of mark code on or off. The debugging output is
<code>\mark_debug_off:</code>	rather coarse and not really intended for normal use at this point in time.

3 Application examples

If you want to figure out if a break was taken at a specific point, e.g., whether a heading appears at the top of the page, you can do something like this:

```
\newcounter{breakcounter}
\NewMarkClass{break}
\newcommand\markedbreak[1]{\stepcounter{breakcounter}%
                           \InsertMark{break}{\arabic{breakcounter}}%
                           \penalty #1\relax
                           \InsertMark{break}{-\arabic{breakcounter}}}
```

To test if the break was taken you can test if `\TopMark{break}` is positive (taken) or negative (not taken) or zero (there was never any marked break so far). The absolute value can be used to keep track of which break it was (with some further coding).

to be extended with additional application examples

4 Legacy L^AT_EX 2_ε interface

Here we describe the interfaces that L^AT_EX 2_ε offered since the early nineties and some minor extensions.

4.1 Legacy design-level and document-level interfaces

<code>\markboth</code>	<code>\markboth {<left>} {<right>}</code>
<code>\markright</code>	<code>\markright {<right>}</code>

L^AT_EX 2_ε uses two marks which aren't fully independent. A “left” mark generated by the first argument of `\markboth` and a “right” mark generated by the second argument of `\markboth` or by the only argument of `\markright`. The command `\markboth` and `\markright` are in turn called from heading commands such as `\chaptermark` or `\sectionmark` and their behavior is controlled by the document class.

For example, in the `article` class with `twoside` in force the `\sectionmark` will issue `\markboth` with an empty second argument and `\subsectionmark` will issue `\markright`. As a result the left mark will contain chapter titles and the right mark subsection titles.

Note, however, that in one-sided documents the standard behavior is that only `\markright` is used, i.e., there will only be right-marks but no left marks!

<code>\leftmark</code>	<code>* \leftmark</code>
<code>\rightmark</code>	<code>* \rightmark</code>

These functions return the appropriate mark value from the current page and work as before, that is `\leftmark` will get the last (!) left mark from the page and `\rightmark` the first (!) right mark.

In other words they work reasonably well if you want to show the section title that is current when you are about to turn the page and also show the first subsection title on the current page (or the last from the previous page if there wasn't one). Other combinations can't be shown using this interface.

The commands are fully expandable, because this is how they have been always defined in L^AT_EX. However, this is of course only true if the content of the mark they return is itself expandable and does not contain any fragile material. Given that this can't be guaranteed for arbitrary content, a programmer using them in this way should use `\protected@edef` and *not* `\edef` to avoid bad surprises as far as this is possible, or use the new interfaces (`\TopMark`, `\FirstMark`, and `\LastMark`) which return the `<text>` in `\exp_not:n` to prevent uncontrolled expansion.

4.2 Legacy interface extensions

The new implementation adds three mark classes: `2e-left`, `2e-right` and `2e-right-nonempty` and patches `\markboth` and `\markright` slightly so that they also update these new mark classes, so that the new classes work with existing document classes.

As a result you can use `\LastMark{2e-left}` and `\FirstMark{2e-right}` instead of `\leftmark` and `\rightmark`. But more importantly, you can use any of the other retrieval commands to get a different status value from those marks, e.g., `\LastMark{2e-right}` would return the last subsection on the page (instead of the first as returned by `\rightmark`).

The difference between `2e-right` and `2e-right-nonempty` is that the latter will only be updated if the material for the mark is not empty. Thus `\markboth{title}{}` as issued by, say, `\sectionmark`, sets a `2e-left` mark with `title` and a `2e-right` mark with the empty string but does not add a `2e-right-nonempty` mark.

Thus, if you have a section at the start of a page and you would ask for `\FirstMark{2e-right}` you would get an empty string even if there are subsections on that page. But `2e-right-nonempty` would then give you the first or last subsection

on that page. Of course, nothing is simple. If there are no subsections it would tell you the last subsection from an earlier page. We therefore need comparison tools, e.g., if top and first are identical you know that the value is bogus, i.e., a suitable implementation would be

```
\IfMarksEqualTF{2e-right-nonempty}{top}{first}
{ <appropriate action if there was no real mark> }
{\FirstMark{2e-right-nonempty}}
```

5 Notes on the mechanism

In contrast to vanilla $\text{\TeX}$, $\varepsilon\text{-}\text{\TeX}$ extends the mark system to allow multiple independent marks. However, it does not solve the `\topmark` problem which means that $\text{\LaTeX}$ still needs to manage marks almost independently of $\text{\TeX}$. The reason for this is that the more complex output routine used by $\text{\LaTeX}$ to handle floats (and related structures) means that `\topmark(s)` remain unreliable. Each time the output routine is fired up, $\text{\TeX}$ moves `\botmark` to `\topmark`, and while $\varepsilon\text{-}\text{\TeX}$ extends this to multiple registers the fundamental concept remains the same. That means that the state of marks needs to be tracked by $\text{\LaTeX}$ itself. An early implementation of this package used $\text{\TeX}$'s `\botmark` only to ensure the correct interaction with the output routine (this was before the $\varepsilon\text{-}\text{\TeX}$ mechanism was even available). However, other than in a prototype implementation for $\text{\LaTeX}3$, this package was never made public.

The new implementation now uses $\varepsilon\text{-}\text{\TeX}$'s marks as they have some advantages, because with them we can leave the mark text within the galley and only extract the marks during the output routine when we are finally shipping out a page or storing away a column for use in the next page. That means we do not have to maintain a global data structure that we have to keep in sync with informational marks in the galley but can rely on everything being in one place and thus manipulations (e.g. reordering of material) will take the marks with them without a need for updating a fragile linkage.

To allow for completely independent marks we use the following procedure:

- For every type of marks we allocate a mark class so that in the output routine $\text{\TeX}$ can calculate for each class the current top, first, and bottom mark independently. For this we use `\newmarks`, i.e., one marks register per class.
- As already mentioned firing up an output routine without shipping out a page means that $\text{\TeX}$'s top marks get wrong so it is impossible to rely on $\text{\TeX}$'s approach directly. What we do instead is to keep track of the real marks (for the last page or more generally last region) in some global variables.
- These variables are updated in the output routine at defined places, i.e., when we do real output processing but not if we use special output routines to do internal housekeeping.
- The trick we use to get correctly updated variables is the following: the material that contains new marks (for example the page to be shipped out) is stored in a box. We then use $\text{\TeX}$ primitive box splitting functions by splitting off the largest amount possible (which should be the whole box if nothing goes really wrong). While that seems a rather pointless thing to do, it has one important side effect: $\text{\TeX}$ sets up first and bottom marks for each mark class from the material it has split off. This way we get the first and last marks (if there have been any) from the material in the box.

- The top marks are simply the last marks from the previous page or region. And if there hasn't been a first or bottom mark in the box then the new top mark also becomes new first and last mark for that class.
- That mark data is then stored in global token lists for use during the output routine and legacy commands such as `\leftmark` or new commands such as `\TopMark` simply access the data stored in these token lists.

That's about it in a nutshell. Of course, there are some details to be taken care of—those are discussed in the implementation sections.

6 Public interfaces for packages such as **multicol**

The functions in this section are public so that packages can make use of them. However, this must be done with great care, e.g., `\mark_update_structure_from_material:nn` updates the global mark structure and can therefore be used only in places where such an update is meaningful, e.g., in special output routines. Elsewhere, a change to the mark structure would break the whole mechanism and querying the marks would return incorrect data.

<code>\mark_update_structure_from_material:nn</code>	<code>\mark_update_structure_from_material:nn {<region>} {<material with marks>}</code>
--	---

Helper function that inspects the marks inside the second argument and assigns new mark values based on that to the `<region>` given in the first argument. For this it first copies the mark structure from `<region>` to `previous-<region>` and then takes all last mark values currently in the region and makes them the new top mark values. Finally it assigns new first and last values for all mark classes based on what was found in the second argument.

As a consequence, the allowed values for `<region>` are `page` and `column` because only they have `previous-...` counterparts.

Another important aspect to keep in mind is that marks are recognized only if they appear on the top level, e.g., if we want to process material stored in boxes we need to put it unboxed (using `\unvcopy` etc.) into the second argument.

<code>\mark_copy_structure:nn</code>	<code>\mark_copy_structure:nn {<alias>} {<source>}</code>
--------------------------------------	---

Helper function that copies all mark values in the `<source>` region to `<alias>`, i.e., make the structures identical. Used to update the `previous-...` structures inside `\mark_update_structure_from_material:nn` and `first-column` and `last-column` structures inside the internal commands `\__mark_update_singlecol_structures:` or `\__mark__update_dbcol_structures:`.

<code>\mark_set_structure_to_err:n</code>	<code>\mark_set_structure_to_err:n {<region>}</code>
---	--

Helper function that sets all mark values in the `<region>` to an error message. This is currently used for `last-column` at times where using marks from it would be questionable/wrong, i.e., when we have just processed the first column in a two-column document.

```
\mark_clear_structure:n \mark_clear_structure:n {\region}
```

Helper function that sets all mark values in the $\langle \textit{region} \rangle$ to empty. This is currently used for `column` when a multicol environment starts; this is because it wouldn't make sense if the top mark in the first column returned the last mark from a previous multicol (which may have been much earlier, with intermediate material).

```
\mark_get_marks_for_reinsertion:nnn \mark_get_marks_for_reinsertion:nnn {\source}
                                     \token-list-var for collecting first marks
                                     \token-list-var for collecting last marks
```

Helper function for extracting marks that would otherwise get lost, for example when they are hidden inside a box. This helper does not update mark structures and can therefore be used outside the output routine as well.

It collects all the top-level marks from inside the $\langle \textit{source} \rangle$ and then adds suitable `\mark_insert:nn` commands to each of the two token lists. These token lists can then be executed at the right place to reinsert the marks, e.g., directly after the box. This is, for example, going to be used⁴⁶ by `multicol` when a short balanced `multicols` is returned to the galley for typesetting.

If the $\langle \textit{source} \rangle$ consists of a single vertical box (plus possibly followed by some glue but nothing else) then the box is unpacked and the top-level marks are collected from its content. However, if it is not a vertical box or there are other data then nothing is unpacked and you have to do the unpacking yourself to get at the marks inside.

It is quite likely that one only needs a single token list for returning the `\mark_insert:nn` statements. If that is the case this command may change to take only two arguments.

7 Internal functions for the standard output routine of L^AT_EX

The functions in this section are tied to the output routine and used in the interface to L^AT_EX 2_ε and perhaps at some later time within a new output routine for L^AT_EX. They are not (yet) meant for general use and are therefore made internal, even though we already use them in `multicol`. Internal means that `@@` automatically gets replaced in the code (and in the documentation) so we have to give it a suitable value.

1 $\langle @@=\textit{mark} \rangle$

```
\_mark_update_singlecol_structures: \_mark_update_singlecol_structures:
```

L^AT_EX 2_ε integration function in case we are doing single column layouts. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It is called as part of `\@opcol`.

```
\_mark_update_dbcol_structures: \_mark_update_singlecol_structures:
```

L^AT_EX 2_ε integration function mark used when we are doing double column documents. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It then does different post-processing depending on the start of the switch `\if@firstcolumn`. If we are in the second column it also has to update page marks, otherwise it only updates column marks. It too is called as part of `\@opcol`.

⁴⁶Probably not before 2025, though.

8 The Implementation

```

2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease>\NewModuleRelease{2022/06/01}{ltmarks}
5 <latexrelease>                {Marks-handling}

```

8.1 Allocating new mark classes

`\g__mark_classes_seq` A list holding all the mark classes that have been declared.

```

6 \seq_new:N \g__mark_classes_seq

```

`\mark_new_class:n` A mark class is created by initializing a number of data structures. First, we get a register number to refer to the mark class. The new mark class is then added to the `\g__mark_classes_seq` sequence to be able to easily loop over all classes. Finally a number of top-level global token lists are declared that hold various versions of the mark for access.

`\__mark_new_class:nn`

```

7 \cs_new_protected:Npn \mark_new_class:n #1
8 {
9   \seq_if_in:NnTF \g__mark_classes_seq {#1}
10     {
11       \msg_error:nnn { mark } { class-already-defined }
12       {#1}
13     }
14     { \__mark_new_class:nn {#1} }
15 }

```

This is only available in the preamble.

```

16 \@onlypreamble \mark_new_class:n

```

The internal command carries out the necessary allocations.

```

17 \cs_new_protected:Npn \__mark_new_class:nn #1
18 {
19   <*trace>
20   \__mark_debug:n { \iow_term:e { Marks:~new~mark:~#1~\msg_line_context: } }
21   </trace>

```

Use the L^AT_EX 2_ε interface for now as the L3 programming layer doesn't have one for marks yet.

```

22 \exp_args:Nc \newmarks {c__mark_class_ #1 _mark}

```

Remember the new class in the sequence.

```

23 \seq_gput_right:Nn \g__mark_classes_seq {#1}
24 \__mark_init_region:nn {page}{#1}

```

For the page region we also keep track of the previous-page.

```

25 \__mark_init_region:nn {previous-page}{#1}

```

Same game for column and previous-column

```

26 \__mark_init_region:nn {column}{#1}
27 \__mark_init_region:nn {previous-column}{#1}

```

But for columns we also allocate token lists for the alias regions `first-column` and `last-column`.

```
28 \__mark_init_region:nn {first-column}{#1}
29 \__mark_init_region:nn {last-column}{#1}
```

To support multiple columns produced by the `multicol` package, we preallocate twenty alias regions (since this is the number of columns that `multicol` supports as a maximum). They are filled by copying the current column into the appropriate `mcol-...`

```
30 %fmi \__mark_init_region:nn {mcol}{#1}
31 %fmi \__mark_init_region:nn {previous-mcol}{#1}
32 \__mark_init_region:nn {mcol-1}{#1}
33 \__mark_init_region:nn {mcol-2}{#1}
34 \__mark_init_region:nn {mcol-3}{#1}
35 \__mark_init_region:nn {mcol-4}{#1}
36 \__mark_init_region:nn {mcol-5}{#1}
37 \__mark_init_region:nn {mcol-6}{#1}
38 \__mark_init_region:nn {mcol-7}{#1}
39 \__mark_init_region:nn {mcol-8}{#1}
40 \__mark_init_region:nn {mcol-9}{#1}
41 \__mark_init_region:nn {mcol-10}{#1}
42 \__mark_init_region:nn {mcol-11}{#1}
43 \__mark_init_region:nn {mcol-12}{#1}
44 \__mark_init_region:nn {mcol-13}{#1}
45 \__mark_init_region:nn {mcol-14}{#1}
46 \__mark_init_region:nn {mcol-15}{#1}
47 \__mark_init_region:nn {mcol-16}{#1}
48 \__mark_init_region:nn {mcol-17}{#1}
49 \__mark_init_region:nn {mcol-18}{#1}
50 \__mark_init_region:nn {mcol-19}{#1}
51 \__mark_init_region:nn {mcol-20}{#1}
```

We also have to initialize the `saved-column` region that is used in `multicol`. Perhaps we should have a `\NewMarkRegion` so that it would be possible for other packages to add further regions. But let's wait and see if there is a real use case for new regions before making the interface more general.

```
52 \__mark_init_region:nn {saved-column}{#1}
53 }
```

(End of definition for `\mark_new_class:n` and `\__mark_new_class:nn`. This function is documented on page 1059.)

`\__mark_init_region:nn` For each class (#2) and region (#1), we need three token lists: one for top, first, and last.
`\c__mark_empty_tl` The default value to be returned is “empty”.

```
54 \cs_new_protected:Npn \__mark_init_region:nn #1 #2 {
55 \tl_new:c { g__mark_#1_top_ #2 _tl }
56 \tl_new:c { g__mark_#1_first_ #2 _tl }
57 \tl_new:c { g__mark_#1_last_ #2 _tl }
58 \tl_gset_eq:cN { g__mark_#1_top_ #2 _tl } \c__mark_empty_tl
59 \tl_gset_eq:cN { g__mark_#1_first_ #2 _tl } \c__mark_empty_tl
60 \tl_gset_eq:cN { g__mark_#1_last_ #2 _tl } \c__mark_empty_tl
61 }
```


All marks will have an identification in the form of a number⁴⁷ that is incremented each time a mark insertion happens; therefore the initial empty values should also have such a number, so that data extraction will be uniform.

```
62 \tl_const:Nn \c__mark_empty_tl { \__mark_value:nn{0}{} }
```

(End of definition for `\__mark_init_region:nn` and `\c__mark_empty_tl`.)

8.2 Updating mark structures

```
\l__mark_box
\l__mark_ii_box
\g__mark_tmp_tl
\g__mark_new_top_tl
```

For some operations we need two temporary private boxes and two private global token lists.

```
63 \box_new:N \l__mark_box
64 \box_new:N \l__mark_ii_box
65 \tl_new:N \g__mark_tmp_tl
66 \tl_new:N \g__mark_new_top_tl
```

(End of definition for `\l__mark_box` and others.)

```
\__mark_extract_and_handle_marks:nn
```

This is the main macro to extract and handle marks inside some vertical material. It is used by `\mark_update_structure_from_material:nn` (for updating the mark structure for a region based on the marks found) and by `\mark_get_marks_for_reinsertion:nnn` (for extracting marks from some material and prepare for reinserting them later (e.g., out of a box that is placed as a box into the main galley)).

```
67 \cs_new_protected:Npn \__mark_extract_and_handle_marks:nn #1#2 {
```

This macro expects code to handle extracted marks in its first argument and vertical material (not boxed or just consisting of a single vertical box) as its second. It extracts top-level mark information from `#2`, stores them as split marks and then calls `#1` to make use of this information.

If it finds a forced break in the material it removes it and then restarts the attempt without it.

We start with a group to keep most changes local.

```
68 \group_begin:
```

Getting the first and last marks out of the material in `#2` is done by putting the material in a box and then doing a split operation to the maximum size possible (which hopefully gets us all of the content).⁴⁸ Because this action is used only to get the mark values, we don't want any underfull box warnings so we (locally) turn those off.

```
69 \dim_set_eq:NN \tex_splitmaxdepth:D \c_max_dim
70 \int_set_eq:NN \tex_vbadness:D \c_max_int
71 \dim_set_eq:NN \tex_vfuzz:D \c_max_dim
```

There is a further complication: if the material contains infinite shrinking glue then a `\vsplit` operation will balk with a low-level error. Now pages or columns, which are our main concern here, can't have such infinite shrinkage if they are cut straight from the galley, however the use of `\enlargethispage` actually does add some at the very bottom (and also wraps the whole page into a box by itself, so if we leave it this way then a) we get this error and b) we don't see any marks because they are hidden one level down).

⁴⁷There are a few cases where special identification strings are used, e.g., `2.09-compat`.

⁴⁸With normal column material cut from the main galley we should always get all material in one go, but in certain situations, for example, in a `multicols` environment that contains some `\columnbreaks` a single split operation will not be enough. Thus, this is something we need to handle.

Another possible issue are packages or user code that place stray `\vboxes` directly into the main galley (an example is `marginnote` that attaches its marginals in this way). If such boxes end up as the last item on the page we should not unpack them.

All these issues need to be handled, which is done in `\__mark_prepare_and_extract:nn`.

```
72 \__mark_prepare_and_extract:nn {#1} {#2}
```

Once all mark classes have been processed, the data structures are updated and we can close the group, which undoes our local changes and retains only the global ones.

```
73 \group_end:
74 }
```

(End of definition for `\__mark_extract_and_handle_marks:nn`.)

`\__mark_prepare_and_extract:nn` This macro does the dirty work. It is not directly integrated in `\__mark_extract_and_handle_marks:nn` because we may have to call it recursively if we find forced breaks.

```
75 \cs_new_protected:Npn \__mark_prepare_and_extract:nn #1#2 {
```

To handle the `\enlargethispage` case we do an `\unskip` to get rid of any glue that is present at the very end of the material and also check if we have then a `\vbox` as the last item and if so unpack that too, but only under certain conditions, see below. All this is temporary done in a group, just for getting the marks out, so it doesn't affect the final page production.

```
76 \vbox_set:Nn \l__mark_box
77 {
78   #2
79   \tex_unskip:D
80   \box_set_to_last:N \l__mark_box
```

After having removed the last box from the current list (if there was one) we check whether the vertical list is now empty. If not, then the last box is definitely not the one from `\enlargethispage` and so we can, and should, leave it alone. Otherwise we check if this last box is a `\vbox`.

```
81 \int_compare:nNnT \tex_lastnodetype:D < 0
82 {
83   \box_if_vertical:NT \l__mark_box
```

If it is, we unpack the box.

```
84   { \vbox_unpack:N \l__mark_box }
85 }
```

If it wasn't a `\vbox`, it was either an `\hbox` or there was no box. Given that we are only interested in the marks we don't need put it back in that case.

```
86 }
```

We are now ready to `\vsplit` the box to get at the marks. If the box contains some infinite negative glue the `TEX` will produce an error complaining about it but it will correctly find the split marks. Given that we can't prevent that error, we hide it from the user and ensure that `TEX` doesn't stop. The error message still shows in the log, but even that is mitigated as best as possible—see the definition of `\__mark_vbox_set_split_to_maxdimen:NN` for the tricks employed.

```
87 \__mark_vbox_set_split_to_maxdimen:NN \l__mark_ii_box \l__mark_box
```

After splitting we check if there is anything left in `\l__mark_box`. If not then the above split has set some split marks that we can then use to finish the extraction:

```

88   \box_if_empty:NTF \l__mark_box
89   { #1 }

If we have a remainder after the split then this means that there was some forced break
in the material. We get rid of that by combining the content of the two boxes and restart.

90   {
91   (*trace)
92   \__mark_debug:n { \iow_term:e
93   { Marks:~ mark~ extraction~needs~ recursion~
94   \msg_line_context: } }
95   </trace>
96   \__mark_prepare_and_extract:nn {#1}
97   { \vbox_unpack:N \l__mark_ii_box
98   \vbox_unpack:N \l__mark_box }
99   }
100 }

```

(End of definition for `\__mark_prepare_and_extract:nn`.)

`\__mark_vbox_set_split_to_maxdimen:NN`

Split a box to get at its marks without pausing even if T_EX is producing an error message because of infinite negative glue in the box. If there is such an error we ensure that it only shows up in the log but not on the terminal.

With a recent T_EX engine that knows the primitive `\ignoreprimitiveerror` we can turn this error into a warning (simply by setting this primitive to the value 1).

```

101 \if_cs_exist:N \tex_ignoreprimitiveerror:D

102 \cs_new_protected:Npn \__mark_vbox_set_split_to_maxdimen:NN #1#2 {
103   \tl_set:Nc \l__mark_saved_parameters_tl
104   { \tex_ignoreprimitiveerror:D
105     \int_use:N \tex_ignoreprimitiveerror:D
106     \scan_stop:
107   }
108   \tex_ignoreprimitiveerror:D 1 \scan_stop:
109   \vbox_set_split_to_ht:NNn #1 #2 { \c_max_dim }
110   \l__mark_saved_parameters_tl
111 }

```

With older T_EX engines we have to make use of David's hack below to render the error (fairly) harmless and prevent T_EX from stopping. But, of course, the above solution is better because jumping over the error with a local change to the interaction mode still means that T_EX thinks there was an error in the run so the return code is no longer 0 (and that might affect workflows that want to test for this).

```

112 \else:

```

The nice low-level hack by DPC records in the `.log` that a glue shrinkage error is harmless.

We disguise `\c_max_dim` in an odd looking csname, which then shows up as part of the display of an error message if that error happens. This csname forms part of the error display so what you get is something like

```

! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
1. ... }

```

which hopefully makes it clear that the error is harmless and should be ignored by the reader of the .log.

```
113 \cs_set_eq:cN {\Infinite~shrink~error~above~ignored~!}\c_max_dim
```

The whole definition of `\__mark_vbox_set_split_to_maxdimen:NN` below is fully expanded, so we have to use a lot of `\exp_not:N` commands to prevent expansion where necessary.

```
114 \cs_new_protected:Npe \__mark_vbox_set_split_to_maxdimen:NN #1#2 {
```

We start by saving the current interaction and escape char settings.

```
115   \tl_set:Np \exp_not:N \l__mark_saved_parameters_tl
116   {
117     \tex_interactionmode:D
118     \exp_not:N \int_use:N \tex_interactionmode:D \scan_stop:
119     \tex_escapechar:D
120     \exp_not:N \int_use:N \tex_escapechar:D \scan_stop:
121   }
```

Then we change them so that no escape char is printed in the error message (accounts for the missing backslash in front of `Infinite shrink ...`) and we set the interaction to `\nonstopmode` so that the error (if any) just goes into the .log file and T_EX doesn't stop at that point.

```
122   \tex_escapechar:D      -1 \scan_stop:
123   \tex_interactionmode:D 0 \scan_stop:
```

Then we do the splitting of the box to `\c_max_dim` to get at the marks. This may generate the error we are worried about, i.e., if the box contains infinite negative glue. However, T_EX makes this glue finite and continues, which means we get our split marks which is really all we care about.

```
124   \tex_setbox:D #1 \tex_vsplit:D #2 to
```

The `\use:n` may seem pointless, and it is to some extent, but we need it to get our disguised `\c_max_dim` displayed properly as part of the error message if there is one. Without it, the display would show only part of what we want it to show (try it).

```
125     \exp_not:N \use:n {
126       \use:c{\Infinite~shrink~error~above~ignored~!}
127     }
```

Finally, we change the escape char and the interaction mode back to what it was before:

```
128   \exp_not:N \l__mark_saved_parameters_tl
129 }
130 \fi:
```

(End of definition for __mark_vbox_set_split_to_maxdimen:NN.)

`\l__mark_saved_parameters_tl` The temporary variable used for resetting escape char and interaction mode.

```
131 \tl_new:N \l__mark_saved_parameters_tl
```

(End of definition for \l__mark_saved_parameters_tl.)

`\mark_update_structure_from_material:nn` This function updates the mark structures of a region. The first argument is the region to update and second argument receives the material that holds the marks. Out of this material we extract the first and last marks for all classes (if there are any) to do the assignments.

```
132 \cs_new_protected:Npn \mark_update_structure_from_material:nn #1#2 {
133   \__mark_extract_and_handle_marks:nn
```

Once the marks can be extracted we update the structure from the split marks (code in `\_mark_update_structure_from_splitmarks:n`).

```

134     { \_mark_update_structure_from_splitmarks:n {#1} }
135     { #2 }
136 }

```

(End of definition for `\mark_update_structure_from_material:nn`. This function is documented on page 1066.)

This macro is called after we have done a `\tex_vsplit:D` operation and the mark data is in the split marks.

```

137 \cs_new_protected:Npn \_mark_update_structure_from_splitmarks:n #1 {

```

The first thing we do is to copy the current region structure to `previous-...`; this leaves the current structure untouched so we can update it class by class (as is necessary).

```

138   \mark_copy_structure:nn { previous-#1 } {#1}

```

After this action we can get first and last marks of the various classes through `\tex_splitfirstmarks:D` and `\tex_splitbotmarks:D`. So now we loop over all classes stored in `\g__mark_classes_seq`.

```

139   \seq_map_inline:Nn \g__mark_classes_seq
140   {

```

First action: get the last mark from the previous region, i.e., `previous-#1`. But because it is also still inside `#1`, at the moment we use that to construct the name because this is a tiny bit faster. Given that we need this value in various assignments we store it away which avoids unnecessary further csname generations.

```

141       \tl_gset_eq:Nc \g__mark_new_top_tl { g__mark_#1_last_##1_tl }

```

This will first of all become the new top mark for the current class.

```

142       \tl_gset_eq:cN { g__mark_#1_top_##1_tl } \g__mark_new_top_tl

```

Next action is to get ourselves the new last mark from the material supplied.

```

143       \tl_gset:Nc \g__mark_tmp_tl
144       { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }

```

If this mark doesn't exist then obviously neither does the first mark, so both become the last mark from the previous region. We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an “empty” mark that should not be confused with no mark at all. But no mark in our material will result in `\g__mark_tmp_tl` being fully empty. This is why we have to make sure that “empty” from `\mark_insert:nn` only appears to be empty when typeset but fails the next test (see below how this is done).

```

145       \tl_if_empty:NTF \g__mark_tmp_tl
146       {
147           \tl_gset_eq:cN { g__mark_#1_last_ ##1_tl }
148           \g__mark_new_top_tl
149           \tl_gset_eq:cN { g__mark_#1_first_##1_tl }
150           \g__mark_new_top_tl
151       }

```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead.

```

152       {
153           \tl_gset_eq:cN { g__mark_#1_last_##1_tl } \g__mark_tmp_tl

```

Because we had a last mark we also have a first mark (which might be the same, but might be not), so we pick that up and assign it to the appropriate token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```

154         \tl_gset:co { g__mark_#1_first_##1_tl }
155         {
156             \tex_splitfirstmarks:D
157             \use:c { c__mark_class_##1_mark }
158         }
159     }
160 }
161 }

```

(End of definition for `\__mark_update_structure_from_splitmarks:n`.)

`\mark_get_marks_for_reinsertion:nNN`

This function extracts the marks from the material in the first argument but it does not update any the mark structures. Instead, it collects the marks in the token lists given as the second and third argument, in such a way that they can be reinserted by just executing the token lists.⁴⁹

```

162 \cs_new_protected:Npn \mark_get_marks_for_reinsertion:nNN #1#2#3 {

```

First we clear the temporary token lists as we haven't seen any marks yet.

```

163   \tl_gclear:N \g__mark_first_marks_tl
164   \tl_gclear:N \g__mark_last_marks_tl

```

Then we extract all top-level marks, thereby filling the token lists with suitable `\mark_insert:nn` calls.

```

165   \__mark_extract_and_handle_marks:nn

```

The first argument holds the code used for filling the token lists and the second holds the material from which all marks should be extracted.

```

166   \__mark_get_from_splitmarks:
167   { #1 }

```

Finally, we copy the updated (or not updated) temporary token lists to the two that have been supplied when the function was called. By convention “get” operations return their values in local variables and `\__mark_extract_and_handle_marks:nn` runs in a group, which is why we have to use global temporary variables for collecting.

```

168   \tl_set_eq:NN #2 \g__mark_first_marks_tl
169   \tl_set_eq:NN #3 \g__mark_last_marks_tl
170 }

```

(End of definition for `\mark_get_marks_for_reinsertion:nNN`. This function is documented on page 1067.)

`\__mark_get_from_splitmarks:`

This function is called after we have done a `\vsplit` to update the split marks. It loops through all mark classes to find out if there are marks for this class and if so updates the global tls used for collecting.

```

171 \cs_new_protected:Npn \__mark_get_from_splitmarks: {
172   \seq_map_inline:Nn \g__mark_classes_seq
173   {

```

⁴⁹It is probably enough to collect everything in a single token list as long as we put the first marks first and the last marks last). But for extra flexibility, I currently use 2 token lists. This might change when it is really clear that this is never needed.

First we to get the last mark for the current class from the material supplied.

```

174 \tl_gset:No \g__mark_tmp_tl
175 { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }

```

If this mark doesn't exist then obviously first mark doesn't either, so we do nothing (other than issuing some debugging info).

We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an “empty” mark that we should not confuse with the case where there is no mark at all.

When there is no mark at all we get a truly empty `\g__mark_tmp_tl` as a result. This is why we have to make sure that an “empty” mark generated with `\mark_insert:nn` only appears to be empty when it is typeset, but fails the next test (see below how this is done).

```

176 \tl_if_empty:NTF \g__mark_tmp_tl
177 {
178   (*trace)
179   \__mark_debug:n { \iow_term:e { Marks:~no~ marks~
180     for~ class~ '##1'~\msg_line_context: } }
181   </trace>
182 }

```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead. This means we put an appropriate `\mark_insert:nn` statement into `\g__mark_last_marks_tl`.

```

183 {
184   (*trace)
185   \__mark_debug:n { \iow_term:e { Marks:~ extract~ last~

```

The mark content in `\g__mark_tmp_tl` may contain arbitrary code that may react badly if it is expanded in a write. So we better avoid that expansion, otherwise debugging might generate spurious errors when turned on.

```

186   mark~ for~ class~ '##1'~ =~ \exp_not:o \g__mark_tmp_tl } }
187 </trace>
188 \tl_gput_right:Ne \g__mark_last_marks_tl
189 { \mark_insert:nn {##1} { \__mark_drop_id:o { \g__mark_tmp_tl } } }

```

Because we had a last mark we also have a first mark (which might be the same, but might not be), so we pick that up and add it to the `\g__mark_first_marks_tl` token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```

190 (*trace)
191 \__mark_debug:n { \iow_term:e {
192   Marks:~ extract~ first~ mark~ for~ class~ '##1'~ =~

```

Again no expansion for the mark content.

```

193   \exp_not:o {
194     \tex_splitfirstmarks:D
195     \use:c { c__mark_class_##1_mark }
196   }
197 } }
198 </trace>
199 \tl_gput_right:Ne \g__mark_first_marks_tl
200 { \mark_insert:nn {##1}
201   {

```

We better drop the id from the returned value otherwise they will accumulate in the marks when reinserted.

```

202             \__mark_drop_id:o {
203                 \tex_splitfirstmarks:D
204                 \use:c { c__mark_class_##1_mark }
205             }
206         }
207     }
208 }
209 }
210 }

```

(End of definition for __mark_get_from_splitmarks:.)

\g__mark_first_marks_tl These are two global temporary variables used in the code above.
\g__mark_last_marks_tl

```

211 \tl_new:N \g__mark_first_marks_tl
212 \tl_new:N \g__mark_last_marks_tl

```

(End of definition for \g__mark_first_marks_tl and \g__mark_last_marks_tl.)

\mark_copy_structure:nn This function copies the structure for one region to another, e.g., from page to previous-page above, or later from column to first-column, etc.

```

213 \cs_new_protected:Npn \mark_copy_structure:nn #1#2 {

```

This requires a simple loop through all mark classes copying the token list from one name to the next.

```

214   \seq_map_inline:Nn \g__mark_classes_seq
215   {
216       \tl_gset_eq:cc { g__mark_ #1_top_   ##1_tl }
217                   { g__mark_ #2_top_   ##1_tl }
218       \tl_gset_eq:cc { g__mark_ #1_first_ ##1_tl }
219                   { g__mark_ #2_first_ ##1_tl }
220       \tl_gset_eq:cc { g__mark_ #1_last_  ##1_tl }
221                   { g__mark_ #2_last_  ##1_tl }
222   }
223 }

```

(End of definition for \mark_copy_structure:nn. This function is documented on page 1066.)

\mark_clear_structure:n This function sets the structure of one region back to an initial state, so that all classes return an empty value if queried.

```

224 \cs_new_protected:Npn \mark_clear_structure:n #1 {

```

This requires a simple loop through all mark classes.

```

225   \seq_map_inline:Nn \g__mark_classes_seq
226   {
227       \tl_gset_eq:cN { g__mark_ #1_top_   ##1_tl }
228                   \c__mark_empty_tl
229       \tl_gset_eq:cN { g__mark_ #1_first_ ##1_tl }
230                   \c__mark_empty_tl
231       \tl_gset_eq:cN { g__mark_ #1_last_  ##1_tl }
232                   \c__mark_empty_tl
233   }
234 }

```


(End of definition for `\mark_clear_structure:n`. This function is documented on page 1067.)

```
\mark_set_structure_to_err:n A slight variation is to install a fixed error message as the value.
  \__mark_error:n
235 \cs_new_protected:Npn \mark_set_structure_to_err:n #1 {
236   \seq_map_inline:Nn \g__mark_classes_seq
237     {
238     \tl_gset:ce { g__mark_ #1 _top_   ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
239     \tl_gset:ce { g__mark_ #1 _first_ ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
240     \tl_gset:ce { g__mark_ #1 _last_  ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
241     }
242 }
```

Given that this is used in only one place, we could hardwire the argument which would be a bit more compact, but who knows, perhaps we end up with another reason to use this error command elsewhere, so for now we keep the argument.

```
243 \cs_new_protected:Npn \__mark_error:nn #1#2 {
244   \msg_error:nnnn { mark } { invalid-use } {#1} {#2}
245 }
```

(End of definition for `\mark_set_structure_to_err:n` and `\__mark_error:n`. This function is documented on page 1066.)

8.3 Placing and retrieving marks

`\mark_insert:nn` This function puts a mark for some `<class>` at the current point.

```
246 \cs_new_protected:Npn \mark_insert:nn #1#2
247 {
248   \seq_if_in:NnTF \g__mark_classes_seq {#1}
249   {
```

We need to pass the evaluated argument into the mark but protected commands should not expand including those protected using the `\protect` approach of L^AT_EX 2_ε. We also disable `\label` and the like.⁵⁰

At this point the code eventually should get a public (and a kernel) hook instead of a set of hardwired settings.

```
250     \group_begin:
```

Within the group we alter some comments, e.g. `\label` or `\index`, to do the right at this point. This is done in the kernel hook `\@kernel@before@insertmark` which is followed by the public hook `insertmark` that can be used by packages to augment or alter that setup as necessary.

```
251     \@kernel@before@insertmark
252     \hook_use:n { insertmark }
253     \unrestored@protected@xdef \g__mark_tmp_tl
254     {
```

To ensure that marks are unique we insert a hidden sequence marker at the beginning of the content of the mark containing the sequence number of the mark.

```
255         \__mark_value:nn{ \int_use:N\g__mark_int }{#2}
256     }
257     <*trace>
258     \__mark_debug:n{ \iow_term:e { Marks:~ set~#1~<--
```

⁵⁰Straight copy from `latex.ltx` but is this even correct? At least a label in a running header makes little sense if it get set several times! Maybe that needs looking at in the 2e kernel.

```

259         '\tl_to_str:V \g__mark_tmp_tl' ~ \msg_line_context: } }
260 \trace>
261 \tex_marks:D \use:c { c__mark_class_ #1 _mark }
262 {

```

Here is the trick to avoid truly empty marks: if the result from the above processing is empty we add something which eventually becomes empty, but not immediately; otherwise we just put `\g__mark_tmp_tl` in.

```

263 % This is no longer needed with 1.0f
264 % \tl_if_empty:NTF \g__mark_tmp_tl
265 % { \exp_not:n { \prg_do_nothing: } }
266 % { \exp_not:o { \g__mark_tmp_tl } }
267 \exp_not:o { \g__mark_tmp_tl }
268 }
269 \group_end:

```

A mark introduces a possible break point and in certain situations that should not happen in vertical mode in L^AT_EX. This may need some checking and possibly cleanup

```

270 \if@nobreak\ifvmode\nobreak\fi\fi
271 }

```

If the mark class was not known, raise an error.

```

272 {
273 \msg_error:nne { mark } { unknown-class }
274 { \tl_to_str:n {#1} }
275 }
276 }

```

(End of definition for `\mark_insert:nn`. This function is documented on page 1059.)

`\__mark_value:nn` A hidden marker is placed into every mark added by `\mark_insert:nn`. It will not show up in the output but its argument (a counter value that is incremented) makes all marks unique so the test for “equal” is not fooled by two different marks having the same mark text.

```

277 \cs_new_protected:Npn \__mark_value:nn #1#2 { #2 }

```

(End of definition for `\__mark_value:nn`.)

`\@kernel@before@insertmark` By default `\label`, `\index`, and `\glossary` do nothing when the mark is inserted.

```

insertmark 278 \int_new:N \g__mark_int
279 \cs_new:Npn \@kernel@before@insertmark {
280 \cs_set_eq:NN \label \scan_stop:
281 \cs_set_eq:NN \index \scan_stop:
282 \cs_set_eq:NN \glossary \scan_stop:

```

We count each mark and use that to place a hidden marker in front of the mark text. To ensure that there is no overflow (very unlikely but you never know) we restart every 100000 marks. Thus, if somebody puts more than that number of marks on a single page you could construct a scenario in which that approach fails.

```

283 \int_compare:nNnTF \g__mark_int < {99999}
284 { \int_gincr:N \g__mark_int }
285 { \int_gzero:N \g__mark_int }
286
287 }

```

The public hook to augment the setup.

```
288 \hook_new:n {insertmark}
```

(End of definition for \@kernel@before@insertmark and insertmark.)

`\mark_use_top:nn` To retrieve the first, last or top region mark, we grab the appropriate value stored in the corresponding token list variable and pass its contents back. These functions should be used only in output routines and only after `\mark_update_structure_from_material:nn` has acted, otherwise their value will be wrong.

```
289 \cs_new:Npn \mark_use_first:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_first_#2_tl } {#1} {#2} }
290 \cs_new:Npn \mark_use_last:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_last_#2_tl } {#1} {#2} }
291 \cs_new:Npn \mark_use_top:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_top_#2_tl } {#1} {#2} }
```

If used with an unknown class or region these commands will generate an error. If that happens in an expandable context then the error generation is delayed (e.g., if used in a `\section`) and happens when the code is finally used in typesetting, e.g., in the TOC or a running header. If used in a `\typeout` you only see something like `\__mark_error:n{page}`. This is not too good, but probably better than low-level errors, I guess, and I don't want to use an expandable error because of the size restrictions in such error messages.

```
292 \cs_new:Npn \__mark_use_check:nnn #1#2#3 {
293   \tl_if_eq:cNTF {#1} \relax
294     { \__mark_error:nn {#2} {#3} }
295     { \__mark_drop_id:v {#1} }
296 }
```

Each mark starts with an id and while the id does not print it is nevertheless better to remove it when returning the mark, so that downstream manipulation of the data doesn't have to deal with it. This is what the `\exp_not:o` accomplishes.

```
297 \cs_new:Npn \__mark_drop_id:n #1 { \exp_not:o { #1 } }
298 \cs_generate_variant:Nn \__mark_drop_id:n { o, v }
```

(End of definition for `\mark_use_top:nn`, `\mark_use_first:nn`, and `\mark_use_last:nn`. These functions are documented on page 1060.)

8.4 Comparing mark values

`\mark_if_eq:nnnnTF` Test if in a given region (`#1`) for a given class (`#2`) the marks in position `#3` and `#4` (top, first, or last) are identical

```
299 \prg_new_conditional:Npnn \mark_if_eq:nnnn #1#2#3#4 { T , F , TF }
300 {
301   \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _tl }
302     { g__mark_ #1 _#4_ #2 _tl }
303     \prg_return_true:
304     \prg_return_false:
305 }
```

The fully general test (with two triplets of the form `<region>`, `<class>`, and `<position>`) is this:

```
306 \prg_new_conditional:Npnn \mark_if_eq:nnnnnn #1#2#3#4#5#6 { T , F , TF }
307 {
308   \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _tl }
309     { g__mark_ #4 _#6_ #5 _tl }
310     \prg_return_true:
311 }
```

```

311         \prg_return_false:
312     }

```

(End of definition for `\mark_if_eq:nnnnTF` and `\mark_if_eq:nnnnnnTF`. These functions are documented on page 1060.)

8.5 Messages

Mark errors are L^AT_EX kernel errors:

```

313 \prop_gput:Nnn \g_msg_module_type_prop { mark } { LaTeX }
314 \msg_new:nnnn { mark } { class-already-defined }
315   { Mark~class~'#1'~already~defined }
316   {
317     \c__msg_coding_error_text_tl
318     LaTeX~was~asked~to~define~a~new~mark~class~called~'#1':~
319     this~mark~class~already~exists.
320     \c__msg_return_text_tl
321   }
322 \msg_new:nnnn { mark } { unknown-class }
323   { Unknown~mark~class~'#1'. }
324   {
325     \c__msg_coding_error_text_tl
326     LaTeX~was~asked~to~manipulate~a~mark~of~class~'#1',~
327     but~this~class~of~marks~does~not~exist.
328   }

```

The next error can also happen if the mark class is unknown, so this should perhaps be separated into two different errors.

```

329 \msg_new:nnnn { mark } { invalid-use }
330   { Mark~region~'#1'~not~usable~or~class~'#2'~unknown }
331   {
332     \c__msg_coding_error_text_tl
333     The~region~'#1'~is~either~not~known~or~data~for~it~
334     still~needs~to~be~assembled,~e.g.,~last~column~
335     while~building~the~first~column.~
336     Also~possible:~the~class~name~'#2'~is~misspelled.
337     \c__msg_return_text_tl
338   }

```

8.6 Debugging the mark structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

```

\g__mark_debug_bool Holds the current debugging state.
339 \bool_new:N \g__mark_debug_bool

```

(End of definition for `\g__mark_debug_bool`.)

```

\mark_debug_on: Turns debugging on and off by redefining \__mark_debug:n.
\mark_debug_off:
\__mark_debug:n
\__mark_debug_gset:
340 \cs_new_eq:NN \__mark_debug:n \use_none:n
341 \cs_new_protected:Npn \mark_debug_on:
342 {
343   \bool_gset_true:N \g__mark_debug_bool
344   \__mark_debug_gset:
345 }
346 \cs_new_protected:Npn \mark_debug_off:
347 {
348   \bool_gset_false:N \g__mark_debug_bool
349   \__mark_debug_gset:
350 }
351 \cs_new_protected:Npn \__mark_debug_gset:
352 {
353   \cs_gset_protected:Npe \__mark_debug:n ##1
354   { \bool_if:NT \g__mark_debug_bool {##1} }
355 }

```

(End of definition for \mark_debug_on: and others. These functions are documented on page 1063.)

\DebugMarksOn CamelCase commands for debugging.

```

\DebugMarksOff
356 \cs_new_eq:NN \DebugMarksOn \mark_debug_on:
357 \cs_new_eq:NN \DebugMarksOff \mark_debug_off:

```

(End of definition for \DebugMarksOn and \DebugMarksOff. These functions are documented on page 1063.)

__mark_class_status:nnn Shows the mark values across all regions for one mark class (#2).

The first argument gives some *info* to help in identifying where the command was called, the second is the class and the third holds the number of mcol-... we should display: inside a multicols environment this will be \col@number, in L^AT_EX's normal output routines it will be 0.

```

358 ⟨*trace⟩
359 \cs_new_protected:Npn \__mark_class_status:nnn #1#2#3 {
360   \typeout{ Marks:~#2~ #1:}
361   \__mark_region_status:nnn {#2}{ page~ (previous) } { previous-page }
362   \__mark_region_status:nnn {#2}{ page~ (current)~ } { page }
363   \__mark_region_status:nnn {#2}{ column~ (previous) }{ previous-column }
364   \__mark_region_status:nnn {#2}{ column~ (current)~ }{ column }
365   \__mark_region_status:nnn {#2}{ column~ (first) } { first-column }
366   \__mark_region_status:nnn {#2}{ column~ (last)~ } { last-column }

```

Then finish by displaying a subset of the mcol-... regions: none (0) in the standard L^AT_EX output routine and \col@number within a multicols environment.

```

367   \int_step_inline:nn {#3}
368   {
369     \__mark_region_status:nnn {#2}{ column~ (##1)~ } { mcol-##1 }
370   }
371 }

```

(End of definition for __mark_class_status:nnn.)

`\__mark_region_status:nnn` Display the top, first, and last mark of a region unless none of them exist or all of them are empty.

```

372 \cs_new_protected:Npn \__mark_region_status:nnn #1#2#3 {
373   \group_begin:
374   \cs_set:Npn \__mark_value:nn ##1##2{ \exp_not:n{ {##1} ~ ##2 } }
375   \tl_if_exist:cT { g__mark_#3_last_ #1 _tl }
376   {
377     \tl_if_eq:cNF { g__mark_#3_last_ #1 _tl } \c__mark_empty_tl
378     {
379       \typeout{\@spaces #2 =
380         ~|~ \use:c { g__mark_#3_top_ #1 _tl } ~|~
381         \use:c { g__mark_#3_first_ #1 _tl } ~|~
382         \use:c { g__mark_#3_last_ #1 _tl } ~|
383       }
384     }
385   }
386   \group_end:
387 }

```

(End of definition for __mark_region_status:nnn.)

`\__mark_status:nn` Show a snapshot of all mark class values across all regions. The first argument is a string to identify the output, the second argument is the number of `mcol-...` regions to show. Outside of a `multicols` environment this is normally set to 0.

```

388 \cs_new_protected:Npn \__mark_status:nn #1#2
389 {
390   \seq_map_inline:Nn \g__mark_classes_seq
391   { \__mark_class_status:nnn {#1} {##1} {#2} }
392 }
393 </trace>

```

(End of definition for __mark_status:nn.)

`\ShowMarksAt` Debugging helper that displays a snapshot of all known mark structures. The first argument is a text string that is displayed to help identifying when the snapshot was made. The optional second one determines how many `mcol-...` regions are displayed (by default 4).

This may not stay like this (or at all), which is why it isn't yet documented as an official command.

```

394 \NewDocumentCommand \ShowMarksAt {m O{4}} { {
395   <*trace>
396   \__mark_debug:n { \__mark_status:nn {#1}{#2} }
397   </trace>
398 }

```

(End of definition for \ShowMarksAt.)

8.7 Designer-level interfaces

`\NewMarkClass` These two are identical to the L3 programming layer commands.

`\InsertMark`

```

399 \cs_new_eq:NN \NewMarkClass \mark_new_class:n
400 \@onlypreamble \NewMarkClass
401 \cs_new_eq:NN \InsertMark \mark_insert:nn

```

(End of definition for `\NewMarkClass` and `\InsertMark`. These functions are documented on page 1059.)

`\TopMark` `\FirstMark` `\LastMark` The following commands take an optional argument that defaults to page. There is no checking that the region is actually valid. If not there is simply an empty return.

```

402 \NewExpandableDocumentCommand \FirstMark { 0{page} m }
403     { \mark_use_first:nn {#1}{#2} }
404 \NewExpandableDocumentCommand \LastMark { 0{page} m }
405     { \mark_use_last:nn {#1}{#2} }
406 \NewExpandableDocumentCommand \TopMark { 0{page} m }
407     { \mark_use_top:nn {#1}{#2} }

```

(End of definition for `\TopMark`, `\FirstMark`, and `\LastMark`. These functions are documented on page 1060.)

`\IfMarksEqualTF` `\IfMarksEqualT` `\IfMarksEqualF` We only provide CamelCase commands for the case with one region (optional) and one class. One could think of also providing a version for the general case with several optional arguments, but use cases for this are most likely rare, so not done yet.

```

408 \NewExpandableDocumentCommand \IfMarksEqualTF {0{page}mmm} {
409     \mark_if_eq:nnnnTF {#1}{#2}{#3}{#4}
410 }
411 \NewExpandableDocumentCommand \IfMarksEqualT {0{page}mmm} {
412     \mark_if_eq:nnnnT {#1}{#2}{#3}{#4}
413 }
414 \NewExpandableDocumentCommand \IfMarksEqualF {0{page}mmm} {
415     \mark_if_eq:nnnnF {#1}{#2}{#3}{#4}
416 }

```

(End of definition for `\IfMarksEqualTF`, `\IfMarksEqualT`, and `\IfMarksEqualF`. These functions are documented on page 1060.)

9 L^AT_EX 2_ε integration

9.1 Core L^AT_EX 2_ε integration

`\_mark\_update\_singlecol\_structures:` This command updates the mark structures if we are producing a single column document.

```

417 \cs_new_protected:Npn \_mark\_update\_singlecol\_structures: {

```

First we update the page region (which also updates the previous-page).

The `\@outputbox` is normally in `\vbox` in L^AT_EX but we can't take that for granted (an `amsmath` test document changed it to an `\hbox` just to trip me up) so we are a little careful with unpack now.

```

418     \box_if_vertical:NTF \@outputbox
419     {
420         \mark_update_structure_from_material:nn {page}
421         { \vbox_unpack:N \@outputbox }
422     }
423     {
424         \mark_update_structure_from_material:nn {page}
425         { \hbox_unpack:N \@outputbox }
426     }

```

Then we provide the necessary updates for the aliases.

```

427 \mark_copy_structure:nn {previous-column}{previous-page}
428 \mark_copy_structure:nn {column}{page}
429 \mark_copy_structure:nn {first-column}{page}
430 \mark_copy_structure:nn {last-column}{page}
431 <{*trace}
432 % move this into status itself?
433 \__mark_debug:n
434 {
435     \__mark_status:nn
436     { in~ OR~ (
437         \legacy_if:nTF {@twoside}
438         { twoside-
439             \int_if_odd:nTF \c@page
440             { odd }{ even }
441         }
442         { oneside }
443     )
444 }
445 {0}
446 }
447 <{/trace}
448 }

```

(End of definition for __mark_update_singlecol_structures:.)

__mark_update_dbcol_structures: This commands handles the updates if we are doing two-column pages.

```

449 \cs_new_protected:Npn \__mark_update_dbcol_structures: {

```

First we update the column and previous-column regions using the material assembled in \@outputbox.

```

450 \box_if_vertical:NTF \@outputbox
451 {
452     \mark_update_structure_from_material:nn {column}
453     { \vbox_unpack:N \@outputbox }
454 }
455 {
456     \mark_update_structure_from_material:nn {column}
457     { \hbox_unpack:N \@outputbox }
458 }

```

How we have to update the alias regions depends on whether or not \@opcol was called to process the first column or to produce the completed page

```

459 \legacy_if:nTF {@firstcolumn}
460 {

```

If we are processing the first column then column is our first-column and there is no last-column yet, so we make those an error.

```

461     \mark_copy_structure:nn {first-column}{column}
462     \mark_set_structure_to_err:n {last-column}
463 }
464 {

```


If we produce the completed page then the `first-column` is the same as the new `previous-column`. However, the structure should already be correct if you think about it (because it was set to `column` last time which is now the `previous-column`), thus there is no need to make an update.

```
465 % \mark_copy_structure:nn {first-column}{previous-column}
```

However, we now have a proper `last-column` so we assign that.

```
466 \mark_copy_structure:nn {last-column}{column}
```

What now remains doing is to update the `page` and `previous-page` regions. For this we have to copy the settings in `page` into `previous-page` and then update `page` such that the top and first marks are taken from the `first-column` region and the last marks are taken from the `last-column` region. All this has to be done for all mark classes so we loop over our sequence.

Note that one loop is needed if we arrange the copy statements in a suitable way.

```
467 \seq_map_inline:Nn \g__mark_classes_seq
468 {
```

The `previous-page` updates need to come before the updates for `page` region because otherwise the values to copy are already overwritten. necessary values.

```
469 \tl_gset_eq:cc { g__mark_previous-page_top_ ##1 _tl }
470 { g__mark_page_top_ ##1 _tl }
471 \tl_gset_eq:cc { g__mark_previous-page_first_ ##1 _tl }
472 { g__mark_page_first_ ##1 _tl }
473 \tl_gset_eq:cc { g__mark_previous-page_last_ ##1 _tl }
474 { g__mark_page_last_ ##1 _tl }
```

To update the top we only have to copy what is in `first-column`:

```
475 \tl_gset_eq:cc { g__mark_page_top_ ##1 _tl }
476 { g__mark_first-column_top_ ##1 _tl }
477
```

Updating the `first` mark for the `page` region is more complicated. We first have to find out if there is any mark in the first column (this can be done by comparing the top and the `first` mark of that region).

```
478 \tl_if_eq:ccTF { g__mark_first-column_top_ ##1 _tl }
479 { g__mark_first-column_first_ ##1 _tl }
480 {
```

If there is no mark in the first column we copy the first mark of the last column. If that doesn't contain a mark we still get the right result because the first mark is then equal to the top mark.

```
481 \tl_gset_eq:cc { g__mark_page_first_ ##1 _tl }
482 { g__mark_last-column_first_ ##1 _tl }
483 }
484 {
```

On the other hand, if there is a mark in the first column we copy over the `first` mark from that column.

```
485 \tl_gset_eq:cc { g__mark_page_first_ ##1 _tl }
486 { g__mark_first-column_first_ ##1 _tl }
487 }
```

The logic for the `last` page mark is again simple, we can just copy the value in the `last` mark of the last column. If that column doesn't contain any marks, then the value in `last` will be automatically the same as the `last` from the first column.

```

488         \tl_gset_eq:cc { g__mark_page_last_      ##1 _tl }
489                     { g__mark_last-column_last_ ##1 _tl }
490     }
491 }
492 < *trace >
493     \__mark_debug:n
494     {
495         \__mark_status:nn
496         { in~ OR~ (
497             \legacy_if:nTF {@twoside}
498             { twoside-
499                 \int_if_odd:nTF \c@page
500                 { odd }{ even }
501             }
502             { oneside }
503             \space
504             \legacy_if:nTF {@firstcolumn}
505             { first~ }{ second~ }
506             column )
507         }
508         {0}
509     }
510 < /trace >
511 }

```

(End of definition for `\__mark_update_dbcol_structures:.`)

9.2 Other L^AT_EX 2_ε output routines

This section will cover support for packages that alter the L^AT_EX output routine (as necessary). The support for multicols (for now) is handled directly in that package.

```

512 < @@= >

```

`\@expl@@mark@update@singlecol@structures@@`

```

513 \cs_new_eq:NN \@expl@@mark@update@singlecol@structures@@
514 \__mark_update_singlecol_structures:

```

(End of definition for `\@expl@@mark@update@singlecol@structures@@.`)

`\@expl@@mark@update@dblcol@structures@@`

```

515 \cs_new_eq:NN \@expl@@mark@update@dblcol@structures@@
516 \__mark_update_dbcol_structures:

```

(End of definition for `\@expl@@mark@update@dblcol@structures@@.`)

9.3 Rollback information

```
517 <latexrelease>\IncludeInRelease{0000/00/00}{ltmarks}%  
518 <latexrelease>                                {Undo~Marks~handling}  
519 <latexrelease>
```

We keep the interface commands around even if we roll back in case they are used in packages that don't roll back. Not likely to do a lot of good, but then there is not much we can do, but this at least they won't give unknown csname errors.

```
520 <latexrelease>\DeclareRobustCommand \NewMarkClass[1]{}  
521 <latexrelease>\DeclareRobustCommand \InsertMark[2]{}  
522 <latexrelease>\RenewExpandableDocumentCommand \FirstMark { 0{ } m } { }  
523 <latexrelease>\RenewExpandableDocumentCommand \LastMark { 0{ } m } { }  
524 <latexrelease>\RenewExpandableDocumentCommand \TopMark { 0{ } m } { }  
525 <latexrelease>\RenewExpandableDocumentCommand \IfMarksEqualTF { 0{ } mmm }{ }  
526 <latexrelease>
```

Same here, this avoided extra roll back code in the OR.

```
527 <latexrelease>\let \@expl@@mark@update@singlecol@structures@@ \relax  
528 <latexrelease>\let \@expl@@mark@update@dblcol@structures@@ \relax  
529 <latexrelease>  
530 <latexrelease>  
531 <latexrelease>\EndModuleRelease  
532 \ExplSyntaxOff  
533 </2ekernel | latexrelease>  
  
Reset module prefix:  
534 <@@=>
```

File 49

ltpage.dtx

1 Page styles and related commands

1.1 Page Style Commands

`\pagestyle{<style>}` : sets the page style of the current and succeeding pages to *style*

`\thispagestyle{<style>}` : sets the page style of the current page only to *style*.

To define a page style *style*, you must define `\ps@style` to set the page style parameters.

1.2 How a page style makes running heads and feet

The `\ps@...` command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, and `\@evenfoot` to define the running heads and feet. (See output routine.) To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, etc., where `\chaptermark{<text>}` is called by `\chapter` to set a mark. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros.

(All the `\...mark` commands should be initialized to no-ops.)

1.3 marking conventions

L^AT_EX extends T_EX's `\mark` facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:

`\markboth{<left>}{<right>}` : Adds both marks.

`\markright{<right>}` : Adds a 'right' mark.

`\leftmark` : Used in the output routine, gets the current 'left' mark. Works like T_EX's `\botmark`.

`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like T_EX's `\firstmark`. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if 2 `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the `\pagestyle` command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

1 <*2ekernel>

`\pagestyle` User command to set the page style for this and following pages.

```
2 \def\pagestyle#1{%
3   \@ifundefined{ps@#1}%
4     \undefinedpagestyle
5     {\@nameuse{ps@#1}}}
```

(End of definition for `\pagestyle`.)

`\thispagestyle` User command to set the page style for this page only.

```

6 \def\thispagestyle#1{%
7   \ifundefined{ps@#1}%
8     \undefinedpagestyle
9     {\global\@specialpagetrue\gdef\@specialstyle{#1}}}
```

(End of definition for \thispagestyle.)

`\ps@empty` The empty page style: No head or foot line.

```

10 \def\ps@empty{%
11   \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
12   \let\@evenhead\@empty\let\@evenfoot\@empty}
```

(End of definition for \ps@empty.)

`\ps@plain` The plain page style: No head, centered page number in foot.

```

13 \def\ps@plain{\let\@mkboth\@gobbletwo
14   \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
15   \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}
```

(End of definition for \ps@plain.)

`\@leftmark` We implement `\@leftmark` and `\@rightmark` in terms of already defined commands to
`\@rightmark` save token space. We can't get rid of them since they are sometimes used in applications.

```

16 \let\@leftmark\@firstoftwo
17 \let\@rightmark\@secondoftwo
```

(End of definition for \@leftmark and \@rightmark.)

```

18 </2ekernel>
19 <*2ekernel | latexrelease>
20 <latexrelease>\IncludeInRelease{2025/06/01}%
21 <latexrelease>          {\markboth}{Drop legacy mark support}%
```

`\markboth` User commands for setting L^AT_EX marks.

`\markright` Test for `@nobreak` added 15 Apr 86 in `\markboth` and `\markright` letting `\label` and `\index` to `\relax` added 22 Feb 86 so these commands can appear in sectioning command arguments RmS 91/06/21 Same for `\glossary`

```

22 \ExplSyntaxOn
23 \DeclareRobustCommand*\markboth[2]{%
24   \mark_insert:nn{2e-left}{#1}
25   \mark_insert:nn{2e-right}{#2}
26   \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
27 }
28 \DeclareRobustCommand*\markright[1]{%
29   \mark_insert:nn{2e-right}{#1}
30   \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }
31 }
32 \ExplSyntaxOff
```

(End of definition for \markboth and \markright. These functions are documented on page 1064.)

```

33 </2ekernel | latexrelease>
34 <latexrelease>\EndIncludeInRelease
35 <latexrelease>\IncludeInRelease{2022/06/01}%
36 <latexrelease>          {\markboth}{New mark support}%
37 <latexrelease>\ExplSyntaxOn
38 <latexrelease>\DeclareRobustCommand*\markboth[2]{%
39 <latexrelease>  \begingroup
40 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
41 <latexrelease>    \unrestored@protected@xdef\@themark {{#1}{#2}}%
42 <latexrelease>    \@temptokena \expandafter{\@themark}%
43 <latexrelease>    \mark_insert:nn{2e-left}{#1}
44 <latexrelease>    \mark_insert:nn{2e-right}{#2}
45 <latexrelease>    \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
46 <latexrelease>    \mark{\the\@temptokena}%
47 <latexrelease>  \endgroup
48 <latexrelease>  \if@nobreak\ifvmode\nobreak\fi\fi}
49 <latexrelease>\DeclareRobustCommand*\markright[1]{%
50 <latexrelease>  \begingroup
51 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
52 <latexrelease>    \expandafter\@markright\@themark {#1}%
53 <latexrelease>    \@temptokena \expandafter{\@themark}%
54 <latexrelease>    \mark_insert:nn{2e-right}{#1}
55 <latexrelease>    \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }
56 <latexrelease>    \mark{\the\@temptokena}%
57 <latexrelease>  \endgroup
58 <latexrelease>  \if@nobreak\ifvmode\nobreak\fi\fi}
59 <latexrelease>\ExplSyntaxOff
60 <latexrelease>\EndIncludeInRelease
61 <latexrelease>\IncludeInRelease{2019/10/01}%
62 <latexrelease>          {\markboth}{Make commands robust}%
63 <latexrelease>
64 <latexrelease>\DeclareRobustCommand*\markboth[2]{%
65 <latexrelease>  \begingroup
66 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
67 <latexrelease>    \unrestored@protected@xdef\@themark {{#1}{#2}}%
68 <latexrelease>    \@temptokena \expandafter{\@themark}%
69 <latexrelease>    \mark{\the\@temptokena}%
70 <latexrelease>  \endgroup
71 <latexrelease>  \if@nobreak\ifvmode\nobreak\fi\fi}
72 <latexrelease>\DeclareRobustCommand*\markright[1]{%
73 <latexrelease>  \begingroup
74 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
75 <latexrelease>    \expandafter\@markright\@themark {#1}%
76 <latexrelease>    \@temptokena \expandafter{\@themark}%
77 <latexrelease>    \mark{\the\@temptokena}%
78 <latexrelease>  \endgroup
79 <latexrelease>  \if@nobreak\ifvmode\nobreak\fi\fi}
80 <latexrelease>
81 <latexrelease>\EndIncludeInRelease
82 <latexrelease>\IncludeInRelease{0000/00/00}%
83 <latexrelease>          {\markboth}{Make commands robust}%
84 <latexrelease>

```

Using `\kernel@make@fragile` doesn't really work if there are more redefinitions later on, so we have to repeat the above definition first and then undo it (or use `\def` directly).

```

85 <latexrelease>%\kernel@make@fragile\markboth
86 <latexrelease>%\kernel@make@fragile\markright
87 <latexrelease>\def\markboth#1#2{%
88 <latexrelease> \begingroup
89 <latexrelease> \let\label\relax \let\index\relax \let\glossary\relax
90 <latexrelease> \unrestored@protected@xdef\@themark {{#1}{#2}}%
91 <latexrelease> \@temptokena \expandafter{\@themark}%
92 <latexrelease> \mark{\the\@temptokena}%
93 <latexrelease> \endgroup
94 <latexrelease> \if@nbreak\ifvmode\nobreak\fi\fi}
95 <latexrelease>\def\markright#1{%
96 <latexrelease> \begingroup
97 <latexrelease> \let\label\relax \let\index\relax \let\glossary\relax
98 <latexrelease> \expandafter\@markright\@themark {#1}%
99 <latexrelease> \@temptokena \expandafter{\@themark}%
100 <latexrelease> \mark{\the\@temptokena}%
101 <latexrelease> \endgroup
102 <latexrelease> \if@nbreak\ifvmode\nobreak\fi\fi}
103 <latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <*2ekernel>

\leftmark
\rightmark
106 </2ekernel>
107 <*2ekernel | latexrelease>
108 <latexrelease>\IncludeInRelease{2025/06/01}%
109 <latexrelease> \{\leftmark\}{Use new mark mechanism}%
110 \ExplSyntaxOn
111 \cs_new:Npn \leftmark {\mark_use_last:nn{page}{2e-left}}
112 \cs_new:Npn \rightmark {\mark_use_first:nn{page}{2e-right}}
113 \ExplSyntaxOff
114 </2ekernel | latexrelease>
115 <latexrelease>\EndIncludeInRelease
116 <latexrelease>\IncludeInRelease{0000/00/00}%
117 <latexrelease> \{\leftmark\}{Use new mark mechanism}%
118 <latexrelease>
119 <latexrelease>\def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
120 <latexrelease>\def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

121 <latexrelease>\def\@themark{{}{}}
\markright 122 <latexrelease>\def\@markright#1#2#3{\@temptokena {#1}%
\mark 123 <latexrelease>\unrestored@protected@xdef\@themark{{\the\@temptokena}{#3}}}
124 <latexrelease>\EndIncludeInRelease
125 <*2ekernel>

```

(End of definition for `\leftmark` and others. These functions are documented on page 1064.)

`\raggedbottom` `\raggedbottom` typesets pages with no vertical stretch, so they have their natural height instead of all being exactly the same height. (Uses a space of .0001fil to avoid interfering with the 1fil space of `\newpage`.)

```

126 \DeclareRobustCommand\raggedbottom{%
127   \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}

(End of definition for \raggedbottom.)

\flushbottom \flushbottom: Inverse of \raggedbottom — makes all pages the same height.
128 \DeclareRobustCommand\flushbottom{%
129   \let\@textbottom\relax \let\@texttop\relax}

(End of definition for \flushbottom.)

\sloppy \sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull
ones. (14 June 85)
130 \DeclareRobustCommand\sloppy{%
131   \tolerance 9999%
132   \emergencystretch 3em%
133   \hfuzz .5\p@
134   \vfuzz\hfuzz}

(End of definition for \sloppy.)

sloppypar (env.) A sloppypar environment is equivalent to {\par \sloppy ... \par}.
135 \def\sloppypar{\par\sloppy}
136 \def\endsloppypar{\par}

\fussy Resets TeX's parameters to their normal finicky values.
137 \DeclareRobustCommand\fussy{%
138   \emergencystretch\z@
139   \tolerance 200%
140   \hfuzz .1\p@
141   \vfuzz\hfuzz}

(End of definition for \fussy.)

\overfullrule LATEX default is no overfull box rule. Changed by document class option.
142 \overfullrule 0pt

(End of definition for \overfullrule.)
143 </2ekernel>

```


File 50

ltclass.dtx

1 Introduction

This file implements the following declarations, which replace `\documentstyle` in $\text{\LaTeX} 2_{\epsilon}$ documents.

Note that old documents containing `\documentstyle` will be run using a compatibility option—thus keeping everyone happy, we hope!

The overall idea is that there are two types of ‘style files’: ‘class files’ which define elements and provide a default formatting for them; and ‘packages’ which provide extra functionality. One difference between $\text{\LaTeX} 2_{\epsilon}$ and $\text{\LaTeX} 2.09$ is that $\text{\LaTeX} 2_{\epsilon}$ packages may have options. Note that options to classes/packages may be implemented such that they input files, but these file names are not necessarily directly related to the option name.

2 User interface

```
\documentclass[<main-option-list>]{<class>}[<version>]
```

There must be exactly one such declaration, and it must come first. The *<main-option-list>* is a list of options which can modify the formatting of elements which are defined in the *<class>* file as well as in all following `\usepackage` declarations (see below). The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the class is found, a warning is issued.

```
\documentstyle[<main-option-list>]{<class>}[<version>]
```

The `\documentstyle` declaration is kept in order to maintain upward compatibility with $\text{\LaTeX} 2.09$ documents. It is similar to `\documentclass`, but it causes all options in *<main-option-list>* that the *<class>* does not use to be passed to `\RequirePackage` after the options have been processed. This maintains compatibility with the 2.09 behaviour. Also a flag is set to indicate that the document is to be processed in $\text{\LaTeX} 2.09$ compatibility mode. As far as most packages are concerned, this only affects the warnings and errors $\text{\LaTeX}$ generates. This flag does affect the definition of font commands, and `\sloppy`.

```
\usepackage[<package-option-list>]{<package-list>}[<version>]
```

There can be any number of these declarations. All packages in *<package-list>* are called with the same options.

Each *<package>* file defines new elements (or modifies those defined in the *<class>*), and thus extends the range of documents which can be processed. The *<package-option-list>* is a list of options which can modify the formatting of elements defined in the *<package>* file. The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the $\langle\text{package-option-list}\rangle$, each package processes the $\langle\text{main-option-list}\rangle$. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

Note that class files have the extension `.cls`, packages have the extension `.sty`.

`filecontents` (*env.*)

The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a ‘current directory’ or ‘default directory’) then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of $\text{\TeX}$. Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

As the environment is intended to be used on document level to bundle different files together, by default it refuses to overwrite files. If the option `force` is used then it always writes the file, but it generates a warning if it overwrites (or might overwrite) an existing file.⁵¹ If necessary, this warning can be suppressed by also using the option `nowarn` but for most use cases we recommend informing the user when an overwrite happens and therefore to not add this option.

The environment is now allowed anywhere in the document, but to ensure that all the necessary packages and options are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

⁵¹ $\text{\TeX}$ might see a file in the `texmf/` tree that is not in current directory, in which case no overwrite happens; but it can’t distinguish that case from the one in which an overwrite takes place.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

3 Class and Package interface

3.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.2 Package name and version

`\ProvidesPackage` A package can identify itself with the `\ProvidesPackage{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.3 Requiring other packages

`\RequirePackage` Packages or classes can load other packages using `\RequirePackage[<options>]{<name>}[<version>]`. If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

`\LoadClass` Similar to `\RequirePackage`, but for classes, may not be used in package files.

`\PassOptionsToPackage` Packages can pass options to other packages using:

`\PassOptionsToPackage{<options>}{<package>}`.

`\PassOptionsToClass` This adds the `<options>` to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\LoadClassWithOptions` `\LoadClassWithOptions{<name>}[<version>]:`

This is similar to `\LoadClass`, but it always calls class `<name>` with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by `\PassOptionsToClass`. `\RequirePackageWithOptions` is the analogous command for packages.

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using `\LoadClassWithOptions` is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the `article` class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case `article` is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

```
\IfPackageLoadedTF      To find out if a package has already been loaded, use
  \IfClassLoadedTF
  \@ifpackageloaded      \IfPackageLoadedTF{<package>}{<true>}{<false>}
  \@ifclassloaded
```

or the old name `\@ifpackageloaded`.

```
\IfPackageAtLeastTF     To find out if a package has already been loaded with a version equal to or more
  \IfClassAtLeastTF      recent than <date>, use
  \IfFileAtLeastTF
  \ifpackagelater        \IfPackageAtLeastTF{<package>}{<date>}{<true>}{<false>}
  \@ifclasslater         or the old name \@ifpackagelater.
  \IfFormatAtLeastTF     To test the format date use
                        \IfFormatAtLeastTF{<date>}{<true>}{<false>}
```

```
\IfPackageLoadedWithOptionsTF  To find out if a package has already been loaded with at least the options <options>,
\IfClassLoadedWithOptionsTF use
  \@ifpackagewith        \IfPackageLoadedWithOptionsTF{<package>}{<options>}{<true>}{<false>}
  \@ifclasswith
```

or the old name `\@ifpackagewith`.

There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

```
\NeedsDocumentMetadata  To require that a document starts with \DocumentMetadata use
                        \NeedsDocumentMetadata
```

3.4 Declaring new options

Options for classes and packages are built using the same macros.

<code>\DeclareOption</code>	To define a builtin option, use <code>\DeclareOption{<name>}{<code>}</code> .
<code>\DeclareOption*</code>	To define the default action to perform for local options which have not been declared, use <code>\DeclareOption*{<code>}</code> . <i>Note:</i> there should be no use of <code>\RequirePackage</code> , <code>\DeclareOption</code> , <code>\DeclareOption*</code> or <code>\ProcessOptions</code> inside <code>\DeclareOption</code> or <code>\DeclareOption*</code> . Possible uses for <code>\DeclareOption*</code> include: <code>\DeclareOption*{}</code> Do nothing. Silently accept unknown options. (This suppresses the usual warnings.) <code>\DeclareOption*{\@unknownoptionerror}</code> Complain about unknown local options. (The initial setting for package files.) <code>\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}}</code> Handle the current option by passing it on to the package <code><pkg-name></code> , which will presumably be loaded via <code>\RequirePackage</code> later in the file. This is useful for building ‘extension’ packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package. <code>\DeclareOption*{\InputIfFileExists{xx-CurrentOption.yyy}% {}% {\OptionNotUsed}}</code> Handle the option foo by loading the file <code>xx-foo.yyy</code> if it exists, otherwise do nothing, but declare that the option was not used. Actually the <code>\OptionNotUsed</code> declaration is only needed if this is being used in class files, but does no harm in package files.

3.5 Safe Input Macros

<code>\InputIfFileExists</code>	<code>\InputIfFileExists{<file>}{<then>}{<else>}</code> Inputs <code><file></code> if it exists. Immediately before the input, <code><then></code> is executed. Otherwise <code><else></code> is executed.
<code>\IfFileExists</code>	As above, but does not input the file. One thing you might like to put in the <code><else></code> clause is
<code>\@missingfileerror</code>	This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped and entering <code>x</code> quits the current run,
<code>\input</code>	This has been redefined from the L ^A T _E X2.09 definition, in terms of the new commands <code>\InputIfFileExists</code> and <code>\@missingfileerror</code> .
<code>\listfiles</code>	Giving this declaration in the preamble causes a list of all files input via the ‘safe input’ commands to be listed at the end. Any strings specified in the optional argument to <code>\ProvidesPackage</code> are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument.

4 Implementation

	<code>1 <*2kernel></code>
<code>\if@compatibility</code>	The flag for compatibility mode. <code>2 \newif\if@compatibility</code> <i>(End of definition for \if@compatibility.)</i>

`\@documentclasshook` This legacy hook is called after the first `\documentclass` command. It is *not* integrated with the new 2020 hook management system! By default this checks to see if `\@normalsize` is undefined, and if so, sets it to `\normalsize`.

```

3 \def\@documentclasshook{%
4   \ifx\@normalsize\undefined
5     \let\@normalsize\normalsize
6   \fi
7 }

```

(End of definition for `\@documentclasshook`.)

`\@declaredoptions` This list is automatically built by `\DeclareOption`. It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the corresponding `\ds<option>` commands are executed. All local `<option>`s which are not declared will be processed in the order defined by the optional argument of `\documentclass` or `\usepackage`.

```

8 \let\@declaredoptions\@empty

```

(End of definition for `\@declaredoptions`.)

`\@classoptionslist` List of options of the main class.

```

9 \let\@classoptionslist\relax
10 %\@onlypreamble\@classoptionslist

```

(End of definition for `\@classoptionslist`.)

`\@raw@classoptionslist` List of options of the main class (unprocessed).

```

11 \let\@raw@classoptionslist\relax

```

(End of definition for `\@raw@classoptionslist`.)

`\@unusedoptionlist` List of options of the main class that haven't been declared or loaded as class option files.

```

12 \let\@unusedoptionlist\@empty
13 %\@onlypreamble\@unusedoptionlist

```

(End of definition for `\@unusedoptionlist`.)

`\CurrentOption` Name of current package or option.

```

14 \let\CurrentOption\@empty

```

(End of definition for `\CurrentOption`.)

`\@currpath` Path to the current file if explicitly given.

```

15 </2kernel>
16 <*2kernel | latexrelease>
17 <latexrelease>
18 <latexrelease>\IncludeInRelease{2020/10/01}{\@currpath}%
19 <latexrelease> {Add \@currpath}%
20 \let\@currpath\@empty
21 <latexrelease>\EndIncludeInRelease
22 %
23 <latexrelease>\IncludeInRelease{0000/00/00}{\@currpath}%
24 <latexrelease> {Add \@currpath}%
25 <latexrelease>\let\@currpath\@undefined

```

```

26 <latexrelease>\EndIncludeInRelease
27 </2ekernel | latexrelease>
28 <*2ekernel>

(End of definition for \@currpath.)

\@currname Name of current package or option.
29 \let\@currname\@empty

(End of definition for \@currname.)

\@currentt The current file extension.
30 \global\let\@currentt=\@empty

(End of definition for \@currentt.)

\@clsextension The two possible values of \@currentt.
\@pkgextension 31 \def\@clsextension{cls}
32 \def\@pkgextension{sty}

(End of definition for \@clsextension and \@pkgextension.)

\@pushfilename Commands to push and pop the file name and extension.
\@popfilename #1 current name.
\@currnamestack #2 current extension.
#3 current catcode of @.
#4 Rest of the stack.
33 </2ekernel>
34 <*2ekernel | latexrelease>
35 <latexrelease>
36 <latexrelease>\IncludeInRelease{2026/11/01}%
37 <latexrelease> { \@pushfilename}{Manage target date requests}%
38 \def\@pushfilename{%

The push and pop macros are injected in \@pushfilename and \@popfilename so that
they correctly keep track of the hook labels.

This needs cleanup with the expl3 interfaces also playing here, e.g., \@expl@push@filename@@
needs cleanup and (and should probably not have this name either).

39 \@expl@push@filename@@
40 \xdef\@currnamestack{%
41 {\@currname}%
42 {\@currentt}%
43 {\the\catcode'\@}%

44 {\pkgcls@outerdate}%
45 \@currnamestack}%

Temporarily add a stack for \@currpath here. This should be integrated in the main file
stack eventually, but other packages rely on \@currnamestack having three elements per
file, so that isn't a trivial change. The prefix \@kernel@... hopefully discourages people
from using it.

46 \xdef\@kernel@currpathstack{%
47 {\@currpath}%
48 \@kernel@currpathstack}%
49 \@expl@push@filename@aux@@}

```

```

50 <latexrelease>\EndIncludeInRelease
51 <latexrelease>\IncludeInRelease{2020/10/01}{\@pushfilename}%
52 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
53 <latexrelease>
54 <latexrelease>\def\@pushfilename{%
55 <latexrelease> \expl@push@filename@@
56 <latexrelease> \xdef\@currnamestack{%
57 <latexrelease> {\@currname}%
58 <latexrelease> {\@currentext}%
59 <latexrelease> {\the\catcode'\@}%
60 <latexrelease> \@currnamestack}%
61 <latexrelease> \xdef\@kernel@currpathstack{%
62 <latexrelease> {\@currpath}%
63 <latexrelease> \@kernel@currpathstack}%
64 <latexrelease> \@expl@push@filename@aux@@}
65 <latexrelease>
66 <latexrelease>\EndIncludeInRelease

```

The following version of \@pushfilename didn't formally exist in this file, but in the 2020/02/02 release, expl3 was preloaded and it patched \@pushfilename (and \@popfilename) by adding some hooks in there. But rolling back to 2020/02/02, expl3 doesn't patch these macros again, so rolling back has to take those hooks into account. Same goes for \@popfilename.

```

67 <latexrelease>
68 <latexrelease>\IncludeInRelease{2020/02/02}{\@pushfilename}%
69 <latexrelease> {Add \@expl@push@filename@@}%
70 <latexrelease>\def\@pushfilename{%
71 <latexrelease> \expl@push@filename@@
72 <latexrelease> \xdef\@currnamestack{%
73 <latexrelease> {\@currname}%
74 <latexrelease> {\@currentext}%
75 <latexrelease> {\the\catcode'\@}%
76 <latexrelease> \@currnamestack}%
77 <latexrelease> \@expl@push@filename@aux@@}
78 <latexrelease>\EndIncludeInRelease
79 <latexrelease>

```

When we roll back from a release that has expl3 preloaded, the definitions of \@pushfilename and \@popfilename can't be completely rolled back otherwise expl3-based packages won't have the automatic \ExplSyntaxOff at the end. Here and below for \@popfilename, we don't roll back all the way through if coming from L^AT_EX > 2020-02-02.

```

80 <latexrelease>\IncludeInRelease{0000/00/00}{\@pushfilename}%
81 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
82 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
83 <latexrelease> \GenericInfo{}\{Defining 00-00-00\string\@pushfilename.\}
84 <latexrelease>\def\@pushfilename{%
85 <latexrelease> \xdef\@currnamestack{%
86 <latexrelease> {\@currname}%
87 <latexrelease> {\@currentext}%
88 <latexrelease> {\the\catcode'\@}%
89 <latexrelease> \@currnamestack}}
90 <latexrelease>\else
91 <latexrelease> \GenericInfo{}\{Defining 2020-02-02\string\@pushfilename.\}

```



```

92 <latexrelease>\def\@pushfilename{%
93 <latexrelease> \expl@push@filename@@
94 <latexrelease> \xdef\@currnamestack{%
95 <latexrelease> {\@currname}%
96 <latexrelease> {\@current}%
97 <latexrelease> {\the\catcode'\@}%
98 <latexrelease> \@currnamestack}%
99 <latexrelease> \expl@push@filename@aux@@}
100 <latexrelease>\fi
101 <latexrelease>\EndIncludeInRelease
102 \onlypreamble\@pushfilename

103 <latexrelease>
104 <latexrelease>\IncludeInRelease{2020/10/01}{\@popfilename}%
105 <latexrelease> {Add \expl@pop@filename@@}%
106 \def\@popfilename{\expl@@@hook@curr@name@pop@@
107 \expandafter\@p@pfilename\@currnamestack\@nil

```

Same for popping:

```

108 \expandafter\@p@pfilepath\@kernel@currpathstack\@nil
109 \expl@pop@filename@@}
110 <latexrelease>\EndIncludeInRelease
111 <latexrelease>
112 <latexrelease>\IncludeInRelease{2020/02/02}{\@popfilename}%
113 <latexrelease> {Add \@expl@push@filename@@}%
114 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
115 <latexrelease> \expl@pop@filename@@}
116 <latexrelease>\EndIncludeInRelease
117 <latexrelease>

118 <latexrelease>\IncludeInRelease{0000/00/00}{\@popfilename}%
119 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
120 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
121 <latexrelease> \GenericInfo{}\{Defining 00-00-00\string\@popfilename.\}
122 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
123 <latexrelease>\else
124 <latexrelease> \GenericInfo{}\{Defining 2020-02-02\string\@popfilename.\}
125 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
126 <latexrelease> \expl@pop@filename@@}
127 <latexrelease>\fi
128 <latexrelease>\EndIncludeInRelease
129 \onlypreamble\@popfilename

130 </2ekernel | latexrelease>
131 <*2ekernel>

132 </2ekernel>
133 <*2ekernel | latexrelease>
134 <latexrelease>\IncludeInRelease{2026/11/01}%
135 <latexrelease> {\@p@pfilename}{Manage target date requests}%
136 \def\@p@pfilename#1#2#3#4#5\@nil{%
137 \gdef\@currname{#1}%
138 \gdef\@current{#2}%
139 \catcode'\@#3\relax

```

The rollback code uses local variables so we do this here too.

```

140 \def\pkgcls@innerdate{#4}%
141 \gdef\@currnamestack{#5}}
142 \</2ekernel|latexrelease>
143 \<latexrelease>\EndIncludeInRelease
144 \<latexrelease>\IncludeInRelease{0000/00/00}%
145 \<latexrelease> \<@p@pfilename>{Manage target date requests}%
146 \<latexrelease>
147 \<latexrelease>\def\@p@pfilename#1#2#3#4\@nil{%
148 \<latexrelease> \gdef\@currname{#1}%
149 \<latexrelease> \gdef\@currentt{#2}%
150 \<latexrelease> \catcode'\@#3\relax
151 \<latexrelease> \gdef\@currnamestack{#4}}
152 \<latexrelease>
153 \<latexrelease>\EndIncludeInRelease
154 \<*2ekernel>
155 \<@onlypreamble>\@p@pfilename
156 \gdef\@currnamestack{}
157 \<@onlypreamble>\@currnamestack

```

(End of definition for \@pushfilename, \@popfilename, and \@currnamestack.)

\@kernel@currpathstack Path to the current file if explicitly given. The auxiliary is needed here to insert a \@empty to prevent the loss of braces.

```

158 \</2ekernel>
159 \<*2ekernel|latexrelease>
160 \<latexrelease>
161 \<latexrelease>\IncludeInRelease{2020/10/01}{\@kernel@currpathstack}%
162 \<latexrelease> {Add \@kernel@currpathstack}%

```

If rolling backwards to this release, \@kernel@currpathstack will be defined, so the \gdef line should not be executed, thus the \@gobblethree will take it out, so the satch isn't touched.

```

163 \<latexrelease>\@ifundefined{\@kernel@currpathstack}{\@gobblethree}
164 \gdef\@kernel@currpathstack{}%

```

If rolling forward to this release, then the \gdef line above will define the path stack to be empty (which it can't be, inside a file), so the code below will traverse the \@currnamestack, and add as many empty items to \@kernel@currpathstack as there are items in \@currnamestack, so both are back in sync. Most of the time latexrelease is loaded on top-level, so only one item is needed, but platexrelease loads it internally, so the more complicated loop is needed.

```

165 \<latexrelease>\ifx\@kernel@currpathstack\@empty
166 \<latexrelease> \def\reserved@a#1#2#3{%
167 \<latexrelease> \ifx\relax#3\else
168 \<latexrelease> \g@addto@macro\@kernel@currpathstack{}}%
169 \<latexrelease> \expandafter\reserved@a
170 \<latexrelease> \fi}%
171 \<latexrelease> \expandafter\reserved@a\@currnamestack{}\@relax}%
172 \<latexrelease>\fi
173 \def\@p@pfilepath#1{%
174 \gdef\@currpath{#1}\@p@pfilepath@aux\@empty}
175 \def\@p@pfilepath@aux#1\@nil{%
176 \xdef\@kernel@currpathstack{#1}}

```

```

177 <latexrelease>\EndIncludeInRelease
178 %
179 <latexrelease>\IncludeInRelease{0000/00/00}{\@kernel@currpathstack}%
180 <latexrelease> {Add \@kernel@currpathstack}%
181 <latexrelease>\let\@kernel@currpathstack\undefined
182 <latexrelease>\let\@p@pfilepath\undefined
183 <latexrelease>\let\@p@pfilepath@aux\undefined
184 <latexrelease>\EndIncludeInRelease
185 </2ekernel | latexrelease>
186 <*2ekernel>

```

(End of definition for \@kernel@currpathstack.)

\@optionlist Returns the option list of the file.

```

187 \def\@optionlist#1{%
188   \ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
189 %\@onlypreamble\@optionlist

```

(End of definition for \@optionlist.)

\@ifpackageloaded **\@ifclassloaded** **\@ifl@aded** Checks to see whether a file has been loaded.

```

190 \def\@ifpackageloaded{\@ifl@aded\@pkgextension}
191 \def\@ifclassloaded{\@ifl@aded\@clsextension}
192 \def\@ifl@aded#1#2{%
193   \expandafter\ifx\csname ver@#2.#1\endcsname\relax
194   \expandafter\@secondoftwo
195   \else
196   \expandafter\@firstoftwo
197   \fi}

```

(End of definition for \@ifpackageloaded and \@ifclassloaded.)

\@ifpackagelater **\@ifclasslater** **\@ifl@ter** Checks that the package loaded is more recent or equal to the given date. A better name for it would therefore been \@ifpackagelaterorequal but it is in use for more than 30 years, so ...

```

198 \def\@ifpackagelater{\@ifl@ter\@pkgextension}
199 \def\@ifclasslater{\@ifl@ter\@clsextension}

```

(End of definition for \@ifpackagelater and \@ifclasslater.)

\IfPackageAtLeastTF **\IfClassAtLeastTF** **\IfFileAtLeastTF** **\IfFormatAtLeastTF** Test if the format is later or equal to the given date.

```

200 </2ekernel>
201 <*2ekernel | latexrelease>
202 <latexrelease>\IncludeInRelease{2020/10/01}%
203 <latexrelease> {\IfFormatAtLeastTF}{Test format date}%
204 \def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
205 \let\IfPackageAtLeastTF\@ifpackagelater
206 \let\IfClassAtLeastTF\@ifclasslater
207 \def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}

```

For rollback pretend it was available since the beginning of dawn.

```

208 </2ekernel | latexrelease>
209 <latexrelease>\EndIncludeInRelease
210 <latexrelease>\IncludeInRelease{0000/00/00}%
211 <latexrelease>          {\IfFormatAtLeastTF}{Test format date}%
212 <latexrelease>\def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
213 <latexrelease>\let\IfPackageAtLeastTF\@ifpackagelater
214 <latexrelease>\let\IfClassAtLeastTF\@ifclasslater
215 <latexrelease>\def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}
216 <latexrelease>\EndIncludeInRelease
217 <*2ekernel>

```

(End of definition for \IfPackageAtLeastTF and others.)

\@ifl@ter

```

218 \def\@ifl@ter#1#2{%
219   \expandafter\@ifl@t@r
220   \csname ver@#2.#1\endcsname}
221 </2ekernel>
222
223   This internal macro is also used in \NeedsTeXFormat.
224
225 <latexrelease>\IncludeInRelease{2018/04/01}%
226 <latexrelease>          {\@ifl@t@r}{Guard against bad input}%
227 <*2ekernel | latexrelease>
228 \def\@ifl@t@r#1#2{%
229   \ifnum\expandafter\@parse@version@#1//00\@nil<%
230     \expandafter\@parse@version@#2//00\@nil
231     \expandafter\@secondoftwo
232   \else
233     \expandafter\@firstoftwo
234   \fi}
235 \def\@parse@version@#1{\@parse@version0#1}
236 </2ekernel | latexrelease>
237 <latexrelease>\EndIncludeInRelease
238 <latexrelease>\IncludeInRelease{0000/00/00}%
239 <latexrelease>          {\@ifl@t@r}{Guard against bad input}%
240 <latexrelease>\def\@ifl@t@r#1#2{%
241 <latexrelease>  \ifnum\expandafter\@parse@version#1//00\@nil<%
242 <latexrelease>    \expandafter\@parse@version#2//00\@nil
243 <latexrelease>    \expandafter\@secondoftwo
244 <latexrelease>  \else
245 <latexrelease>    \expandafter\@firstoftwo
246 <latexrelease>  \fi}
247 <latexrelease>\let\@parse@version@\@undefined
248 <latexrelease>\EndIncludeInRelease
249 <*2ekernel>

```

(End of definition for \@ifl@ter.)

```

247 </2ekernel>
248 <*2ekernel | latexreleasefirst>
249 \def\@parse@version#1/#2/#3#4#5\@nil{%
250 \@parse@version@dash#1-#2-#3#4\@nil
251 }

```

The `\if` test here ensures that an argument with no / or - produces 0 (actually 00).

```

252 \def\@parse@version@dash#1-#2-#3#4#5\@nil{%
253   \if\relax#2\relax\else#1\fi#2#3#4 }
254 </2ekernel | latexreleasefirst>
255 <*2ekernel>

\@ifpackagewith \@ifpackagewith{<name>}{<option-list>} Checks that <option-list> is a subset of
\@ifclasswith the options with which <name> was loaded.

256 \def\@ifpackagewith{\@ifoptions\@pkgextension}
257 \def\@ifclasswith{\@ifoptions\@clsextension}

258 \def\@ifoptions#1#2{%
259   \@expandtwoargs\@ifpti@ns{\@optionlist{#2.#1}}

    Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)

260 </2ekernel>
261 <latexrelease>\IncludeInRelease{2017/01/01}%
262 <latexrelease>           {\@ifpti@ns}{Spaces in option clash check}%
263 <*2ekernel | latexrelease>
264 \def\@ifpti@ns#1#2{%
265   \let\reserved@a\@firstoftwo

266   \edef\reserved@b{\zap@space#2 \@empty}%
267   \@for\reserved@b:=\reserved@b\do{%
268     \ifx\reserved@b\@empty
269     \else
270       \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
271       \ifin@
272       \else
273         \let\reserved@a\@secondoftwo
274       \fi
275     \fi
276   }%
277   \reserved@a}
278 </2ekernel | latexrelease>
279 <latexrelease>\EndIncludeInRelease
280 <latexrelease>\IncludeInRelease{0000/00/00}%
281 <latexrelease>           {\@ifpti@ns}{Spaces in option clash check}%
282 <latexrelease>\def\@ifpti@ns#1#2{%
283 <latexrelease> \let\reserved@a\@firstoftwo
284 <latexrelease> \@for\reserved@b:=#2\do{%
285 <latexrelease>   \ifx\reserved@b\@empty
286 <latexrelease>   \else
287 <latexrelease>     \expandafter\in@\expandafter
288 <latexrelease>           {\expandafter,\reserved@b,}{, #1,}%
289 <latexrelease>   \ifin@
290 <latexrelease>   \else
291 <latexrelease>     \let\reserved@a\@secondoftwo
292 <latexrelease>   \fi
293 <latexrelease> \fi
294 <latexrelease> }%
295 <latexrelease> \reserved@a}
296 <latexrelease>\EndIncludeInRelease
297 <*2ekernel>

```

(End of definition for \@ifpackagewith and \@ifclasswith.)

```
\IfPackageLoadedTF More public names for the commands already available for a long time.
\IfPackageLoadedWithOptionsTF 298 \</2kernel>
\IfClassLoadedTF 299 \<2kernel | latexrelease>
\IfClassLoadedWithOptionsTF 300 \<latexrelease>\IncludeInRelease{2024/06/01}%
301 \<latexrelease> {\IfPackageLoadedTF}{Test package loading}%
302 \let \IfPackageLoadedTF \<@ifpackageloaded
303 \let \IfClassLoadedTF \<@ifclassloaded
304 \let \IfPackageLoadedWithOptionsTF \<@ifpackagewith
305 \let \IfClassLoadedWithOptionsTF \<@ifclasswith
```

For rollback/rollforward pretend everything was available since the beginning of dawn.

```
306 \</2kernel | latexrelease>
307 \<latexrelease>\EndIncludeInRelease
308 \<latexrelease>\IncludeInRelease{0000/00/00}%
309 \<latexrelease> {\IfPackageLoadedTF}{Test package loading}%
310 \<latexrelease>
311 \<latexrelease>\let \IfPackageLoadedTF \<@ifpackageloaded
312 \<latexrelease>\let \IfClassLoadedTF \<@ifclassloaded
313 \<latexrelease>\let \IfPackageLoadedWithOptionsTF \<@ifpackagewith
314 \<latexrelease>\let \IfClassLoadedWithOptionsTF \<@ifclasswith
315 \<latexrelease>
316 \<latexrelease>\EndIncludeInRelease
317 \<2kernel>
```

(End of definition for \IfPackageLoadedTF and others.)

```
\IfPackageLoadedT A few more conditionals for convenience
\IfPackageLoadedF 318 \</2kernel>
\IfPackageAtLeastT 319 \<2kernel | latexrelease>
\IfPackageAtLeastF 320 \<latexrelease>\IncludeInRelease{2024/06/01}%
\IfClassAtLeastT 321 \<latexrelease> {\IfPackageLoadedT}{More conditionals}%
\IfClassAtLeastF 322 \def\IfPackageLoadedT #1{\IfPackageLoadedTF{#1}\@firstofone@gobble}
\IfFileAtLeastT 323 \def\IfPackageLoadedF #1{\IfPackageLoadedTF{#1}{}}
\IfFileAtLeastF 324 \def\IfClassLoadedT #1{\IfClassLoadedTF{#1}\@firstofone@gobble}
\IfFormatAtLeastT 325 \def\IfClassLoadedF #1{\IfClassLoadedTF{#1}{}}
\IfFormatAtLeastF 326 \def\IfPackageAtLeastT#1#2{\IfPackageAtLeastTF{#1}{#2}\@firstofone@gobble}
\IfFormatAtLeastF 327 \def\IfPackageAtLeastF#1#2{\IfPackageAtLeastTF{#1}{#2}{}}
\IfPackageLoadedWithOptionsT 328 \def\IfClassAtLeastT #1#2{\IfClassAtLeastTF{#1}{#2}\@firstofone@gobble}
\IfPackageLoadedWithOptionsF 329 \def\IfClassAtLeastF #1#2{\IfClassAtLeastTF{#1}{#2}{}}
\IfClassLoadedT 330 \def\IfFileAtLeastT #1#2{\IfFileAtLeastTF{#1}{#2}\@firstofone@gobble}
\IfClassLoadedF 331 \def\IfFileAtLeastF #1#2{\IfFileAtLeastTF{#1}{#2}{}}
\IfClassLoadedWithOptionsT 332 \def\IfFormatAtLeastT #1{\IfFormatAtLeastTF{#1}\@firstofone@gobble}
\IfClassLoadedWithOptionsF 333 \def\IfFormatAtLeastF #1{\IfFormatAtLeastTF{#1}{}}
\IfClassLoadedWithOptionsTF 334 \def\IfPackageLoadedWithOptionsT #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}\@firstofone@gobb
335 \def\IfPackageLoadedWithOptionsF #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
336 \def\IfClassLoadedWithOptionsT #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}\@firstofone@gobble}
337 \def\IfClassLoadedWithOptionsF #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}
```

These three commands haven't been there at all in the past.

```
338 \def\IfFileLoadedTF#1{%
339 \expandafter\ifx\csname ver@#1\endcsname\relax
340 \expandafter\@secondoftwo
```

```

341 \else
342 \expandafter\@firstoftwo
343 \fi}
344 \def\IfFileLoadedT #1{\IfFileLoadedTF{#1}\@firstofone\@gobble}
345 \def\IfFileLoadedF #1{\IfFileLoadedTF{#1}{}}

```

For rollback/rollforward pretend everything was available since the beginning of dawn.

```

346 </2ekernel | latexrelease>
347 <latexrelease>\EndIncludeInRelease
348 <latexrelease>\IncludeInRelease{0000/00/00}%
349 <latexrelease> \IfPackageLoadedT{More conditionals}%
350 <latexrelease>
351 <latexrelease>\def\IfPackageLoadedT #1#2{\IfPackageLoadedTF{#1}{#2}{}}
352 <latexrelease>\def\IfPackageLoadedF #1{\IfPackageLoadedTF{#1}{}}
353 <latexrelease>\def\IfClassLoadedT #1#2{\IfClassLoadedTF{#1}{#2}{}}
354 <latexrelease>\def\IfClassLoadedF #1{\IfClassLoadedTF{#1}{}}
355 <latexrelease>\def\IfPackageAtLeastT#1#2#3{\IfPackageAtLeastTF{#1}{#2}{#3}{}}
356 <latexrelease>\def\IfPackageAtLeastF #1#2{\IfPackageAtLeastTF{#1}{#2}{}}
357 <latexrelease>\def\IfClassAtLeastT #1#2#3{\IfClassAtLeastTF{#1}{#2}{#3}{}}
358 <latexrelease>\def\IfClassAtLeastF #1#2{\IfClassAtLeastTF{#1}{#2}{}}
359 <latexrelease>\def\IfFileAtLeastT #1#2#3{\IfFileAtLeastTF{#1}{#2}{#3}{}}
360 <latexrelease>\def\IfFileAtLeastF #1#2{\IfFileAtLeastTF{#1}{#2}{}}
361 <latexrelease>\def\IfFormatAtLeastT #1#2{\IfFormatAtLeastTF{#1}{#2}{}}
362 <latexrelease>\def\IfFormatAtLeastF #1{\IfFormatAtLeastTF{#1}{}}
363 <latexrelease>\def\IfPackageLoadedWithOptionsT #1#2#3{\IfPackageLoadedWithOptionsTF{#1}{#2}{#3}{}}
364 <latexrelease>\def\IfPackageLoadedWithOptionsF #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
365 <latexrelease>\def\IfClassLoadedWithOptionsT #1#2#3{\IfClassLoadedWithOptionsTF{#1}{#2}{#3}{}}
366 <latexrelease>\def\IfClassLoadedWithOptionsF #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}
367 <latexrelease>
368 <latexrelease>\def\IfFileLoadedTF#1{%
369 <latexrelease> \expandafter\ifx\csname ver@#1\endcsname\relax
370 <latexrelease> \expandafter\@secondoftwo
371 <latexrelease> \else
372 <latexrelease> \expandafter\@firstoftwo
373 <latexrelease> \fi}
374 <latexrelease>\def\IfFileLoadedT#1#2{\IfFileLoadedTF{#1}{#2}{}}
375 <latexrelease>\def\IfFileLoadedF #1{\IfFileLoadedTF{#1}{}}
376 <latexrelease>
377 <latexrelease>\EndIncludeInRelease
378 <*2ekernel>

```

(End of definition for \IfPackageLoadedT and others.)

\ProvidesPackage Checks that the current filename is correct, and defines `\ver@filename`.

```

379 </2ekernel>
380 <latexrelease>\IncludeInRelease{2020/10/01}%
381 <latexrelease> \ProvidesPackage{Check name with \strcmp}%
382 <*2ekernel | latexrelease>
383 \def\ProvidesPackage#1{%
384 \xdef\@gtempa{#1}%

```

Here `\@currpath` is explicitly added to the file name to report when a package or class is loaded using an explicit path. Loading using a path in the argument is supported but not encouraged.

```

385 \expandtwoargs\expl@str@if@eq@nnTF

```



```

429 <latexrelease>                {\@pr@videpackage}{Protection for package info}%
430 <latexrelease>
431 <latexrelease>\def\@pr@videpackage[#1]{%
432 <latexrelease>  \expandafter\protected@xdef                %      <-- protected...
433 <latexrelease>    \csname ver@\@currname.\@current\endcsname{#1}%
434 <latexrelease>\ifx\@current\@clsextension
435 <latexrelease>  \typeout{Document Class: \@gtempa\space#1}%
436 <latexrelease>  \else
437 <latexrelease>    \protected@wlog{Package: \@gtempa\space#1}%  <--- protected
438 <latexrelease>  \fi}
439 <latexrelease>
440 <latexrelease>\EndIncludeInRelease
441 <latexrelease>\IncludeInRelease{0000/00/00}%
442 <latexrelease>                {\@pr@videpackage}{Protection for package info}%
443 <latexrelease>
444 <latexrelease>\def\@pr@videpackage[#1]{%
445 <latexrelease>  \expandafter\xdef\csname ver@\@currname.\@current\endcsname{#1}%
446 <latexrelease>  \ifx\@current\@clsextension
447 <latexrelease>    \typeout{Document Class: \@gtempa\space#1}%
448 <latexrelease>  \else
449 <latexrelease>    \wlog{Package: \@gtempa\space#1}%
450 <latexrelease>  \fi}
451 <latexrelease>\let\protected@wlog\@undefined
452 <latexrelease>
453 <latexrelease>\EndIncludeInRelease
454 <*2ekernel>
455 \@onlypreamble\@pr@videpackage

```

\ProvidesClass Like **\ProvidesPackage**, but for classes. This needs a dummy latexrelease block to copy the definition of **\ProvidesPackage** as it changes across releases.

```

456 </2ekernel>
457 <latexrelease>\IncludeInRelease{0000/00/00}%
458 <latexrelease>  {\ProvidesClass}{Track \ProvidesPackage}%
459 <*2ekernel | latexrelease>
460 \let\ProvidesClass\ProvidesPackage
461 \@onlypreamble\ProvidesClass
462 </2ekernel | latexrelease>
463 <latexrelease>\EndIncludeInRelease
464 <*2ekernel>

```

(End of definition for \ProvidesClass.)

\ProvidesFile Like **\ProvidesPackage**, but for arbitrary files. Do not apply **\@onlypreamble** to these, as we may want to label files input during the document.

```

\@providesfile 465 \def\ProvidesFile#1{%
466   \begingroup
467   \catcode'\ 10 %
468   \ifnum \endlinechar<256 %
469     \ifnum \endlinechar>\m@ne
470       \catcode\endlinechar 10 %
471   \fi
472   \fi

```

```

473 \makeother\/%
474 \makeother\&%
475 \kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]

```

During initex a special version of \@providesfile is used. The real definition is installed right at the end, in ltfinal.dtx.

```

def\@providesfile#1[#2]{%
  \wlog{File: #1 #2}%
  \expandafter\xdef\csname ver@#1\endcsname{#2}%
  \endgroup}

```

(End of definition for \ProvidesFile and \@providesfile.)

\PassOptionsToPackage If the package has been loaded, we check that it was first loaded with the options.
 \PassOptionsToClass Otherwise we add the option list to that of the package.

```

476 </2kernel>
477 <latexrelease>\IncludeInRelease{2021/06/01}%
478 <latexrelease>          {\@passOptions}{Raw option lists}%
479 <*2kernel|latexrelease>
480 \def\@passOptions#1#2#3{%
481   \@expl@@@filehook@set@curr@file@@nNN
482   {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
483   \reserved@a\reserved@b
484   \@expl@@@filehook@clear@replacement@flag@@
485   \expandafter\protected\xdef\csname opt@\reserved@a\endcsname{%
486     \ifundefined{opt@\reserved@a}\@empty
487     {\csname opt@\reserved@a\endcsname,%}
488     \zap@space#2 \@empty}%
489   \expandafter\let
490     \csname opt@#3.#1\expandafter\endcsname
491     \csname opt@\reserved@a\endcsname

```

Extend raw option list

```

492   \ifundefined{@raw@opt@#3.#1}%
493     {\expandafter\gdef\csname @raw@opt@#3.#1\expandafter\endcsname
494       \expandafter{#2}}%
495     {\expandafter\g@addto@macro\csname @raw@opt@#3.#1\expandafter\endcsname
496       \expandafter{\expandafter,#2}}%
497   }
498 </2kernel|latexrelease>
499 <latexrelease>\EndIncludeInRelease
500 <latexrelease>\IncludeInRelease{2020/10/01}{\@passOptions}
501 <latexrelease> {Add file replacement in \@passOptions}%
502 <latexrelease>
503 <latexrelease>\def\@passOptions#1#2#3{%
504 <latexrelease>   \@expl@@@filehook@set@curr@file@@nNN
505 <latexrelease>   {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
506 <latexrelease>   \reserved@a\reserved@b
507 <latexrelease>   \@expl@@@filehook@clear@replacement@flag@@
508 <latexrelease>   \expandafter\xdef\csname opt@\reserved@a\endcsname{%
509 <latexrelease>     \ifundefined{opt@\reserved@a}\@empty

```

```

510 <latexrelease>      {\csname opt@\reserved@a\endcsname,}%
511 <latexrelease>      \zap@space#2 \@empty}%
512 <latexrelease>      \expandafter\let
513 <latexrelease>      \csname opt@#3.#1\expandafter\endcsname
514 <latexrelease>      \csname opt@\reserved@a\endcsname}
515 <latexrelease>\EndIncludeInRelease

516 <latexrelease>\IncludeInRelease{0000/00/00}{\@pass@options}
517 <latexrelease>      {\@pass@options}%
518 <latexrelease>
519 <latexrelease>\def\@pass@options#1#2#3{%
520 <latexrelease>      \expandafter\xdef\csname opt@#3.#1\endcsname{%
521 <latexrelease>      \ifundefined{opt@#3.#1}\@empty
522 <latexrelease>      {\csname opt@#3.#1\endcsname,}%
523 <latexrelease>      \zap@space#2 \@empty}}
524 <latexrelease>\EndIncludeInRelease
525 <*2ekernel>

526 \onlypreamble\@pass@options

527 \def\PassOptionsToPackage{\@pass@options\@pkgextension}
528 \def\PassOptionsToClass{\@pass@options\@clsextension}
529 \onlypreamble\PassOptionsToPackage
530 \onlypreamble\PassOptionsToClass

```

(End of definition for \PassOptionsToPackage and \PassOptionsToClass.)

\DeclareOption Adds an option as a \ds@ command, or the default \default@ds command.
\DeclareOption*

```

531 \def\DeclareOption{%
532   \let\@fileswith@pti@ns\@badrequireerror
533   \@ifstar\@defdefault@ds\@declareoption}
534 \long\def\@declareoption#1#2{%
535   \xdef\@declaredoptions{\@declaredoptions,#1}%
536   \toks@{#2}%
537   \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
538 \long\def\@defdefault@ds#1{%
539   \toks@{#1}%
540   \edef\default@ds{\the\toks@}}
541 \onlypreamble\DeclareOption
542 \onlypreamble\@declareoption
543 \onlypreamble\@defdefault@ds

```

(End of definition for \DeclareOption and \DeclareOption*.)

\OptionNotUsed If we are in a class file, add \CurrentOption to the list of unused options. Otherwise, in
\@remove@eq@value a package file do nothing.

```

544 </2ekernel>
545 <latexrelease>\IncludeInRelease{2021/06/01}%
546 <latexrelease>      {\OptionNotUsed}{filter unused option list}%
547 <*2ekernel | latexrelease>
548 \ExplSyntaxOn
549 \def\@remove@eq@value#1=#2\@nil{\tl_trim_spaces:n{#1}}
550 \ExplSyntaxOff

551 \def\OptionNotUsed{%
552   \ifx\@current\@clsextension
553     \xdef\@unusedoptionlist{%

```

```

554 \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
555 \expandafter\@remove@eq@value\CurrentOption=\@nil}%
556 \fi}
557 </2ekernel | latexrelease>
558 <latexrelease>\EndIncludeInRelease
559 <latexrelease>\IncludeInRelease{0000/00/00}%
560 <latexrelease> \OptionNotUsed}{filter unused option list}%
561 <latexrelease>\let\@remove@eq@value\@undefined

562 <latexrelease>\def\OptionNotUsed{%
563 <latexrelease> \ifx\@current\@clsextension
564 <latexrelease> \xdef\@unusedoptionlist{%
565 <latexrelease> \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
566 <latexrelease> \CurrentOption}%
567 <latexrelease> \fi}
568 <latexrelease>\EndIncludeInRelease
569 <*2ekernel>

```

```
570 \@onlypreamble\OptionNotUsed
```

(End of definition for \OptionNotUsed and \@remove@eq@value.)

\default@ds The default option code. Set by \@onefilewithoptions to either \OptionNotUsed for classes, or \@unknownoptionerror for packages. This may be reset in either case with \DeclareOption*.

```
571 % \let\default@ds\OptionNotUsed
```

(End of definition for \default@ds.)

\ProcessOptions \ProcessOptions calls \ds@option for each known package option, then calls \default@ds for each option on the local options list. Finally resets all the declared options to \relax. The empty option does nothing, this has to be reset on the off chance it's set to \relax if an empty element gets into the \@declaredoptions list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```

572 \def\ProcessOptions{%
573 \let\ds@\@empty
574 \protected@edef\@curroptions{\@optionlist{\@currname.\@current}}}%
575 \@ifstar\@xprocessoptions\@processoptions}
576 \@onlypreamble\ProcessOptions

577 \def\@processoptions{%
578 \@for\CurrentOption:=\@declaredoptions\do{%
579 \ifx\CurrentOption\@empty\else
580 \@expandtwoargs\in@{,\CurrentOption,}%
581 ,\ifx\@current\@clsextension\else\@classoptionslist,\fi
582 \@curroptions,}%
583 \ifin@
584 \@use@option
585 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
586 \fi
587 \fi}%
588 \@processoptions}
589 \@onlypreamble\@processoptions

```

```

590 </2ekernel>
591 <latexrelease>\IncludeInRelease{2021/06/01}%
592 <latexrelease>          {\xprocessoptions}{safer @xprocessoptions}%
593 <*2ekernel | latexrelease>
594 \def\xprocessoptions{%
595   \ifx\@currentx\@clsextension\else
596   \ifx\@classoptionslist\relax\else
597   \@for\CurrentOption:=\@classoptionslist\do{%
598     \ifx\CurrentOption\@empty\else
599     \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{}{%
600       \useoption
601       \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
602     }%
603   }%
604   \fi
605 }%
606 \processoptions}
607 </2ekernel | latexrelease>
608 <latexrelease>\EndIncludeInRelease
609 <latexrelease>\IncludeInRelease{0000/00/00}%
610 <latexrelease>          {\xprocessoptions}{safer @xprocessoptions}%
611 <latexrelease>\let\@remove@eq@value\@undefined
612 <latexrelease>\def\xprocessoptions{%
613 <latexrelease>   \ifx\@currentx\@clsextension\else
614 <latexrelease>     \@for\CurrentOption:=\@classoptionslist\do{%
615 <latexrelease>       \ifx\CurrentOption\@empty\else
616 <latexrelease>         \@expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
617 <latexrelease>         \ifin@
618 <latexrelease>           \useoption
619 <latexrelease>           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
620 <latexrelease>         \fi
621 <latexrelease>       \fi}%
622 <latexrelease>   \fi
623 <latexrelease>   \processoptions}
624 <latexrelease>\EndIncludeInRelease
625 <*2ekernel>
626 \onlypreamble\xprocessoptions

```

The common part of \ProcessOptions and \ProcessOptions*.

```

627 </2ekernel>
628 <*2ekernel | latexrelease>
629 <latexrelease>\IncludeInRelease{2020/10/01}%
630 <latexrelease>          {\@processoptions}{Unused options issue}%
631 \def\@processoptions{%
632   \@for\CurrentOption:=\@curroptions\do{%
633     \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}%
634     {\useoption
635      \default@ds}%

```

There should not be any non-empty definition of \CurrentOption at this point, as all the declared options were executed earlier. This is for compatibility with 2.09 styles which use \def\ds@... directly, and so have options which do not appear in \@declaredoptions.

```

636     \useoption}%

```

Clear all the definitions for option code. First set all the declared options to `\relax`, then reset the ‘default’ and ‘empty’ options. and the list of declared options.

```

637 \for\CurrentOption:=\@declaredoptions\do{%
638   \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%

639 \let\CurrentOption\@empty
640 \let\@fileswith@pti@ns\@fileswith@pti@ns
641 \AtEndOfPackage{\expandafter\let
642   \csname unprocessedoptions-\@currname.\@currentx\endcsname
643   \relax}}
644 \@onlypreamble\@process@pti@ns
645 </2ekernel | latexrelease>
646 <latexrelease>\EndIncludeInRelease
647 <latexrelease>\IncludeInRelease{0000/00/00}%
648 <latexrelease>          {\@process@pti@ns}{Unused options issue}%
649 <latexrelease>
650 <latexrelease>\def\@process@pti@ns{%
651 <latexrelease>  \for\CurrentOption:=\@curroptions\do{%
652 <latexrelease>    \ifundefined{ds@\CurrentOption}%
653 <latexrelease>      {\@use@option
654 <latexrelease>        \default@ds}%
655 <latexrelease>      \@use@option}%
656 <latexrelease>  \for\CurrentOption:=\@declaredoptions\do{%
657 <latexrelease>    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
658 <latexrelease>  \let\CurrentOption\@empty
659 <latexrelease>  \let\@fileswith@pti@ns\@fileswith@pti@ns
660 <latexrelease>  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
661 <latexrelease>\EndIncludeInRelease
662 <*2ekernel>

```

(End of definition for \ProcessOptions and \ProcessOptions.)*

\@options `\@options` is a synonym for `\ProcessOptions*` for upward compatibility with L^AT_EX 2.09 style files.

```

663 \def\@options{\ProcessOptions*}
664 \@onlypreamble\@options

```

(End of definition for \@options.)

\@use@option Execute the code for the current option.

```

665 </2ekernel>
666 <latexrelease>\IncludeInRelease{2021/06/01}%
667 <latexrelease>          {\@use@option}{filter unused option list}%
668 <*2ekernel | latexrelease>
669 \def\@use@option{%
670   \@expandtwoargs\@removeelement
671     {\expandafter\@remove@eq@value\CurrentOption=\@nil}%
672   \@unusedoptionlist\@unusedoptionlist
673   \csname ds@\detokenize\expandafter{\CurrentOption}\endcsname}
674 </2ekernel | latexrelease>
675 <latexrelease>\EndIncludeInRelease
676 <latexrelease>\IncludeInRelease{0000/00/00}%
677 <latexrelease>          {\@use@option}{filter unused option list}%
678 <latexrelease>\def\@use@option{%

```

```

679 <latexrelease> \expandtwoargs\@removeelement\CurrentOption
680 <latexrelease> \unusedoptionlist\unusedoptionlist
681 <latexrelease> \csname ds@\CurrentOption\endcsname}
682 <latexrelease>\EndIncludeInRelease
683 <*2ekernel>

684 \@onlypreamble\@use@option
(End of definition for \@use@option.)

```

\ExecuteOptions \ExecuteOptions{<option-list>} executes the code declared for each option.

```

685 </2ekernel>
686 <latexrelease>\IncludeInRelease{2017/01/01}%
687 <latexrelease> \ExecuteOptions}{Spaces in \ExecuteOptions}%
688 <*2ekernel | latexrelease>
689 \def\ExecuteOptions#1{%

```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```

690 \edef\@fortmp{\zap@space#1 \@empty}%
691 \def\reserved@a##1\@nil{%
692 \for\CurrentOption:=\@fortmp\do
693 {\csname ds@\CurrentOption\endcsname}%
694 \edef\CurrentOption{##1}}%
695 \expandafter\reserved@a\CurrentOption\@nil}
696 </2ekernel | latexrelease>
697 <latexrelease>\EndIncludeInRelease
698 <latexrelease>\IncludeInRelease{0000/00/00}%
699 <latexrelease> \ExecuteOptions}{Spaces in \ExecuteOptions}%
700 <latexrelease>\def\ExecuteOptions#1{%
701 <latexrelease> \def\reserved@a##1\@nil{%
702 <latexrelease> \for\CurrentOption:=#1\do
703 <latexrelease> {\csname ds@\CurrentOption\endcsname}%
704 <latexrelease> \edef\CurrentOption{##1}}%
705 <latexrelease> \expandafter\reserved@a\CurrentOption\@nil}
706 <latexrelease>\EndIncludeInRelease
707 <*2ekernel>

708 \@onlypreamble\ExecuteOptions
(End of definition for \ExecuteOptions.)

```

The top-level commands, which just set some parameters then call the internal command, \@fileswithoptions.

\documentclass The main new-style class declaration.

```

709 \def\documentclass{%
710 \let\documentclass\@twoclasseserror
711 \if@compatibility\else\let\usepackage\RequirePackage\fi
712 \@fileswithoptions\@clsextension}
713 \@onlypreamble\documentclass
(End of definition for \documentclass.)

```

\documentstyle 2.09 style class ‘style’ declaration.

```

714 \def\documentstyle{%
715 \makeatletter\input{latex209.def}\makeatother
716 \documentclass}
717 \@onlypreamble\documentstyle

```

(End of definition for \documentstyle.)

\RequirePackage Load package if not already loaded.

```
718 \def\RequirePackage{%  
719   \@fileswithoptions\@pkgextension}  
720 \@onlypreamble\RequirePackage
```

(End of definition for \RequirePackage.)

\LoadClass Load class.

```
721 \def\LoadClass{%  
722   \ifx\@current\@pkgextension  
723     \@latex@error  
724     {\noexpand\LoadClass in package file}%  
725     {You may only use \noexpand\LoadClass in a class file.}%  
726   \fi  
727   \@fileswithoptions\@clsextension}  
728 \@onlypreamble\LoadClass
```

(End of definition for \LoadClass.)

\@loadwithoptions Pass the current option list on to a class or package. #1 is \@cls-or-pkgextension, #2 is \RequirePackage or \LoadClass, #3 is the class or package to be loaded.

```
729 </2ekernel>  
730 <latexrelease>\IncludeInRelease{2021/06/01}%  
731 <latexrelease>          {\@loadwithoptions}{Raw option lists load with options}%  
732 <*2ekernel | latexrelease>  
733 \def\@loadwithoptions#1#2#3{%  
734   \expandafter\let\csname opt@#3.#1\expandafter\endcsname  
735     \csname opt@\@currname.\@current\endcsname  
736   \expandafter\let\csname @raw@opt@#3.#1\expandafter\endcsname  
737     \csname @raw@opt@\@currname.\@current\endcsname  
738   #2{#3}}  
739 </2ekernel | latexrelease>  
740 <latexrelease>\EndIncludeInRelease  
  
741 <latexrelease>\IncludeInRelease{0000/00/00}  
742 <latexrelease>          {\@loadwithoptions}{Raw option lists load with options}%  
743 <latexrelease>\def\@loadwithoptions#1#2#3{%  
744   <latexrelease> \expandafter\let\csname opt@#3.#1\expandafter\endcsname  
745   <latexrelease> \csname opt@\@currname.\@current\endcsname  
746   <latexrelease> #2{#3}}  
747 <latexrelease>\EndIncludeInRelease  
748 <*2ekernel>  
  
749 \@onlypreamble\@loadwithoptions
```

(End of definition for \@loadwithoptions.)

\LoadClassWithOptions Load class ‘#1’ with the current option list.

```
750 \def\LoadClassWithOptions{%  
751   \@loadwithoptions\@clsextension\LoadClass}  
752 \@onlypreamble\LoadClassWithOptions
```

(End of definition for \LoadClassWithOptions.)

`\RequirePackageWithOptions` Load package ‘#1’ with the current option list.

```

753 </2ekernel>
754 <*2ekernel | latexrelease>
755 <latexrelease>\IncludeInRelease{2020/10/01}%
756 <latexrelease>          {\RequirePackageWithOptions}{Unused options issue}%
757 \def\RequirePackageWithOptions{%
The resetting of the unprocessed options is now done on a par package basis.
758   \AtEndOfPackage{\expandafter\let
759                     \csname unprocessedoptions-\@currname.\@current\endcsname
760                     \relax}%
761   \@loadwithoptions\@pkgextension\RequirePackage}
762 \@onlypreamble\RequirePackageWithOptions
763 </2ekernel | latexrelease>
764 <latexrelease>\EndIncludeInRelease
765 <latexrelease>\IncludeInRelease{0000/00/00}%
766 <latexrelease>          {\RequirePackageWithOptions}{Unused options issue}%
767 <latexrelease>
768 <latexrelease>\def\RequirePackageWithOptions{%
769 <latexrelease>   \AtEndOfPackage{\let\unprocessedoptions\relax}%
770 <latexrelease>   \@loadwithoptions\@pkgextension\RequirePackage}
771 <latexrelease>\EndIncludeInRelease
772 <*2ekernel>

```

(End of definition for \RequirePackageWithOptions.)

`\usepackage` To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```

773 \def\usepackage#1{%
774   \@latex@error
775     {\noexpand \usepackage before \string\documentclass}%
776     {\noexpand \usepackage may only appear in the document
777      preamble, i.e.,\MessageBreak
778      between \noexpand\documentclass and
779      \string\begin{document}.}%
780   \@gobble}
781 \@onlypreamble\usepackage

```

(End of definition for \usepackage.)

`\NeedsTeXFormat` Check that the document is running on the correct system.

```

782 \def\NeedsTeXFormat#1{%
783   \def\reserved@a{#1}%
784   \ifx\reserved@a\fmtname
785     \expandafter\@needsformat
786   \else
787     \@latex@error{This file needs format ‘\reserved@a’%
788     \MessageBreak but this is ‘\fmtname’}{%
789     The current input file will not be processed
790     further,\MessageBreak
791     because it was written for some other flavor of
792     TeX.\MessageBreak\@ehd}%

```

If the file is not meant to be processed by L^AT_EX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```

793     \endinput \fi}
794 \@onlypreamble\NeedsTeXFormat

795 \def\@needsformat{%
796   \@ifnextchar[%]
797     \@needsformat
798   {}
799 \@onlypreamble\@needsformat

800 \def\@needsformat[#1]{%
801   \ifl@t@r\fmtversion{#1}{}%
802   {\@latex@warning@no@line
803     {You have requested release ‘#1’ of LaTeX,\MessageBreak
804       but only release ‘\fmtversion’ is available}}
805 \@onlypreamble\@needsformat

```

(End of definition for \NeedsTeXFormat.)

\NeedsDocumentMetadata Check that \DocumentMetadata has been used.

```

806 \protected\def\NeedsDocumentMetadata{%
807   \IfDocumentMetadataF{%
808     \@latex@error{This file needs \string\DocumentMetadata}%
809     {The current input file will not be processed
810       further,\MessageBreak
811       because the document didn't start with \string\DocumentMetadata.%
812       \MessageBreak\@ehd}}
813   \IfDocumentMetadataF{\endinput}}%

```

(End of definition for \NeedsDocumentMetadata.)

\zap@space \zap@space foo\space\@empty removes all spaces from foo that are not protected by { } groups.

```

814 \def\zap@space#1 #2{%
815   #1%
816   \ifx#2\@empty\else\expandafter\zap@space\fi
817   #2}

```

(End of definition for \zap@space.)

\@fileswithoptions The common part of \documentclass and \usepackage.

```

818 </2ekernel>
819 <latexrelease>\IncludeInRelease{2024/06/01}%
820 <latexrelease>          {\@fileswithoptions}{Check Group}%
821 <*2ekernel | latexrelease>
822 \def\@fileswithoptions#1{%
823   \ifnum\currentgrouplevel>\z@
824     \@latex@error
825       {Loading a class or package in a group}%
826       {Classes and packages should only be loaded at the top level}%
827   \fi
828   \@ifnextchar[%]
829     {\@fileswithoptions#1}%
830     {\@fileswithoptions#1[]}}

```

```

831 </2ekernel | latexrelease>
832 <latexrelease>\EndIncludeInRelease
833 <latexrelease>\IncludeInRelease{0000/00/00}%
834 <latexrelease>          {\@fileswithoptions}{Check Group}%
835 <latexrelease>\def\@fileswithoptions#1{%
836 <latexrelease>  \ifnextchar[%
837 <latexrelease>    {\@fileswithoptions#1}%
838 <latexrelease>    {\@fileswithoptions#1[]}}
839 <latexrelease>\EndIncludeInRelease
840 <*2ekernel>

841 \@onlypreamble\@fileswithoptions

842 \def\@fileswithoptions#1[#2]#3{%
843   \ifnextchar[%
844     {\@fileswithoptions#1[#{#2}]#3}%
845     {\@fileswithoptions#1[#{#2}]#3[]}}
846 \@onlypreamble\@fileswithoptions

```

Then we do some work.

First of all, we define the global variables. Then we look to see if the file has already been loaded. If it has, we check that it was first loaded with at least the current options. If it has not, we add the current options to the package options, set the default version to be 0000/00/00, and load the file if we can find it. Then we check the version number.

Finally, we restore the old file name, reset the default option, and we set the catcode of @.

For classes, we can immediately process the file. For other types, #2 could be a comma separated list, so loop through, processing each one separately.

```

847 </2ekernel>
848 <latexrelease>\IncludeInRelease{2020/10/01}%
849 <latexrelease>          {\@fileswithoptions}{ifx tests in \@fileswithoptions}%
850 <*2ekernel | latexrelease>
851 \def\@fileswithoptions#1[#2]#3[#4]{%
852   \ifx#1\@clsextension
853     \ifx\@classoptionslist\relax
854       \protected@xdef\@classoptionslist{\zap@space#2 \@empty}%

```

Save raw class list.

```

855   \gdef\@raw@classoptionslist{#2}%
856   \def\reserved@a{%
857     \@onefilewithoptions#3[#{#2}][#{#4}]#1%
858     \@documentclasshook}%
859   \else
860     \def\reserved@a{%
861       \@onefilewithoptions#3[#{#2}][#{#4}]#1%
862     \fi
863   \else

```

build up a list of calls to \@onefilewithoptions (one for each package) without thrashing the parameter stack.

```

864   \def\reserved@b##1,{%

```

If #1 is \@nnil we have reached the end of the list (older version used \@nil here but \@nil is undefined so \ifx equal to all undefined commands)

```

865   \ifx\@nnil##1\relax\else

```

If `\ifx\@nnil##1\@nnil` is true then #1 is (presumably) empty (Older code used `\relax` which is slightly easier to get into #1 by mistake, which would spoil this test.)

```

866     \ifx\@nnil##1\@nnil\else

867         \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][{#4}]%
868         \noexpand\@pkgextension
869     \fi
870     \expandafter\reserved@b
871 \fi}%
872 \edef\reserved@a{\zap@space#3 \@empty}%
873 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
874 \fi
875 \reserved@a}
876 </2ekernel | latexrelease>
877 <latexrelease>\EndIncludeInRelease
878 <latexrelease>\IncludeInRelease{2017/01/01}%
879 <latexrelease>    {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
880 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
881 <latexrelease>    \ifx#1\@clsextension
882 <latexrelease>        \ifx\@classoptionslist\relax
883 <latexrelease>            \xdef\@classoptionslist{\zap@space#2 \@empty}%
884 <latexrelease>            \def\reserved@a{%
885 <latexrelease>                \@onefilewithoptions#3[{#2}][{#4}]#1%
886 <latexrelease>                \@documentclasshook}%
887 <latexrelease>        \else
888 <latexrelease>            \def\reserved@a{%
889 <latexrelease>                \@onefilewithoptions#3[{#2}][{#4}]#1%
890 <latexrelease>        \fi
891 <latexrelease>    \else
892 <latexrelease>        \def\reserved@b##1,{%
893 <latexrelease>            \ifx\@nnil##1\relax\else
894 <latexrelease>                \ifx\@nnil##1\@nnil\else
895 <latexrelease>                    \noexpand\@onefilewithoptions##1[{#2}][{#4}]%
896 <latexrelease>                    \noexpand\@pkgextension
897 <latexrelease>                \fi
898 <latexrelease>                \expandafter\reserved@b
899 <latexrelease>            \fi}%
900 <latexrelease>            \edef\reserved@a{\zap@space#3 \@empty}%
901 <latexrelease>            \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
902 <latexrelease>        \fi
903 <latexrelease>    \reserved@a}
904 <latexrelease>\EndIncludeInRelease
905 <latexrelease>\IncludeInRelease{0000/00/00}%
906 <latexrelease>    {\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
907 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
908 <latexrelease>    \ifx#1\@clsextension
909 <latexrelease>        \ifx\@classoptionslist\relax
910 <latexrelease>            \xdef\@classoptionslist{\zap@space#2 \@empty}%
911 <latexrelease>            \def\reserved@a{%
912 <latexrelease>                \@onefilewithoptions#3[{#2}][{#4}]#1%
913 <latexrelease>                \@documentclasshook}%
914 <latexrelease>        \else
915 <latexrelease>            \def\reserved@a{%
916 <latexrelease>                \@onefilewithoptions#3[{#2}][{#4}]#1%

```

```

917 <latexrelease> \fi
918 <latexrelease> \else
919 <latexrelease> \def\reserved@b##1,{%
920 <latexrelease> \ifx\@nil##1\relax\else
921 <latexrelease> \ifx\relax##1\relax\else
922 <latexrelease> \noexpand\@onefilewithoptions##1[#{#2}][#{#4}]%
923 <latexrelease> \noexpand\@pkgextension
924 <latexrelease> \fi
925 <latexrelease> \expandafter\reserved@b
926 <latexrelease> \fi}%
927 <latexrelease> \edef\reserved@a{\zap@space#3 \@empty}%
928 <latexrelease> \edef\reserved@a{%
929 <latexrelease> \expandafter\reserved@b\reserved@a,\@nil,}%
930 <latexrelease> \fi
931 <latexrelease> \reserved@a}
932 <latexrelease>\EndIncludeInRelease
933 <*2ekernel>
934 \@onlypreamble\@fileswith@ptions

```

This macro is used when loading packages or classes.

`\load@onefilewithoptions` Have the main argument as #1, so we only need one `\expandafter` above.

```

935 </2ekernel>
936 <*2ekernel | latexrelease>
937 <latexrelease>\IncludeInRelease{2020/10/01}%
938 <latexrelease> {\@onefilewithoptions}{Hooks and unused options issue}%

```

Here this macro is called `\@onefilewithoptions`, but further ahead in this file it is renamed to `\load@onefilewithoptions`, and `\@onefilewithoptions` becomes a wrapper around this, used for bookkeeping when rolling back. Therefore, when in `latexrelease`, we need to define `\load@onefilewithoptions` instead, thus the extra guarded `\def` line below:

```

939 <*2ekernel>
940 \def\@onefilewithoptions#1[#2][#3]#4{%
941 </2ekernel>
942 <latexrelease>\def\load@onefilewithoptions#1[#2][#3]#4{%

```

We have to sanitize file names, so that something like

```

\usepackage{some/local/path/array}
\usepackage{array}

```

won't load `array.sty` twice. It is remotely possible that those are two different files, but as a matter of principles, we will consider that the base file name uniquely identifies a package, regardless of where it lives. This assumption already holds for file hooks, for example, which address the hook to a file by its base name only.

We'll use `\@expl@@@filehook@set@curr@file@@nNN` to parse the file name and return the `<path>` and `<base+ext>` in separate token lists. Further ahead, most operations use `\@currname` which doesn't have a path attached to it; only few actions prepend `\@currpath` to `\@currname` (namely loading, as we have to respect the given path).

A file substitution isn't followed just yet because at this point we are parsing user input, so the file is still what the user asked for, and not the file actually loaded.

```

943 \@expl@@@filehook@set@curr@file@@nNN{#1.#4}\reserved@a\reserved@b
944 \edef\reserved@c{\def\noexpand\reserved@c####1%

```

```

945 \detokenize\expandafter{\expanded{. #4}}%
946 \noexpand\@nil{\def\noexpand\reserved@a{####1}}\reserved@c
947 \expandafter\reserved@c\reserved@a\@nil
948 \@pushfilename
949 \xdef\@currname{\string\makeletter\reserved@a}%
950 \xdef\@currpath{\ifx\reserved@b\@empty\else\reserved@b\fi}%
951 \global\let\@currentx#4%

```

The command `\ver@<file>.<ext>` is used to signal that a package is already loaded, either because it is in fact loaded, or because its loading was suppressed. In minimal installations, said package may not exist but still have its loading suppressed with `\ver@<file>.<ext>`, so before checking if the file exists we have to check that we do need to load it with `\@ifl@aded`. If we don't, then there's no point in checking for a typo or load-disabling.

```

952 \@ifl@aded\@currentx\@currname

```

In the current preferred approach, a key family name will exist for processing using `lkeys`. In that case, we replace the previous package options with the new ones, then call the key handler. Otherwise, we use the more classical clash handler.

```

953 {%
954 \@ifundefined{opt@handler@\@currname.\@currentx}
955 {\@onefilewithoptions@clashchk{#2}}
956 {%
957 \expandafter\protected@edef
958 \csname opt@\@currname.\@currentx\endcsname
959 {\zap@space#2 \@empty}%
960 \expandafter\def
961 \csname @raw@opt@\@currname.\@currentx\expandafter\endcsname
962 \expandafter{#2}%
963 \@nameuse{opt@handler@\@currname.\@currentx}%
964 }%
965 }%
966 {\makeatletter

```

The next line seems to be necessary for 2.09 compatibility (the way the code is written there) This seems questionable and should be look at as in 2e it is definitely unnecessary at this point!

```

967 \@reset@ptions

```

First we take the `<name>` and `<ext>` given in the argument and check if the file exists, and issue an error otherwise asking for a correction with `\@missingfileerror`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```

968 \IfFileExists{\@currpath\@currname.\@currentx}{}%
969 {\@missing@onefilewithoptions{#2}}%

```

If `\@currname` is empty (the user replied to the “Enter file name” prompt with `<RETURN>`), so stop here (do `\@popfilename` to pop the item just added above).

This `\@gobble` omits the date check at the end.

```

970 \ifx\@currname\@empty
971 \expandafter\@gobble
972 \else

```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with `\@filehook@file@push` then call `\set@curr@file` to set `\@curr@file` (and do any required substitution), then actually load the class/package with `\load@onefile@withoptions`. `\set@curr@file` also needs the file path.

```

973      \@disable@packageload@do{\@currname.\@currentx}%
974      {\@expl@@@filehook@file@push@@
975      \set@curr@file{\@currpath\@currname.\@currentx}%
976      \@filehook@set@CurrentFile

```

The `\set@curr@file` line above might have replaced the file, so `\@currname` and `\@currentx` may no longer hold the actual package being loaded, so in that case we need to update these two token lists (`\@curr@file` holds the file name after replacement, so we parse that).

The requested file is saved in `\@currpkg@reqd` to be used in `\InputIfFileExists` later: if the updated `\@currname` and `\@currentx` are used we lose track of the substitution, so `\CurrentFile` and `\CurrentFileUsed` will be (incorrectly) the same.

```

977      \expandafter\@swaptwoargs\expandafter
978      {\expandafter{\@currpkg@reqd}}%
979      {% <

```

`\@currpkg@reqd` doesn't take a path because it is used later to assign `\opt@...` and `\ver@...`

```

980      \edef\@currpkg@reqd{\@currname.\@currentx}%
981      \ifx\CurrentFile\CurrentFileUsed
982      \else
983      \filename@parse\@curr@file
984      \edef\@currpath{\string@makeletter\filename@area}%
985      \edef\@currname{\string@makeletter\filename@base}%
986      \edef\@currentx{\string@makeletter\filename@ext}%
987      \fi
988      \load@onefile@withoptions{#2}%
989      \def\@currpkg@reqd%\@currpkg@reqd{
990      }% >

```

Now just clean up and exit.

```

991      \@expl@@@filehook@file@pop@@}%
992      \expandafter\@firstofone
993      \fi}%

```

Except in the case where `\@currname` is empty, the date is checked against the date marked in the package file:

```

994      {\@ifl@ter\@currentx{\@currname}{#3}{}}%
995      {\@latex@warning@no@line
996      {You have requested,\on@line,
997      version\MessageBreak
998      '#3' of \@cls@pkg\space \@currname,\MessageBreak
999      but only version\MessageBreak
1000      '\csname ver@\@currname.\@currentx\endcsname'\MessageBreak
1001      is available}}%

1002      \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi}%
1003      \@popfilename
1004      \@reset@options}

```

If the package is already loaded, check that there were no option clashes.

```

1005 \def\@onefilewithoptions@clashchk#1{%
\@onefilewithoptions@clashchk
1006   \if@options\@currentx{\@currname}{#1}{}%
1007   {\@latex@error
1008     {Option clash for \@cls@pkg\space \@currname}%
1009     {The package \@currname\space has already been loaded
1010      with options:\MessageBreak
1011      \space\space[\@optionlist{\@currname.\@currentx}]\MessageBreak
1012      There has now been an attempt to load it
1013      with options:\MessageBreak
1014      \space\space[#1]\MessageBreak
1015      Adding the global options:\MessageBreak
1016      \space\space
1017      \@optionlist{\@currname.\@currentx},#1\MessageBreak
1018      to your \noexpand\documentclass declaration may fix this.%
1019      \MessageBreak
1020      Try typing \space <return> \space to proceed.}}%
1021   \@firstofone}

1022 \let\@currpkg@reqd\@empty
1023 \@onlypreamble\@onefilewithoptions
1024 \let\@unprocessedoptions\@undefined

```

Now the action taken when a file is not found. Path must be included here as it eventually leads to a file lookup.

```

1025 \def\@missing@onefilewithoptions#1{%
1026   \@missingfileerror{\@currpath\@currname}\@currentx
1027   \global\let\@currpath\@missingfile@area
1028   \global\let\@currname\@missingfile@base
1029   \global\let\@currentx\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions
1030 \def\load@onefile@withoptions#1{%
1031   \let\CurrentOption\@empty
1032   \@reset@ptions

```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```

1033   \def\reserved@a{%
1034     \@pass@ptions\@currentx{#1}{\@currname}%

1035     \expandafter\let
1036     \csname opt@\@currpkg@reqd\expandafter\endcsname
1037     \csname opt@\@currname.\@currentx\endcsname
1038     \expandafter\let
1039     \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
1040     \csname @raw@opt@\@currname.\@currentx\endcsname
1041     \global\expandafter
1042     \let\csname ver@\@currname.\@currentx\endcsname\@empty

```

We initialize \dots-h@@k here and only if we load the file so that it remains undefined otherwise.

```

1043     \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname\@empty

```


When the current extension is `\@pkgextension` we are loading a package otherwise, if it is `\@clsextension`, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```

1044 %-----
1045 \ifx\@currentx\@pkgextension
1046 \UseHook{package/before}%
1047 \UseOneTimeHook{package/\@currname/before}%
1048 \else
1049 \ifx\@currentx\@clsextension
1050 \UseHook{class/before}%
1051 \UseOneTimeHook{class/\@currname/before}%
1052 \fi
1053 \fi

```

Now actually load the file (at this point we are certain it exists, but use `\InputIfFileExists` so that file hooks are executed). `\@currpath` is needed here too.

```

1054 \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
1055 {\@latex@error
1056 {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
1057 %-----

```

In older versions of the code `\@unprocessedoptions` would generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

This has changed in 2020. We now use a separate macro per package to avoid interference in case of nested packages. The whole code for handling this issue (GitHub 22) was provided by Hironobu Yamashita, thanks for that.

```

1058 \expandafter\let\csname unprocessedoptions-\@currname.\@currentx\endcsname
1059 \@@unprocessedoptions
1060 \csname\@currname.\@currentx-h@@k\endcsname
1061 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1062 \undefined

```

Catch the case where the packages has handled the options and redefined `\@unprocessedoptions` to `\relax` (old interface). In that case no error should be produced.

```

1063 \ifx\@unprocessedoptions\relax
1064 \let\@unprocessedoptions\undefined

```

Otherwise run the per package set of unused options.

```

1065 \else
1066 \csname unprocessedoptions-\@currname.\@currentx\endcsname
1067 \fi

```

In either case we drop the macro afterwards as it is no longer needed.

```

1068 \expandafter\let
1069 \csname unprocessedoptions-\@currname.\@currentx\endcsname
1070 \undefined

```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```

1071 %-----
1072 \ifx\@currentx\@pkgextension
1073 \UseOneTimeHook{package/\@currname/after}%
1074 \UseHook{package/after}%
1075 \else

```

```

1076 \ifx\@current\@clsextension
1077 \UseOneTimeHook{class/\@currname/after}%
1078 \UseHook{class/after}%
1079 \fi
1080 \fi}%
1081 %-----
1082 \@ifl@aded\@current\@currname{}\reserved@a}}

Now declare the non-generic package and class hooks used above:

1083 \NewHook{package/before}
1084 \NewHook{class/before}
1085 \NewReversedHook{package/after}
1086 \NewReversedHook{class/after}

1087 </2ekernel | latexrelease>
1088 <latexrelease>\EndIncludeInRelease
1089 <latexrelease>\IncludeInRelease{0000/00/00}%
1090 <latexrelease> \{\@onefilewithoptions\}{Hooks and unused options issue}%
1091 <latexrelease>

```

Because of the way \@onfilewithoptions is changed for rollback handling below we have to define \load@onefilewithoptions when rolling back!

```

1092 <latexrelease>\def\load@onefilewithoptions#1[#2][#3]#4{%
1093 <latexrelease> \@pushfilename
1094 <latexrelease> \xdef\@currname{#1}%
1095 <latexrelease> \global\let\@current#4%
1096 <latexrelease> \let\CurrentOption\@empty
1097 <latexrelease> \@resetoptions
1098 <latexrelease> \makeatletter
1099 <latexrelease> \def\reserved@a{%
1100 <latexrelease> \@ifl@aded\@current{#1}%
1101 <latexrelease> {\@ifoptions\@current{#1}{#2}{}}%
1102 <latexrelease> {\@latexerror
1103 <latexrelease> {Option clash for \@cls@pkg\space #1}%
1104 <latexrelease> {The package #1 has already been loaded
1105 <latexrelease> with options:\MessageBreak
1106 <latexrelease> \space\space[\@optionlist{#1.\@current}]\MessageBreak
1107 <latexrelease> There has now been an attempt to load it
1108 <latexrelease> with options:\MessageBreak
1109 <latexrelease> \space\space[#2]\MessageBreak
1110 <latexrelease> Adding the global options:\MessageBreak
1111 <latexrelease> \space\space
1112 <latexrelease> \@optionlist{#1.\@current},#2\MessageBreak
1113 <latexrelease> to your \noexpand\documentclass declaration may fix this.%
1114 <latexrelease> \MessageBreak
1115 <latexrelease> Try typing \space <return> \space to proceed.}}}%
1116 <latexrelease> {\@passoptions\@current{#2}{#1}%
1117 <latexrelease> \global\expandafter
1118 <latexrelease> \let\csname ver@\@currname.\@current\endcsname\@empty
1119 <latexrelease> \expandafter\let\csname\@currname.\@current-h@k\endcsname\@empty
1120 <latexrelease> \InputIfFileExists
1121 <latexrelease> {\@currname.\@current}%
1122 <latexrelease> {}%
1123 <latexrelease> {\@missingfileerror\@currname\@current}%
1124 <latexrelease> \let\@unprocessedoptions\@unprocessedoptions

```

```

1125 <latexrelease> \csname\@currname.\@current-h@@k\endcsname
1126 <latexrelease> \expandafter\let\csname\@currname.\@current-h@@k\endcsname
1127 <latexrelease> \undefined
1128 <latexrelease> \@unprocessedoptions}%
1129 <latexrelease> \@ifl@ter\@current{#1}{#3}{}%
1130 <latexrelease> {\@latex@warning@no@line
1131 <latexrelease> {You have requested,\on@line,
1132 <latexrelease> version\MessageBreak
1133 <latexrelease> ‘#3’ of \cls@pkg\space #1,\MessageBreak
1134 <latexrelease> but only version\MessageBreak
1135 <latexrelease> ‘\csname ver@#1.\@current\endcsname’\MessageBreak
1136 <latexrelease> is available}}%
1137 <latexrelease> \ifx\@current\@clsextension\let\LoadClass\@twoloadclasserror\fi
1138 <latexrelease> \popfilename
1139 <latexrelease> \@reset@options}%
1140 <latexrelease> \reserved@a}
1141 <latexrelease>
1142 <latexrelease>\let \load@onefile@withoptions \undefined
1143 <latexrelease>\let \@missing@onefilewithoptions \undefined
1144 <latexrelease>
1145 <latexrelease>\EndIncludeInRelease
1146 <*2ekernel>

```

(End of definition for \@fileswithoptions and others.)

\@@fileswith@pti@ns Save the definition (for error checking).

```

1147 \let\@@fileswith@pti@ns\fileswith@pti@ns
1148 \@onlypreamble\@@fileswith@pti@ns

```

(End of definition for \@@fileswith@pti@ns.)

\@reset@ptions Reset the default option, and clear lists of declared options.

```

1149 \def\@reset@ptions{%
1150 \global\ifx\@current\@clsextension
1151 \let\default@ds\OptionNotUsed
1152 \else
1153 \let\default@ds\@unknownoptionerror
1154 \fi
1155 \global\let\ds@\@empty
1156 \global\let\@declaredoptions\@empty}
1157 \@onlypreamble\@reset@ptions

```

(End of definition for \@reset@ptions.)

4.1 Hooks

Allow code to be saved to be executed at specific later times.

Here we save things in macros. I considered using toks registers (and `\addto@hook` from the NFSS code), but that would require stacking the contents in the case of required packages, so just generate a new macro for each package.

\@begindocumenthook Stuff to appear at the beginning or end of the document.

```

\@enddocumenthook 1158 \ifx\@begindocumenthook\undefined
1159 \let\@begindocumenthook\empty
1160 \fi
1161 \let\@enddocumenthook\empty

```

(End of definition for \@begindocumenthook and \@enddocumenthook.)

```

\AtEndOfPackage The access functions.
\AtEndOfClass
\AtBeginDocument 1162 \def\AtEndOfPackage{%
\AtEndDocument 1163 \expandafter\g@addto@macro\csname\@currname.\@current-h@@k\endcsname}
1164 \let\AtEndOfClass\AtEndOfPackage
1165 \@onlypreamble\AtEndOfPackage
1166 \@onlypreamble\AtEndOfClass

1167 \</2ekernel>
1168 \<*2ekernel | latexrelease>
1169 \<latexrelease>\IncludeInRelease{2020/10/01}%
1170 \<latexrelease> \<\AtBeginDocument>\{Use hook system}%
1171 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
1172 \DeclareRobustCommand\AtEndDocument {\AddToHook{enddocument}}
1173 %\DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl

1174 \</2ekernel | latexrelease>
1175 \<latexrelease>\EndIncludeInRelease
1176 \<latexrelease>\IncludeInRelease{0000/00/00}%
1177 \<latexrelease> \<\AtBeginDocument>\{Use hook system}%
1178 \<latexrelease>
1179 \<latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
1180 \<latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
1181 \<latexrelease>
1182 \<latexrelease>\EndIncludeInRelease
1183 \<*2ekernel>

```

In its initial implementation (not using the hook system) \AtBeginDocument was made \@onlypreamble because using it later had no effect whatsoever, thus was most certainly an unintended programming error. With the reimplementa- tion, using the begindocument hook internally, this has changed because adding to a onetime hook after it has already been used simply executes the additional code immediately. We therefore no longer generate an error if it is used inside the document so that \AddToHook{begindocument} and \AtBeginDocument are truly equivalent (as claimed in the hook documentation).

```
1184 %\@onlypreamble\AtBeginDocument
```

(End of definition for \AtEndOfPackage and others.)

\@cls@pkg The current file type.

```

1185 \</2ekernel>
1186 \<*2ekernel | latexrelease>
1187 \<latexrelease>\IncludeInRelease{2024/11/01}%
1188 \<latexrelease> \<\@cls@pkg>\{Allow for more extensions}%
1189 \def\@cls@pkg{%
1190 \ifx\@current\@clsextension
1191 document class%
1192 \else
1193 \ifx\@current\@pkgextension
1194 package%
1195 \else
1196 file%
1197 \fi
1198 \fi}

```

```

1199 </2ekernel | latexrelease>
1200 <latexrelease>\EndIncludeInRelease

1201 <latexrelease>\IncludeInRelease{0000/00/00}%
1202 <latexrelease>          {\@cls@pkg}{Allow for more extensions}%
1203 <latexrelease>
1204 <latexrelease>\def\@cls@pkg{%
1205 <latexrelease>  \ifx\@currentx\@clsextension
1206 <latexrelease>    document class%
1207 <latexrelease>  \else
1208 <latexrelease>    package%
1209 <latexrelease>  \fi}
1210 <latexrelease>\EndIncludeInRelease
1211 <*2ekernel>

1212 \@onlypreamble\@cls@pkg

(End of definition for \@cls@pkg.)

```

\@unknownoptionerror Bad option.

```

1213 \def\@unknownoptionerror{%
1214   \@latex@error
1215     {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\@currname’}%
1216     {The option ‘\CurrentOption’ was not declared in
1217       \@cls@pkg\space‘\@currname’, perhaps you\MessageBreak
1218         misspelled its name.
1219       Try typing \space <return>
1220       \space to proceed.}}
1221 \@onlypreamble\@unknownoptionerror

(End of definition for \@unknownoptionerror.)

```

\@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.

```

1222 \def\@unprocessedoptions{%
1223   \ifx\@currentx\@pkgextension
1224     \protected@edef\@curroptions{\@optionlist{\@currname.\@currentx}}%
1225     \@for\CurrentOption:=\@curroptions\do{%
1226       \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
1227   \fi}
1228 \@onlypreamble\@unprocessedoptions
1229 \@onlypreamble\@unprocessedoptions

(End of definition for \@unprocessedoptions.)

```

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.

```

1230 \def\@badrequireerror#1[#2]#3[#4]{%
1231   \@latex@error
1232     {\noexpand\RequirePackage or \noexpand\LoadClass
1233       in Options Section}%
1234     {The \@cls@pkg\space ‘\@currname’ is defective.\MessageBreak
1235       It attempts to load ‘#3’ in the options section, i.e.,\MessageBreak
1236       between \noexpand\DeclareOption and \string\ProcessOptions.}}
1237 \@onlypreamble\@badrequireerror

(End of definition for \@badrequireerror.)

```

```

\@twoloadclasserror Two \LoadClass in a class.
1238 \def\@twoloadclasserror{%
1239   \latex@error
1240     {Two \noexpand\LoadClass commands}%
1241     {You may only use one \noexpand\LoadClass in a class file}}
1242 \onlypreamble\@twoloadclasserror

(End of definition for \@twoloadclasserror.)

\@twoclasseserror Two \documentclass or \documentstyle.
1243 \def\@twoclasseserror#1#{%
1244   \latex@error
1245     {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
1246     {The document may only declare one class.}\@gobble}
1247 \onlypreamble\@twoclasseserror

(End of definition for \@twoclasseserror.)

```

4.2 Providing shipment

```

\two@digits Prefix a number less than 10 with ‘0’.
1248 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}

(End of definition for \two@digits.)

```

```

filecontents This environment implements inline files. The star-form does not write extra comments
\endfilecontents into the file.
1249 \</2kernel>
1250 \<*2kernel | latexrelease>
1251 \<latexrelease>\IncludeInRelease{2020/10/01}%
1252 \<latexrelease>          {\filecontents}{Define \q@curr@file directly (gh/220)}%
1253 %

```

We use @tempswa to mean no preamble writing and reuse @filesw to indicate no over-writing:

```

1254 \def\filecontents{\@tempswatrue\@fileswtrue
1255   \ifnextchar[\filecontents@opt\filecontents
1256 }
1257 \namedef{filecontents*}{\@tempswafalse\@fileswtrue
1258   \ifnextchar[\filecontents@opt\filecontents
1259 }

```

To handle the optional argument we execute for each option the command \filecontents@OPTION if it exist or complain about unknown option.

```

1260 \def\filecontents@opt[#1]{%
1261   \edef\@fortmp{\zap@space#1 \@empty}%
1262   \@for\reserved@a:=\@fortmp\do{%
1263     \ifcsname filecontents@\reserved@a\endcsname
1264       \csname filecontents@\reserved@a\endcsname
1265     \else
1266       \latex@error{Unknown filecontents option \reserved@a}%
1267       {Valid options are force (or overwrite), nosearch, noheader, nowarn}%
1268     \fi}%
1269   \filecontents
1270 }

```

Option `force` (or `overwrite`) changes the overwriting switch

```
1271 \let\filec@ntents@force\@fileswfalse
1272 \let\filec@ntents@overwrite\@fileswfalse % alternative name
```

and option `noheader` the preamble switch (which is equivalent to using the star form of the environment).

```
1273 \let\filec@ntents@noheader\@tempswafalse
```

Option `nosearch` only checks the current directory not the whole TeX tree for the existence of the file to write.

```
1274 \def\filec@ntents@nosearch{%
1275   \let\filec@ntents@checkdir\@currdir
1276   \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```
1277 \let\filec@ntents@checkdir\@empty
1278 \def\filec@ntents@where{exists on the system}
```

Option `nowarn` does not show any warning on the terminal but still writes it to the `.log`.

```
1279 \def\filec@ntents@nowarn{%
1280   \let\filec@ntents@warning\@latex@note@no@line
1281 }
```

By default we show terminal warnings.

```
1282 \let\filec@ntents@warning\@latex@warning@no@line
```

```
1283 \begingroup%
1284 \@tempcnta=1
1285 \loop
1286   \catcode\@tempcnta=12 %
1287   \advance\@tempcnta\@ne %
1288   \ifnum\@tempcnta<32 %
1289     \repeat %
1290   \catcode'\*=11 %
1291   \catcode'\^M\active%
1292   \catcode'\^L\active\let^L\relax%
1293   \catcode'\^I\active%
```

```
1294 \gdef\filec@ntents#1{%
1295   \set@curr@file{\filec@ntents@checkdir#1}%
1296   \edef\q@curr@file{"\@curr@file"}%
```

LuaTeX has more writes (and 18 is safe here).

```
1297 \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1298 \openin\@inputcheck\q@curr@file \space %
1299 \ifeof\@inputcheck%
1300   \@latex@note@no@line%
1301   {Writing file '\@currdir\@curr@file'}%
1302   \ch@ck7\reserved@c\write\relax%
1303   \immediate\openout\reserved@c\q@curr@file\relax%
1304 \else%
```

```

1305 \if@filesw%
1306 \latex@note@no@line%
1307 {File '@curr@file' already \filec@tents@where.\MessageBreak%
1308 Not generating it from this source}%
1309 \let\write\@gobbletwo%
1310 \let\closeout\@gobble%
1311 \else%

```

If we are overwriting, we try to make sure that the user is not by mistake overwriting the input file (\jobname). Of course, this only works for input files ending in .tex. If a different extension is used there is no way to see that we are overwriting ourselves!

```

1312 \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1313 \ifx\@curr@file\reserved@b%
1314 \@fileswtrue%
1315 \else%
1316 \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1317 \ifx\@curr@file\reserved@b
1318 \@fileswtrue%
1319 \fi%
1320 \fi%

```

We allocate a write channel but we open it only if it is (hopefully) safe. If not opened that means we are going to write on the terminal.

```

1321 \ch@ck7\reserved@c\write\relax%
1322 \if@filesw% % Foul ... trying to overwrite \jobname!
1323 \latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1324 write to the file you are reading from!\MessageBreak%
1325 Data is written to screen instead.}%
1326 \else%
1327 \filec@tents@warning%
1328 {Writing or overwriting file '@curr@file'}%
1329 \immediate\openout\reserved@c\q@curr@file\relax%
1330 \fi%
1331 \fi%
1332 \fi%

```

Closing the \inputcheck is done here to avoid having to do this in each branch.

```

1333 \closein\inputcheck%
1334 \if@tempswa%
1335 \immediate\write\reserved@c{%
1336 \@percentchar\@percentchar\space%
1337 \expandafter\@gobble\string\LaTeX2e file '@curr@file'^~J%
1338 \@percentchar\@percentchar\space generated by the %
1339 '@curr@envir' \expandafter\@gobblefour\string\newenvironment^^J%
1340 \@percentchar\@percentchar\space from source '\jobname' on %
1341 \number\year/\two@digits\month/\two@digits\day.^~J%
1342 \@percentchar\@percentchar}%
1343 \fi%
1344 \let\do\@makeother\dospecials%

```

If there are active characters in the upper half (e.g., from inputenc) there would be confusion so we render everything harmless.

```

1345 \count@ 128\relax%
1346 \loop%

```



```

1347 \catcode\count@ 11\relax%
1348 \advance\count@ \@ne%
1349 \ifnum\count@<\@ccclvi%
1350 \repeat%

1351 \edef\E{\@backslashchar end\string{\@currentvir\string}}%
1352 \edef\reserved@b{%
1353 \def\noexpand\reserved@b%
1354 #####1\E####2\E####3\relax}%
1355 \reserved@b{%
1356 \ifx\relax##3\relax%

There was no \end{filecontents}

1357 \immediate\write\reserved@c{##1}%
1358 \else%

There was a \end{filecontents}, so stop this time.

1359 \edef^^M{\noexpand\end{\@currentvir}}%
1360 \ifx\relax##1\relax%
1361 \else%

Text before the \end, write it with a warning.

1362 \@latex@warning{Writing text ‘##1’ before %
1363 \string\end{\@currentvir}\MessageBreak
1364 as last line of \@curr@file}%
1365 \immediate\write\reserved@c{##1}%
1366 \fi%
1367 \ifx\relax##2\relax%
1368 \else%

Text after the \end, ignore it with a warning.

1369 \@latex@warning{%
1370 Ignoring text ‘##2’ after \string\end{\@currentvir}}%
1371 \fi%
1372 \fi%
1373 ^^M}%

1374 \catcode‘^^L\active%
1375 \let\L\@undefined%
1376 \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1377 \catcode‘^^I\active%
1378 \let\I\@undefined%
1379 \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1380 \catcode‘^^M\active%
1381 \edef^^M##1^^M{%
1382 \noexpand\reserved@b##1\E\E\relax}}%
1383 \endgroup%

1384 </2ekernel | latexrelease>
1385 <latexrelease>\EndIncludeInRelease
1386 <latexrelease>\IncludeInRelease{2019/10/01}%
1387 <latexrelease> \{filec@ntents\}{Spaces in file names + optional arg}%
1388 <latexrelease>
1389 <latexrelease>\def\filecontents{\@tempwatrue\@fileswtrue
1390 <latexrelease> \@ifnextchar[\filec@ntents@opt\filec@ntents
1391 <latexrelease>}
1392 <latexrelease>\@namedef{filecontents*}{\@tempwafalse\@fileswtrue

```

```

1393 <latexrelease> \@ifnextchar[\filec@ntents@opt\filec@ntents
1394 <latexrelease>}
1395 <latexrelease>\def\filec@ntents@opt[#1]{%
1396 <latexrelease> \edef\@fortmp{\zap@space#1 \@empty}%
1397 <latexrelease> \@for\reserved@a:=\@fortmp\do{%
1398 <latexrelease> \ifcsname filec@ntents@\reserved@a\endcsname
1399 <latexrelease> \csname filec@ntents@\reserved@a\endcsname
1400 <latexrelease> \else
1401 <latexrelease> \@latex@error{Unknown filecontents option \reserved@a}%
1402 <latexrelease> {Valid options are force (or overwrite), nosearch, noheader}%
1403 <latexrelease> \fi}%
1404 <latexrelease> \filec@ntents
1405 <latexrelease>}
1406 <latexrelease>\let\filec@ntents@force\@fileswfalse
1407 <latexrelease>\let\filec@ntents@overwrite\@fileswfalse % alternative name
1408 <latexrelease>\let\filec@ntents@noheader\@tempswafalse
1409 <latexrelease>\def\filec@ntents@nosearch{%
1410 <latexrelease> \let\filec@ntents@checkdir\@currdir
1411 <latexrelease> \def\filec@ntents@where{in current directory}}
1412 <latexrelease>\let\filec@ntents@checkdir\@empty
1413 <latexrelease>\def\filec@ntents@where{exists on the system}
1414 <latexrelease>\begingroup%
1415 <latexrelease>\@tempcnta=1
1416 <latexrelease>\loop
1417 <latexrelease> \catcode\@tempcnta=12 %
1418 <latexrelease> \advance\@tempcnta\@one %
1419 <latexrelease> \ifnum\@tempcnta<32 %
1420 <latexrelease> \repeat %
1421 <latexrelease>\catcode'\*=11 %
1422 <latexrelease>\catcode'\^M\active%
1423 <latexrelease>\catcode'\^L\active\let\^L\relax%
1424 <latexrelease>\catcode'\^I\active%
1425 <latexrelease>\gdef\filec@ntents#1{%
1426 <latexrelease> \set@curr@file{\filec@ntents@checkdir#1}%
1427 <latexrelease> \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
1428 <latexrelease> \chardef\reserved@c\ifx\directlua\undefined 15 \else 127 \fi%
1429 <latexrelease> \openin\@inputcheck\q@curr@file \space %
1430 <latexrelease> \ifeof\@inputcheck%
1431 <latexrelease> \@latex@warning@no@line%
1432 <latexrelease> {Writing file '\@currdir\@curr@file'}%
1433 <latexrelease> \ch@ck7\reserved@c\write\relax%
1434 <latexrelease> \immediate\openout\reserved@c\q@curr@file\relax%
1435 <latexrelease> \else%
1436 <latexrelease> \if@filesw%
1437 <latexrelease> \@latex@warning@no@line%
1438 <latexrelease> {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1439 <latexrelease> Not generating it from this source}%
1440 <latexrelease> \let\write\@gobbletwo%
1441 <latexrelease> \let\closeout\@gobble%
1442 <latexrelease> \else%
1443 <latexrelease> \edef\reserved@a{#1}%
1444 <latexrelease> \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1445 <latexrelease> \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1446 <latexrelease> \ifx\reserved@a\reserved@b%

```

```

1447 <latexrelease> \fileswtrue%
1448 <latexrelease> \else%
1449 <latexrelease> \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1450 <latexrelease> \ifx\reserved@a\reserved@b
1451 <latexrelease> \fileswtrue%
1452 <latexrelease> \fi%
1453 <latexrelease> \fi%
1454 <latexrelease> \ch@ck7\reserved@c\write\relax%
1455 <latexrelease> \if@filesw% % Foul ... trying to overwrite \jobname!
1456 <latexrelease> \@latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1457 <latexrelease> write to the file you are reading from!\MessageBreak%
1458 <latexrelease> Data is written to screen instead.}%
1459 <latexrelease> \else%
1460 <latexrelease> \latex@warning@no@line%
1461 <latexrelease> {Writing or overwriting file '\currdir\curr@file'}%
1462 <latexrelease> \immediate\openout\reserved@c\q\curr@file\relax%
1463 <latexrelease> \fi%
1464 <latexrelease> \fi%
1465 <latexrelease> \fi%
1466 <latexrelease> \closein\@inputcheck%
1467 <latexrelease> \if@tempswa%
1468 <latexrelease> \immediate\write\reserved@c{%
1469 <latexrelease> \@percentchar\@percentchar\space%
1470 <latexrelease> \expandafter\@gobble\string\LaTeX2e file '\curr@file'^^J%
1471 <latexrelease> \@percentchar\@percentchar\space generated by the %
1472 <latexrelease> '\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1473 <latexrelease> \@percentchar\@percentchar\space from source '\jobname' on %
1474 <latexrelease> \number\year/\two@digits\month/\two@digits\day.^^J%
1475 <latexrelease> \@percentchar\@percentchar}%
1476 <latexrelease> \fi%
1477 <latexrelease> \let\do\@makeoother\dospecials%
1478 <latexrelease> \count@ 128\relax%
1479 <latexrelease> \loop%
1480 <latexrelease> \catcode\count@ 11\relax%
1481 <latexrelease> \advance\count@ \@ne%
1482 <latexrelease> \ifnum\count@<\@cclvi%
1483 <latexrelease> \repeat%
1484 <latexrelease> \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1485 <latexrelease> \edef\reserved@b{%
1486 <latexrelease> \def\noexpand\reserved@b%
1487 <latexrelease> #####1\E####2\E####3\relax}%
1488 <latexrelease> \reserved@b{%
1489 <latexrelease> \ifx\relax##3\relax%
1490 <latexrelease> \immediate\write\reserved@c{##1}%
1491 <latexrelease> \else%
1492 <latexrelease> \edef^^M{\noexpand\end{\@currenvir}}%
1493 <latexrelease> \ifx\relax##1\relax%
1494 <latexrelease> \else%
1495 <latexrelease> \latex@warning{Writing text '##1' before %
1496 <latexrelease> \string\end{\@currenvir}\MessageBreak as last line of \curr@file}%
1497 <latexrelease> \immediate\write\reserved@c{##1}%
1498 <latexrelease> \fi%
1499 <latexrelease> \ifx\relax##2\relax%
1500 <latexrelease> \else%

```

```

1501 <latexrelease> \@latex@warning{%
1502 <latexrelease> Ignoring text ‘##2’ after \string\end{\@currentvir}}%
1503 <latexrelease> \fi%
1504 <latexrelease> \fi%
1505 <latexrelease> ^^M}%
1506 <latexrelease> \catcode‘^^L\active%
1507 <latexrelease> \let\L\undefined%
1508 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1509 <latexrelease> \catcode‘^^I\active%
1510 <latexrelease> \let\I\undefined%
1511 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1512 <latexrelease> \catcode‘^^M\active%
1513 <latexrelease> \edef^^M##1^^M{%
1514 <latexrelease> \noexpand\reserved@b##1\E\E\relax}}%
1515 <latexrelease> \endgroup%
1516 <latexrelease> \EndIncludeInRelease
1517 <latexrelease> \IncludeInRelease{0000/00/00}%
1518 <latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1519 <latexrelease>
1520 <latexrelease> \let\filec@ntents@opt \@undefined
1521 <latexrelease> \let\filec@ntents@force \@undefined
1522 <latexrelease> \let\filec@ntents@overwrite \@undefined
1523 <latexrelease> \let\filec@ntents@noheader \@undefined
1524 <latexrelease> \let\filec@ntents@nosearch \@undefined
1525 <latexrelease> \let\filec@ntents@checkdir \@undefined
1526 <latexrelease> \let\filec@ntents@where \@undefined
1527 <latexrelease>
1528 <latexrelease> \begingroup%
1529 <latexrelease> \@tempcnta=1
1530 <latexrelease> \loop
1531 <latexrelease> \catcode\@tempcnta=12 %
1532 <latexrelease> \advance\@tempcnta\@ne %
1533 <latexrelease> \ifnum\@tempcnta<32 %
1534 <latexrelease> \repeat %
1535 <latexrelease> \catcode‘\*=11 %
1536 <latexrelease> \catcode‘^^M\active%
1537 <latexrelease> \catcode‘^^L\active\let^^L\relax%
1538 <latexrelease> \catcode‘^^I\active%
1539 <latexrelease>
1540 <latexrelease> \gdef\filec@ntents#1{%
1541 <latexrelease> \openin\@inputcheck#1 %
1542 <latexrelease> \ifeof\@inputcheck%
1543 <latexrelease> \@latex@warning@no@line%
1544 <latexrelease> {Writing file ‘\@currdir#1’}%
1545 <latexrelease> \chardef\reserved@c15 %
1546 <latexrelease> \ch@ck7\reserved@c\write%
1547 <latexrelease> \immediate\openout\reserved@c#1\relax%
1548 <latexrelease> \else%
1549 <latexrelease> \closein\@inputcheck%
1550 <latexrelease> \@latex@warning@no@line%
1551 <latexrelease> {File ‘#1’ already exists on the system.\MessageBreak%
1552 <latexrelease> Not generating it from this source}%
1553 <latexrelease> \let\write\@gobbletwo%
1554 <latexrelease> \let\closeout\@gobble%

```

```

1555 <latexrelease> \fi%
1556 <latexrelease> \if@tempswa%
1557 <latexrelease> \immediate\write\reserved@c{%
1558 <latexrelease> \@percentchar\@percentchar\space%
1559 <latexrelease> \expandafter\@gobble\string\LaTeXe file '#1'^^J%
1560 <latexrelease> \@percentchar\@percentchar\space generated by the %
1561 <latexrelease> '@\currentenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1562 <latexrelease> \@percentchar\@percentchar\space from source '\jobname' on %
1563 <latexrelease> \number\year/\two@digits\month/\two@digits\day.^^J%
1564 <latexrelease> \@percentchar\@percentchar}%
1565 <latexrelease> \fi%
1566 <latexrelease> \let\do\@makeother\dospecials%
1567 <latexrelease> \count@ 128\relax%
1568 <latexrelease> \loop%
1569 <latexrelease> \catcode\count@ 11\relax%
1570 <latexrelease> \advance\count@ \@ne%
1571 <latexrelease> \ifnum\count@<\@ccclvi%
1572 <latexrelease> \repeat%
1573 <latexrelease> \edef\E{\@backslashchar end\string{\@currentenvir\string}}%
1574 <latexrelease> \edef\reserved@b{%
1575 <latexrelease> \def\noexpand\reserved@b%
1576 <latexrelease> #####1\E####2\E####3\relax}%
1577 <latexrelease> \reserved@b{%
1578 <latexrelease> \ifx\relax##3\relax%
1579 <latexrelease> \immediate\write\reserved@c{##1}%
1580 <latexrelease> \else%
1581 <latexrelease> \edef^^M{\noexpand\end{\@currentenvir}}%
1582 <latexrelease> \ifx\relax##1\relax%
1583 <latexrelease> \else%
1584 <latexrelease> \@latex@warning{Writing text '#1' before %
1585 <latexrelease> \string\end{\@currentenvir}\MessageBreak as last line of #1}%
1586 <latexrelease> \immediate\write\reserved@c{##1}%
1587 <latexrelease> \fi%
1588 <latexrelease> \ifx\relax##2\relax%
1589 <latexrelease> \else%
1590 <latexrelease> \@latex@warning{%
1591 <latexrelease> Ignoring text '##2' after \string\end{\@currentenvir}}%
1592 <latexrelease> \fi%
1593 <latexrelease> \fi%
1594 <latexrelease> ^^M}%
1595 <latexrelease>
1596 <latexrelease> \catcode'\^^L\active%
1597 <latexrelease> \let\L\@undefined%
1598 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1599 <latexrelease> \catcode'\^^I\active%
1600 <latexrelease> \let\I\@undefined%
1601 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1602 <latexrelease> \catcode'\^^M\active%
1603 <latexrelease> \edef^^M##1^^M{%
1604 <latexrelease> \noexpand\reserved@b##1\E\E\relax}}%
1605 <latexrelease> \endgroup%
1606 <latexrelease> \EndIncludeInRelease
1607 <*2ekernel>
1608 \begingroup

```

```

1609 \catcode'\=\catcode'\%
1610 \catcode'\%=12
1611 \catcode'\*=11
1612 \gdef\@percentchar{%}
1613 \gdef\endfilecontents{
1614   \immediate\closeout\reserved@
1615   \def\T##1##2##3{
1616     \ifx##1\@undefined\else
1617       \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
1618     \fi}|
1619   \T\L{Form Feed}{Lin}|
1620   \T\I{Tab}{Spac}|
1621   \immediate\write\@unused{}}
1622 \global\let\endfilecontents*\endfilecontents

```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```

1623 %\@onlypreamble\filecontents
1624 %\@onlypreamble\endfilecontents
1625 %\@onlypreamble\filecontents*
1626 %\@onlypreamble\endfilecontents*
1627 \endgroup
1628 %\@onlypreamble\filecontents
(End of definition for \filecontents and \endfilecontents.)

```

5 Package/class rollback mechanism

```

1629 </2ekernel>
1630 <*2ekernel | latexreleasefirst>

```

`\pkgcls@debug` For testing we have a few extra lines of code that by default do nothing but one can set `\pkgcls@debug` to `\typeout` to get extra info. Sometime in the future this will be dropped.

```

1631 <*tracerollback>
1632 %\let\pkgcls@debug\typeout
1633 \let\pkgcls@debug\@gobble
1634 </tracerollback>

```

(End of definition for \pkgcls@debug.)

`\requestedLaTeXdate` The macro (!) `\requestedLaTeXdate` holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```

1635 \def\requestedLaTeXdate{0}

```

(End of definition for \requestedLaTeXdate.)

`\pkgcls@targetdate` If a rollback for a package or class is requested then `\pkgcls@targetdate` holds the requested date as a number YYYYMMDD (if there was one, otherwise the value of `\requestedLaTeXdate`) and `\pkgcls@targetlabel` will be empty. If there was a request for a named version then `\pkgcls@targetlabel` holds the version name and `\pkgcls@targetdate` is set to 1.

`\pkgcls@targetdate=0` is used to indicate that there was no rollback request. While loading an old release `\pkgcls@targetdate` is also reset to zero so that `\DeclareRelease` declarations are bypassed.

In contrast `\pkgcls@innerdate` will always hold the requested date (in a macro not a counter) if there was one, otherwise, e.g., if there was no request or a request to a version name it will contain TeX largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

In `\pkgcls@outerdate` we save the inner date before it is changed to a specific target date. That value is then put on the file stack and retrieved when the file has been fully read. It is then used again for `\pkgcls@innerdate` for the remainder of the currently open file.

```
1636 \ifx\pkgcls@targetdate\undefined
1637   \newcount\pkgcls@targetdate
1638 \fi
1639 \let\pkgcls@targetlabel\empty
1640 \def\pkgcls@innerdate{\maxdimen}

1641 \let\pkgcls@outerdate\pkgcls@innerdate

(End of definition for \pkgcls@targetdate and others.)
```

`\pkgcls@candidate` When looping through the `\DeclareRelease` declarations we record if the release is the best candidate we have seen so far. This is recorded in `\pkgcls@candidate` and we update it whenever we see a better one.

In `\pkgcls@releasedate` we keep track of the release date of that candidate.

```
1642 \let\pkgcls@candidate\empty
1643 \let\pkgcls@releasedate\empty

(End of definition for \pkgcls@candidate and \pkgcls@releasedate.)
```

`\load@onefilewithoptions` the best place to add the rollback code is at the point where `\@onefilewithoptions` is called to load a single class or package.

To make things easy we save the old definition as `\load@onefilewithoptions` and then provide a new interface.

Important: as this code is also unconditionally placed into `latexrelease` we can only do this name change once otherwise both macros will contain the same code.

```
1644 \ifx\load@onefilewithoptions\undefined
1645   \let\load@onefilewithoptions\@onefilewithoptions
1646   \def\@onefilewithoptions#1[#2][#3]#4{%
```

First a bit of tracing normally disabled.

```
1647 < *tracerollback>
1648   \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
1649   \pkgcls@debug{\@spaces 1: #1}%
1650   \pkgcls@debug{\@spaces 2: #2}%
1651   \pkgcls@debug{\@spaces 3: #3}%
1652   \pkgcls@debug{\@spaces 4: #4}%
1653 < /tracerollback>
```

Three of the arguments are needed later on in error/warning messages so we save them.

```
1654   \def\pkgcls@name{#1}%           % for info message
1655   \def\pkgcls@arg {#3}%           % for info message
1656   \edef\pkgcls@ext{%
1657     \ifx#4\@clsextension document class\else
1658       \ifx#4\@pkgextension package\else
1659         file
```

```

1660     \fi
1661     \fi
1662     }%                                % for info message

```

then we parse the final optional argument to determine if there is a specific rollback request for the current file. This will set `\pkgcls@targetdate`, `\pkgcls@targetlabel` and `\pkgcls@mindate`.

```

1663     \pkgcls@parse@date@arg{#3}%

```

When determining the correct release to load we keep track of candidates in `\pkgcls@candidate` and initially we don't have any:

```

1664     \let\pkgcls@candidate\@empty

```

If we had a rollback request then #3 may contain data but not necessarily a “minimal date” so instead of passing it on we pass on the content of `\pkgcls@mindate`. We need to pass the value not the command, otherwise nested packages may pick up the wrong information.

```

1665     \begingroup
1666     \edef\reserved@a{%
1667         \endgroup
1668         \unexpanded{\load@onefilewithoptions#1[#2]}%
1669         [\pkgcls@mindate]%
1670         \unexpanded{#4}}%
1671     \reserved@a
1672 }
1673 \fi

```

(End of definition for `\load@onefilewithoptions` and `\@onefilewithoptions`.)

`\pkgcls@parse@date@arg` The `\pkgcls@parse@date@arg` command parses the second optional argument of `\usepackage`, `\RequirePackage` or `\documentclass` for a rollback request setting the values of `\pkgcls@targetdate` and `\pkgcls@targetlabel`.

This optional argument has a dual purpose: If it just contains a date string then this means that the package should have at least that date (to ensure that a certain feature is actually available, or a certain bug has been fixed). When the package gets loaded the information in `\Provides...` will then be checked against this request.

But if it starts with an equal sign followed by a date string or followed by a version name then this means that we should roll back to the state of the package at that date or to the version with the requested name.

If there was no optional argument or the optional argument does not start with “=” then the `\pkgcls@targetdate` is set to the date of the overall rollback request (via `latexrelease`) or if that was not given it is set to 0. In either case `\pkgcls@targetlabel` will be made empty.

If the argument doesn't start with “=” then it is supposed to be a “minimal date” and we therefore save the value in `\pkgcls@mindate`, otherwise this macro is made empty.

So in summary we have:

Input	<code>\pkgcls@targetdate</code>	<code>\pkgcls@targetlabel</code>	<code>\pkgcls@mindate</code>
<code><empty></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><empty></code>
<code><date></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><date></code>
<code>=<date></code>	<code><date-as-number></code>	<code><empty></code>	<code><empty></code>
<code>=<version></code>	1	<code><version></code>	<code><empty></code>
<code><other></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><other></code>

where `\global-rollbackdate-as-number` is a date request given via `latexrelease` or if there wasn't one 0.

```
1674 \def\pkgcls@parse@date@arg #1{%
```

If the argument is empty we use the rollback date from `latexrelease` which has the value of zero if there was no rollback request. The label and the minimal date is made empty in that case.

```
1675 \ifx\@nil#1\@nil
1676 \pkgcls@targetdate\requestedLaTeXdate\relax
1677 \let\pkgcls@targetlabel\@empty
1678 \let\pkgcls@mindate\@empty
```

Otherwise we parse the argument further, checking for a = as the first character. We append a = at the end so that there is at least one such character in the argument.

```
1679 \else
1680 \pkgcls@parse@date@arg#1=\@nil\relax
1681 \fi
1682 }
```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```
1683 \def\pkgcls@parse@date@arg@#1=#2\@nil{%
```

We set `\pkgcls@targetdate` depending on the parsing result; the code is expandable so we can do the parsing as part of the assignment.

```
1684 \pkgcls@targetdate
```

If a = was in first position then #1 will be empty. In that case #2 will be the original argument with a = appended.

This can be parsed with `\@parse@version`, the trailing character is simply ignored. This macro returns the parsed date as a number (or zero if it wasn't a date) and accepts both YYYY/MM/DD and YYYY-MM-DD formats.

```
1685 \ifx\@nil#1\@nil
1686 \@parse@version0#2//00\@nil\relax
```

Whatever is returned is thus assigned to `\pkgcls@targetdate` and therefore we can now test its value. If the value is zero we assume that the remaining argument string represents a version and change `\pkgcls@targetdate` and set `\pkgcls@targetlabel` to the version name (after stripping off the trailing =).

```
1687 \ifnum \pkgcls@targetdate=\z@
1688 \pkgcls@targetdate\@ne
1689 \def\pkgcls@innerdate{\maxdimen}%
1690 \pkgcls@parse@date@arg@version#2%
1691 \else
```

Save the `\pkgcls@innerdate` so that it can be placed on the file stack later.

```
1692 \let\pkgcls@outerdate\pkgcls@innerdate
1693 \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1694 \fi
1695 \let\pkgcls@mindate\@empty
1696 \else
```

If #1 was not empty then there wasn't a = character in first position so we are dealing either with a "minimum date" or with some incorrect data. We assume the former and make the following assignments (the first one finishing the assignment of `\pkgcls@targetdate`):

```

1697 \requestedLaTeXdate\relax
1698 \let\pkgcls@targetlabel\@empty
1699 \def\pkgcls@innerdate{\maxdimen}%
1700 \def\pkgcls@mindate{#1}%

```

If the min-date is after the requested rollback date (if there is any, i.e., if it is not zero) then we have a conflict and therefore issue a warning.

```

1701 \ifnum \pkgcls@targetdate > \z@
1702 \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
1703 \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
1704 A minimal date of #1 has been specified for
1705 \pkgcls@ext\MessageBreak '\pkgcls@name'.\MessageBreak
1706 But this is in conflict
1707 with a rollback request to \requestedpatchdate}
1708 \fi
1709 \fi
1710 \fi
1711 }

```

Strip off the trailing = and assign the version name to \pkgcls@targetlabel.

```

1712 \def\pkgcls@parse@date@arg@version#1={%
1713 \def\pkgcls@targetlabel{#1}}

```

(End of definition for \pkgcls@parse@date@arg.)

\DeclareRelease First argument is the “name” of the release and it can be left empty if one doesn’t like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external file name of this release, by convention this should be `<pkg/cls-name>-<date>.<extension>` but this is not enforced and through this argument one can overwrite it.

```

1714 \def\DeclareRelease#1#2#3{%
1715 \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1716 <*tracerollback>
1717 \pkgcls@debug{---\string\DeclareRelease:}%
1718 \pkgcls@debug{\@spaces 1: #1}%
1719 \pkgcls@debug{\@spaces 2: #2}%
1720 \pkgcls@debug{\@spaces 3: #3}%
1721 </tracerollback>

```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a “beta” release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```

1722 \ifx\@nil#2\@nil
1723 \ifnum\pkgcls@targetdate=\@ne % named request
1724 \def\reserved@a{#1}%
1725 \ifx\pkgcls@targetlabel\reserved@a
1726 \pkgcls@use@this@release{#3}{}%
1727 <*tracerollback>
1728 \else
1729 \pkgcls@debug{Label doesn't match}%
1730 </tracerollback>
1731 \fi
1732 <*tracerollback>

```

```

1733     \else
1734     \pkgcls@debug{Date request: ignored}%
1735 \tracerollback)
1736     \fi
1737 \else

```

If the value of `\pkgcls@targetdate` is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```

1738     \ifnum\pkgcls@targetdate>\@ne % a real request

```

So we parse the date of this release to check if it is before or after the request date.

```

1739     \ifnum\@parse@version#2//00\@nil
1740     >\pkgcls@targetdate

```

If it is after we have to distinguish between two cases: If there was an earlier candidate we use that one because the other is too late, but if there wasn't one (i.e., if current release is the oldest that exists) we use it as the best choice. However in that case something is wrong (as there shouldn't be a rollback to a date when a package used didn't yet exists). So we make a complained to the user.

```

1741     \ifx\pkgcls@candidate\@empty
1742     \pkgcls@rollbackdate@error{#2}%
1743     \pkgcls@use@this@release{#3}{#2}%
1744     \else
1745     \pkgcls@use@this@release\pkgcls@candidate
1746     \pkgcls@releasedate
1747     \fi
1748 \else

```

Otherwise, if the release date of this version is before the target rollback and we record it as a candidate. But we don't use it yet as there may be another release which is still before the target rollback.

```

1749     \def\pkgcls@candidate{#3}%
1750     \def\pkgcls@releasedate{#2}%
1751 (*tracerollback)
1752     \pkgcls@debug{New candidate: #3}%
1753 \tracerollback)
1754     \fi
1755 \else

```

If we end up in this branch we have a named version request. So we check if `\pkgcls@targetlabel` matches the current name and if yes we use this release immediately, otherwise we do nothing as a later declaration may match it.

```

1756     \def\reserved@a{#1}%
1757     \ifx\pkgcls@targetlabel\reserved@a
1758     \pkgcls@use@this@release{#3}{#2}%
1759 (*tracerollback)
1760     \else
1761     \pkgcls@debug{Label doesn't match}%
1762 \tracerollback)
1763     \fi
1764     \fi
1765     \fi
1766     \fi
1767 }

```

(End of definition for \DeclareRelease.)

`\pkgcls@use@this@release` If a certain release has been selected (stored in the external file given in #1) we need to input it and afterwards stop reading the current file.

```
1768 \def\pkgcls@use@this@release#1#2{%
```

Before that we record the selection made inside the transcript.

```
1769 \pkgcls@show@selection{#1}{#2}%
```

We then set the `\pkgcls@targetdate` to zero so that any `\DeclareRelease` or `\DeclareCurrentRelease` in the file we now load are bypassed⁵² and then we finally load the correct release.

After loading that file we need to stop reading the current file so we issue `\endinput`. Note that the `\relax` before that is essential to ensure that the `\endinput` is only happening after the file has been fully processed, otherwise it would act after the first line of the `\@@input`!

```
1770 \pkgcls@targetdate\z@
```

```
1771 \@addtofilelist{#1}%
```

```
1772 \@@input #1\relax
```

```
1773 \endinput
```

```
1774 }
```

(End of definition for `\pkgcls@use@this@release`.)

`\pkgcls@show@selection` This command records what selection was made. As that is needed in two places (and it is rather lengthy) it was placed in a separate command. The first argument is the name of the external file that is being loaded and is only needed for debugging. The second argument is the date that corresponds to this file and it is used as part of the message.

```
1775 \def\pkgcls@show@selection#1#2{%
```

```
1776 \*traceroollback)
```

```
1777 \pkgcls@debug{Result: use #1}%
```

```
1778 \*traceroollback)
```

```
1779 \GenericInfo
```

```
1780 {\@spaces\@spaces\space}{Rollback for
```

```
1781 \@cls@pkg\space'\@currname' requested ->
```

```
1782 \ifnum\pkgcls@targetdate>\@ne
```

```
1783 date
```

```
1784 \ifnum\requestedLaTeXdate=\pkgcls@targetdate
```

```
1785 \requestedpatchdate
```

```
1786 \else
```

```
1787 \expandafter\@gobble\pkgcls@arg
```

```
1788 \fi.\MessageBreak
```

Instead of “best approximation” we could say that we have been able to exactly match the date (if it is exact), but that would mean extra tests without much gain, so not done.

```
1789 Best approximation is
```

```
1790 \else
```

```
1791 version '\pkgcls@targetlabel'.\MessageBreak
```

```
1792 This corresponds to
```

```
1793 \fi
```

```
1794 \ifx\@nil#2\@nil
```

```
1795 a special release%
```

```
1796 \else
```

⁵²The older release may also have such declarations inside if it was a simply copy of the `.sty` or `.cls` file current at that date. Removing these declarations would make the file load a tiny bit faster, but this way it works in any case.

```

1797         the release introduced on #2%
1798     \fi
1799     \@gobble}%
1800 }

```

(End of definition for \pkgcls@show@selection.)

\pkgcls@rollbackdate@error This is called if the requested rollback date is earlier than the earliest known release of a package or class.

A similar error is given if global rollback date and min-date on a specific package conflict with each other, but that case is happens only once so it is inlined.

```

1801 \def\pkgcls@rollbackdate@error#1{%
1802   \@latex@error{Suspicious rollback date given}%
1803   {The \cls@pkg\space'\@currname' has no rollback data
1804     before #1 which\MessageBreak
1805     is after your requested rollback date --- so
1806     something may be wrong here.\MessageBreak
1807     Continue and we use the earliest known release.}}

```

(End of definition for \pkgcls@rollbackdate@error.)

\DeclareCurrentRelease This declares the date (and possible name) of the current version of a package or class.

```

1808 \def\DeclareCurrentRelease#1#2{%

```

First we test if \pkgcls@targetdate is greater than zero, otherwise this code is bypassed (as there is no rollback request).

```

1809   \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1810   \traceroollback
1811     \pkgcls@debug{---DeclareCurrentRelease}%
1812     \pkgcls@debug{ 1: #1}%
1813     \pkgcls@debug{ 2: #2}%
1814   \traceroollback

```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with \pkgcls@targetdate.

```

1815     \ifnum\pkgcls@targetdate>\@ne % a date request
1816     \ifnum\@parse@version#2//00\@nil
1817       >\pkgcls@targetdate

```

If it is greater that means the release date if this file is later than the requested rollback date. Again we have two cases: If there was a previous candidate release we use that one as the current release is too young, but if there wasn't we have to use this release nevertheless as there isn't any alternative.

However this case can only happen if there is a \DeclareCurrentRelease but no declared older releases (so basically the use of the declaration is a bit dubious).

```

1818     \ifx\pkgcls@candidate\@empty
1819       \pkgcls@rollbackdate@error{#2}%
1820     \else
1821       \pkgcls@use@this@release\pkgcls@candidate
1822       \pkgcls@releasedate
1823     \fi

```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```

1824     \else
1825         \pkgcls@show@selection{current version}{#2}%
1826     \fi
1827     \else % a label request

```

Otherwise we have a rollback request to a named version so we check if that fits the current name and if not give an error as this was the last possible opportunity.

```

1828     \def\reserved@a{#1}%
1829     \ifx\pkgcls@targetlabel\reserved@a
1830         \pkgcls@show@selection{current version}{#2}%
1831     \else
1832         \@latex@error{Requested version '\pkgcls@targetlabel' for
1833             \cls@pkg\space'\@currname' is unknown}\@ehc
1834     \fi
1835 \fi
1836 \fi
1837 }

```

(End of definition for \DeclareCurrentRelease.)

\IfTargetDateBefore This enables a simple form of conditional code inside a class or package file. If there is a date request and the request date is earlier than the first argument the code in the second argument is processed otherwise the code in the third argument is processed. If there was no date request then we also execute the third argument, i.e., we will get the “latest” version of the file.

Most often the second argument (before-date-code) will be empty.

```

1838 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1839     \ifnum\pkgcls@innerdate <%
1840         \expandafter\@parse@version\expandafter0#1//00\@nil
1841         \typeout{Exclude code introduced on #1}%
1842         \expandafter\@firstoftwo
1843     \else
1844         \typeout{Include code introduced on #1}%
1845         \expandafter\@secondoftwo
1846     \fi
1847 }

```

(End of definition for \IfTargetDateBefore.)

```

1848 </2kernel | latexreleasefirst>

```

6 After Preamble

Finally we declare a package that allows all the commands declared above to be `\onlypreamble` to be used after `\begin{document}`.

```

1849 <*afterpreamble>
1850 \NeedsTeXFormat{LaTeX2e}
1851 \ProvidesPackage{pkgindoc}
1852     [2020-08-08 v1.3m Package Interface in Document (DPC)]
1853 \def\reserved@a#1\do\@classoptionslist#2\do\filecontents#3\relax{%
1854     \gdef\@preamblecmds{#1#3}}

```

```
1855 \expandafter\reserved@a\@preamblecmds\relax
1856 \afterpreamble>
```

File 51

ltkeys.dtx

1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts: creating the key options and setting (using) them. Options created in this way *may* be used after package loading as general key–value settings: this will depend on the nature of the underlying code.

`\DeclareKeys` `\DeclareKeys [<family>] {<declarations>}`

Creates a series of options from a comma-separated *<declarations>* list. The *<family>* is a namespace for the keys; if not given, the basename of the current class or package is used. Each entry in the *<declarations>* (a commas-separated list) is a key–value pair, with the *<key>* having one or more *<properties>*. A small number of “basic” *<properties>* are described below. The full range of properties, provided by `l3keys`, can also be used for more powerful processing. See `interface3` for the full details.

The basic properties provided here are

- `.code` — execute arbitrary code
- `.if` — sets a TeX `\if...` switch
- `.ifnot` — sets an inverted TeX `\if...` switch
- `.pass-to-packages` — for class options, this specifies whether the option should be treated “global” (read by packages from the global list); for package options this property has no effect
- `.store` — stores a value in a macro
- `.usage` — defines whether the option can be given only when loading (`load`), in the preamble (`preamble`) or has no limitation on scope (`general`)

The part of the *<key>* before the *<property>* is the *<name>*, with the *<value>* working with the *<property>* to define the behaviour of the option.

For example, with

```
\DeclareKeys[mypkg]
{
  draft.if           = @mypkg@draft      ,
  draft.usage        = preamble          ,
  name.store         = \@mypkg@name      ,
  name.usage         = load              ,
  second-name.store = \@mypkg@other@name
}
```

three options would be create. The option `draft` can be given anywhere in the preamble, and will set a switch called `\if@mypkg@draft`. The option `name` can only be given during package loading, and will save whatever value it is given in `\@mypkg@name`. Finally, the option `second-name` can be given anywhere, and will save its value in `\@mypkg@other@name`.

Keys created *before* the use of `\ProcessKeyOptions` act as package options.

`\DeclareUnknownKeyHandler` `\DeclareUnknownKeyHandler [<family>] {<code>}`

The function `\DeclareUnknownKeyHandler` may be used to define the behavior when an undefined key is encountered. The *<code>* will receive the unknown key name as `#1` and the value as `#2`. These can then be processed as appropriate, e.g. by forwarding to another package.

\ProcessKeyOptions \ProcessKeyOptions [*<families>*]

The `\ProcessKeyOptions` function is used to check the current option list against the keys defined for *<families>* (a comma-separated list). Global (class) options and local (package) options are checked when this function is called in a package. Where there is more than one *<family>* in the list of *<families>*, processing takes place in the order of the *<families>*. Every key known in a *<family>* is passed for processing, thus a key defined in more than one *<family>* will be processed several times, and if the code paths are shared, the outcome will be determined by the last entry in the list of *<families>*.

\SetKeys \SetKeys [*<family>*] {*<keyvals>*}

Sets (applies) the explicit list of *<keyvals>* for the *<family>*: if the latter is not given, the value of `\@currname` is used. This command may be used within a package to set options before or after using `\ProcessKeyOptions`.

1.1 Implementation of `ltkeys`

```

1 <@=keys>
2 <*2ekernel>
3 \ExplSyntaxOn

```

1.2 Key properties

```

.code
  .if      4 \group_begin:
  .ifnot   5 \cs_set_protected:Npn \__keys_tmp:nn #1#2
  .store   6 {
  .usage   7 \quark_if_recursion_tail_stop:n {#1}
           8 \cs_new_eq:cc
           9 { \c__keys_props_root_str . #2 }
          10 { \c__keys_props_root_str . #1 }
          11 \__keys_tmp:nn
          12 }
          13 \__keys_tmp:nn
          14 { code:n } { code }
          15 { legacy_if_set:n } { if }
          16 { legacy_if_set_inverse:n } { ifnot }
          17 { tl_set:N } { store }
          18 { usage:n } { usage }
          19 { \q_recursion_tail } { }
          20 \q_recursion_stop
          21 \group_end:

```

(End of definition for `.code` and others.)

.pass-to-packages Used to force options to be global: as this property (uniquely) has an *optional* value, there is a bit of work to do.

```

\__keys_scope:n
\__keys_scope:N 22 \cs_new_protected:cpn { \c__keys_props_root_str .pass-to-packages }
                23 {
                24 \bool_if:NTF \l__keys_no_value_bool
                25 { \__keys_scope:n { true } }
                26 { \__keys_scope:n }

```

```

27 }
28 \cs_new_protected:Npn \__keys_scope:n #1
29 {
30   \str_case:nnF {#1}
31   {
32     { true }
33     { \__keys_scope:N \clist_put_right:NV }
34     { false }
35     { \__keys_scope:N \clist_remove_all:NV }
36   }
37   {
38     \msg_error:nnnn { keys }
39     { choice-unknown }
40     { .pass-to-packages }
41     {#1}
42   }
43 }
44 \cs_new_protected:Npn \__keys_scope:N #1
45 {
46   \exp_after:wN \__keys_find_key_module:wNN
47   \l_keys_path_str \s_keys_stop
48   \l_keys_key_tl \l_keys_key_str
49   #1 \l__keys_forced_global_clist \l_keys_key_str
50 }

```

(End of definition for `.pass-to-packages`, `\__keys_scope:n`, and `\__keys_scope:N`.)

1.3 Main mechanism

```

51 \cs_generate_variant:Nn \clist_if_in:NnT { Ne }
52 \cs_generate_variant:Nn \clist_if_in:NnTF { Ne }

```

`\l__keys_class_only_clist` Used to track class-only options.

```
53 \clist_new:N \l__keys_class_only_clist
```

(End of definition for `\l__keys_class_only_clist`.)

`\l__keys_forced_global_clist` Used to force options to be global.

```
54 \clist_new:N \l__keys_forced_global_clist
```

(End of definition for `\l__keys_forced_global_clist`.)

`\l__keys_options_clist` A single list is used for all options, into which they are collected before processing.

```
55 \clist_new:N \l__keys_options_clist
```

(End of definition for `\l__keys_options_clist`.)

`\l__keys_local_clist` Holds the local options when appropriate: otherwise empty. Needed as the L^AT_EX 2_ε setup here can be equal to `\scan_stop:`, which is not generally supported.

```
56 \clist_new:N \l__keys_local_clist
```

(End of definition for `\l__keys_local_clist`.)

`\l__keys_options_loading_bool`

Used to indicate we are in the loading phase: controls the outcome of warnings.

```
57 \bool_new:N \l__keys_options_loading_bool
```

```

__keys_options:n The main function calls functions to collect up the global and local options into \l__-
__keys_options_aux:n keys_options_clist before calling the underlying functions to actually do the pro-
cessing. So that a suitable message is produced if the option is unknown, the special
unknown key is set if it does not already exist for the current family, and is cleaned up
afterwards if required. To allow the LATEX 2ε layer to know this mechanism is active, and
to deal with the key family not matching the file name, we store the family in all cases.
Global options are only considered the first time a package is loaded: this is tracked using
opt@handler@\@currname.\@currentx, as this is defined once keyval processing has been
applied for the first time.

58 \cs_new_protected:Npn __keys_options:n #1
59 { __keys_options_expand_module:Nn __keys_options_aux:n {#1} }
60 \cs_new_protected:Npn __keys_options_aux:n #1
61 {
62   __keys_options_local:
63   \clist_map_inline:nn {#1}
64   {
65     \clist_clear:N \l__keys_options_clist
66     \cs_if_exist:cF { opt@handler@ \@currname . \@currentx }
67     { __keys_options_global:n {##1} }
68     __keys_options_local:n {##1}
69     \bool_set_true:N \l__keys_options_loading_bool
70     \clist_map_variable:NNn \l__keys_options_clist \CurrentOption
71     { \keys_set:nV {##1} \CurrentOption }
72     \bool_set_false:N \l__keys_options_loading_bool
73     __keys_options_loaded:n {##1}
74   }
75   \clist_if_empty:NF \l__keys_unused_clist
76   {
77     \clist_map_inline:Nn \l__keys_unused_clist
78     {
79       \msg_error:nnee { keys } { option-unknown }
80       {##1} { \@currname }
81     }
82   }
83   \cs_if_exist:cF { opt@handler@ \@currname . \@currentx }
84   {
85     \cs_gset_protected:cpn { opt@handler@ \@currname . \@currentx }
86     { \ProcessKeyOptions [ {#1} ] }
87   }
88   \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
89 }

90 \msg_new:nnnn { keys } { option-unknown }
91 { Unknown~option~'#1'~for~package~#2. }
92 {
93   LaTeX~has~been~asked~to~set~an~option~called~'#1'~
94   but~the~package~"\msg_module_name:n {#2}"~has~not~created~an~option~with~this~name.
95 }

```

(End of definition for __keys_options:n and __keys_options_aux:n.)

__keys_options_global:n Global (class) options are handled differently for L^AT_EX 2_ε packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as

L^AT_EX 2_ε allows variables to be equal to `\scan_stop:`, which is usually forbidden in expl3 code.

```

96 \cs_new_protected:Npn \__keys_options_global:n #1
97 {
98   \cs_if_eq:NNF \@raw@classoptionslist \scan_stop:
99   {
100     \cs_if_eq:NNTF \@currentx \crlsexextension
101     { \__keys_options_class:n {#1} }
102     { \__keys_options_package:n {#1} }
103   }
104 }

```

(End of definition for `\__keys_options_global:n`.)

```

\__keys_options_class:n
\__keys_options_class:nnn
\__keys_options_class:nn

```

For classes, each option (stripped of any content after =) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`. An earlier version of this code checked for the `unknown` key just once and if found short-cutted the loop: that though makes handling more complex situations harder, so we take the performance hit instead. Options used by classes are tracked but the catch-all `unknown` is excluded (hence not using a lazy evaluation for the key testing).

```

105 \cs_new_protected:Npn \__keys_options_class:n #1
106 {
107   \cs_if_free:cF { \@raw@opt@ \@currname . \@currentx }
108   {
109     \clist_map_inline:cn { \@raw@opt@ \@currname . \@currentx }
110     {
111       \exp_args:Ne \__keys_options_class:nnn
112       { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } } }
113     }
114   }
115 }
116 }
117 \cs_new_protected:Npn \__keys_options_class:nnn #1#2#3
118 {
119   \keys_if_exist:nnTF {#3} {#1}
120   {
121     \__keys_options_class:nn {#1} {#2}
122     \clist_put_right:Nn \l__keys_class_only_clist { \tl_to_str:n {#1} }
123   }
124   {
125     \keys_if_exist:nnTF {#3} { unknown }
126     { \__keys_options_class:nn {#1} {#2} }
127     {
128       \clist_if_in:NnF \@unusedoptionlist {#1}
129       { \clist_put_right:Nn \@unusedoptionlist {#1} }
130     }
131   }
132 }
133 \cs_new_protected:Npn \__keys_options_class:nn #1#2
134 {
135   \clist_remove_all:Nn \@unusedoptionlist {#1}
136   \clist_put_right:Nn \l__keys_options_clist {#2}
137 }

```

(End of definition for `\_keys_options_class:n`, `\_keys_options_class:nnn`, and `\_keys_options_class:nn`.)

`\_keys_options_package:n` For global options when processing a package, the tasks are slightly different from those
`\_keys_options_package:nnn` for a class. The check is the same, but here there is nothing to do if the option is not
`\_keys_options_package:nn` applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

138 \cs_new_protected:Npn \_keys_options_package:n #1
139 {
140   \clist_map_inline:Nn \@raw@classoptionslist
141   {
142     \exp_args:Ne \_keys_options_package:nnn
143     { \tl_trim_spaces:e { \_keys_remove_equals:n {##1} } }
144     {##1} {#1}
145   }
146 }
```

The forced-global test here needs to use `\tl_to_str:n` as the data come from a key name, which is always a string.

```

147 \cs_new_protected:Npn \_keys_options_package:nnn #1#2#3
148 {
149   \keys_if_exist:nnT {#3} {#1}
150   {
151     \clist_if_in:NtF \l__keys_class_only_clist { \tl_to_str:n {#1} }
152     {
153       \clist_if_in:NtF \l__keys_forced_global_clist { \tl_to_str:n {#1} }
154       { \_keys_options_package:nn {#1} {#2} }
155     }
156     { \_keys_options_package:nn {#1} {#2} }
157   }
158 }
159 \cs_new_protected:Npn \_keys_options_package:nn #1#2
160 {
161   \clist_put_right:Nn \l__keys_options_clist {#2}
162   \clist_remove_all:Nn \@unusedoptionlist {#1}
163 }
```

(End of definition for `\_keys_options_package:n`, `\_keys_options_package:nnn`, and `\_keys_options_package:nn`.)

`\_keys_options_local:` If local options are found, they are added to the processing list. L^AT_EX 2_ε stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`. For tracking the unused options, we need to hold only the option name, hence needing a mapping.

```

164 \cs_new_protected:Npn \_keys_options_local:
165 {
166   \clist_clear:N \l__keys_local_clist
167   \cs_if_eq:NNF \@current \@clsextension
168   {
169     \cs_if_exist:cT { @raw@opt@ \@currname . \@current }
170     {
171       \clist_set_eq:Nc \l__keys_local_clist
172       { @raw@opt@ \@currname . \@current }
173     }
174 }
```

```

175 \clist_map_inline:Nn \l__keys_local_clist
176 {
177     \clist_put_right:Ne \l__keys_unused_clist
178     { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
179 }
180 }

```

(End of definition for __keys_options_local:.)

__keys_options_local:n
__keys_options_local:nnn

A simpler version of the code used for package options: we stack up those that are known, unless the family supports unknown keys in which case we hover up everything.

```

181 \cs_new_protected:Npn \__keys_options_local:n #1
182 {
183     \keys_if_exist:nnTF {#1} { unknown }
184     {
185         \clist_put_right:NV \l__keys_options_clist \l__keys_local_clist
186         \clist_clear:N \l__keys_unused_clist
187     }
188     {
189         \clist_map_inline:Nn \l__keys_local_clist
190         {
191             \exp_args:Ne \__keys_options_local:nnn
192             { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
193             {##1} {#1}
194         }
195     }
196 }
197 \cs_new_protected:Npn \__keys_options_local:nnn #1#2#3
198 {
199     \keys_if_exist:nnT {#3} {#1}
200     {
201         \clist_put_right:Nn \l__keys_options_clist {#2}
202         \clist_remove_all:Nn \l__keys_unused_clist {#1}
203     }
204 }

```

(End of definition for __keys_options_local:n and __keys_options_local:nnn.)

__keys_remove_equals:n
__keys_remove_equals:w

As the name suggests, this is a simple function to remove an equals sign from the input. This is all wrapped up in an n function so that there will always be a sign available.

```

205 \cs_new:Npn \__keys_remove_equals:n #1
206 { \__keys_remove_equals:w #1 = \s__keys_stop }
207 \cs_new:Npn \__keys_remove_equals:w #1 = #2 \s__keys_stop { \exp_not:n {#1} }

```

(End of definition for __keys_remove_equals:n and __keys_remove_equals:w.)

1.4 The document interfaces

_keys_options_expand_module:Nn
_keys_options_expand_module:nN

To deal with active characters inside the module argument whilst also expanding that argument, we use a combination of c- and f-type expansion. This works as the definitions for active UTF-8 bytes contain an \ifincsname test.

```

208 \cs_new_protected:Npn \_keys_options_expand_module:Nn #1#2
209 {
210     \cs:w \_keys_options_expand_module:nN \use:e { \cs_end: {#2} } #1

```

```

211 }
212 \cs_new_protected:Npn \__keys_options_expand_module:nN #1#2
213 { #2 {#1} }

```

(End of definition for __keys_options_expand_module:Nn and __keys_options_expand_module:nN.)

\DeclareKeys Defining key options is quite straight-forward: we have an intermediate function to allow for potential set-up steps.

```

214 \NewDocumentCommand \DeclareKeys { 0 { \@currname } +m }
215 { \__keys_options_expand_module:Nn \keys_define:nn {#1} {#2} }

```

(End of definition for \DeclareKeys. This function is documented on page 1150.)

\DeclareUnknownKeyHandler

```

216 \NewDocumentCommand \DeclareUnknownKeyHandler { 0 { \@currname } +m }
217 {
218   \cs_set_protected:cpn { __keys_unknown_handler_ #1 :nn } ##1##2 {#2}
219   \__keys_options_expand_module:Nn \keys_define:ne {#1}
220   {
221     unknown .code:n =
222     \exp_not:N \exp_args:NV
223     \exp_not:c { __keys_unknown_handler_ #1 :nn }
224     \exp_not:N \l_keys_key_str {##1}
225   }
226 }

```

(End of definition for \DeclareUnknownKeyHandler. This function is documented on page 1150.)

\ProcessKeyOptions We need to deal with the older interface from l3keys2e here: it had a mandatory argument. We can mop that up using a look-ahead, and then exploit that information to determine whether the package option handling is set up for the new approach for clash handling.

```

227 \NewDocumentCommand \ProcessKeyOptions { 0 { \@currname } }
228 { \__keys_options:n {#1} }
229 \@onlypreamble \ProcessKeyOptions

```

(End of definition for \ProcessKeyOptions. This function is documented on page 1151.)

1.5 Option usage scope

__keys_options_loaded:n Indicates that the load-time options for a package have been processed: once this has happened, make them unavailable either with a warning or an error.

```

230 \cs_new_protected:Npn \__keys_options_loaded:n #1
231 {
232   \prop_get:NnNT \l_keys_usage_load_prop {#1} \l_keys_tmpa_tl
233   {
234     \clist_map_inline:Nn \l_keys_tmpa_tl
235     {
236       \keys_define:nn {#1}
237       {
238         ##1 .code:n =
239         \__keys_options_loaded:nn {#1} {##1}
240       }
241     }
242   }

```



```

243 }
244 \cs_new_protected:Npn \__keys_options_loaded:nn #1#2
245 {
246   \bool_if:NTF \l__keys_options_loading_bool
247     { \msg_warning:nnnn { keys } { load-option-ignored } }
248     { \msg_error:nnnn { keys } { load-only } }
249     {#1} {#2}
250 }

251 \msg_new:nnn { keys } { load-option-ignored }
252 {
253   Package~"\msg_module_name:n {#1}"~has~already~been~loaded:~
254   ignoring~load-time-option~"#2".
255 }

256 \msg_new:nnnn { keys } { load-only }
257 {
258   Key~"#2"~may~only~be~used~during~loading~of~package~
259   "\msg_module_name:n {#1}".
260 }
261 {
262   LaTeX~was~asked~to~set~a~key~called~"#2",~but~this~is~only~allowed~
263   in~the~optional~argument~when~loading~package~"\msg_module_name:n{#1}".
264 }

(End of definition for \__keys_options_loaded:n and \__keys_options_loaded:nn.)
Disable all preamble options in one shot.

265 \tl_gput_left:Nn \@kernel@after@begindocument
266 {
267   \prop_map_inline:Nn \l_keys_usage_preamble_prop
268   {
269     \clist_map_inline:nn {#2}
270     {
271       \keys_define:nn {#1}
272       {
273         ##1 .code:n =
274         \msg_error:nnn { keys } { preamble-only } {##1}
275       }
276     }
277   }
278 }
279 \msg_new:nnnn { keys } { preamble-only }
280 { Key~"#1"~may~only~be~used~in~the~preamble. }
281 {
282   LaTeX~was~asked~to~set~a~key~called~"#1",~but~this~is~only~allowed~
283   before~\begin{document}.~You~will~need~to~set~the~key~earlier.
284 }

```

1.6 General key setting

\SetKeys A simple wrapper.

```

285 \NewDocumentCommand \SetKeys { 0 { \@currname } +m }
286 { \__keys_options_expand_module:Nn \keys_set:nn {#1} {#2} }

```

(End of definition for \SetKeys. This function is documented on page 1151.)

287 \ExplSyntaxOff

288 </2kernel>

File 52

ltfilehook.dtx

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed). Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all “include files” or all “packages”, and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

<code>file/before</code> <code>file/.../before</code> <code>file/.../after</code> <code>file/after</code>	These are: <code>file/before</code>, <code>file/⟨file-name⟩/before</code> These hooks are executed in that order just before the file is loaded for reading. The code of the first hook is used with every file, while the second is executed only for the file with matching <code>⟨file-name⟩</code> allowing you to specify code that only applies to one file.
--	--

`file/⟨file-name⟩/after`, `file/after` These hooks are executed after the file with name `⟨file-name⟩` has been fully consumed. The order is swapped (the specific one comes first) so that the `/before` and `/after` hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of L^AT_EX is:

```
\UseHook{file/before}
\UseHook{file/⟨file name⟩/before}
  ⟨file contents⟩
\UseHook{file/⟨file name⟩/after}
\UseHook{file/after}
```

The file hooks only refer to the file by its name and extension, so the `⟨file name⟩` should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, the `.tex` extension is not assumed in hook `⟨file name⟩`, so `.tex` files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that T_EX prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the `<file name>` is available in `\CurrentFile`, which is then used when accessing the `file/<file name>/before` and `file/<file name>/after`.

<code>\CurrentFile</code>	The name of the file about to be read (or just finished) is available to the hooks through <code>\CurrentFile</code> (there is no <code>expl3</code> name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, <code>\input{sample}</code> and <code>\input{app/sample.tex}</code> would both have <code>\CurrentFile</code> being <code>sample.tex</code> .
---------------------------	--

<code>\CurrentFilePath</code>	The path to the current file (complement to <code>\CurrentFile</code>) is available in <code>\CurrentFilePath</code> if needed. The paths returned in <code>\CurrentFilePath</code> are only user paths, given through <code>\input@path</code> (or <code>expl3</code> 's equivalent <code>\l_file_search_path_seq</code>) or by directly typing in the path in the <code>\input</code> command or equivalent. Files located by <code>kpsewhich</code> get the path added internally by the T _E X implementation, so at the macro level it looks as if the file were in the current folder, so the path in <code>\CurrentFilePath</code> is empty in these cases (package and class files, mostly).
-------------------------------	--

<code>\CurrentFileUsed</code> <code>\CurrentFilePathUsed</code>	In normal circumstances these are identical to <code>\CurrentFile</code> and <code>\CurrentFilePath</code> . They will differ when a file substitution has occurred for <code>\CurrentFile</code> . In that case, <code>\CurrentFileUsed</code> and <code>\CurrentFilePathUsed</code> will hold the actual file name and path loaded by L ^A T _E X, while <code>\CurrentFile</code> and <code>\CurrentFilePath</code> will hold the names that were <i>asked for</i> . Unless doing very specific work on the file being read, <code>\CurrentFile</code> and <code>\CurrentFilePath</code> should be enough.
--	---

1.3 Hooks for package and class files

Commands to load package and class files (e.g., `\usepackage`, `\RequirePackage`, `\LoadPackageWithOptions`, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., `file/array.sty/after` would be called after the `array` package got loaded. But as packages and classes form as special group of files, there are some additional hooks available that only apply when a package or class is loaded.

<code>package/before</code> <code>package/after</code> <code>package/.../before</code> <code>package/.../after</code> <code>class/before</code> <code>class/after</code> <code>class/.../before</code> <code>class/.../after</code>	<p>These are:</p> <p><code>package/before</code>, <code>package/after</code> These hooks are called for each package being loaded.</p> <p><code>package/<name>/before</code>, <code>package/<name>/after</code> These hooks are additionally called if the package name is <code><name></code> (without extension).</p> <p><code>class/before</code>, <code>class/after</code> These hooks are called for each class being loaded.</p> <p><code>class/<name>/before</code>, <code>class/<name>/after</code> These hooks are additionally called if the class name is <code><name></code> (without extension).</p>
--	---

All `/after` hooks are implemented as reversed hooks.

The overall sequence of execution for `\usepackage` and friends is:

```
\UseHook{package/before}
\UseOneTimeHook{package/<package name>/before}

  \UseHook{file/before}
  \UseHook{file/<package name>.sty/before}
  <package contents>
  \UseHook{file/<package name>.sty/after}
  \UseHook{file/after}

code from \AtEndOfPackage if used inside the package

\UseOneTimeHook{package/<package name>/after}
\UseHook{package/after}
```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{package/varioref/after}
{ ... apply my customizations if the package gets
  loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

1.4 Hooks for `\include` files

To manage `\include` files, L^AT_EX issues a `\clearpage` before and after loading such a file. Depending on the use case one may want to execute code before or after these `\clearpages` especially for the one that is issued at the end.

Executing code before the final `\clearpage`, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.⁵³ None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if L^AT_EX makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is “No file `<filename>.tex`”).

⁵³If you want to execute code before the first `\clearpage` there is no need to use a hook—you can write it directly in front of the `\include`.

<code>include/before</code>	These are:
<code>include/.../before</code>	
<code>include/end</code>	<code>include/before, include/<name>/before</code> These hooks are executed (in that order) after the initial <code>\clearpage</code> and after <code>.aux</code> file is changed to use <code><name>.aux</code> , but
<code>include/.../end</code>	before the <code><name>.tex</code> file is loaded. In other words they are executed at the very
<code>include/after</code>	beginning of the first page of the <code>\include</code> file.
<code>include/.../after</code>	

`include/<name>/end, include/end` These hooks are executed (in that order) after $\text{\LaTeX}$ has stopped reading from the `\include` file, but before it has issued a `\clearpage` to output any deferred floats.

`include/<name>/after, include/after` These hooks are executed (in that order) after $\text{\LaTeX}$ has issued the `\clearpage` but before it has switched back writing to the main `.aux` file. Thus technically we are still inside the `\include` and if the hooks generate any further typeset material including anything that writes to the `.aux` file, then it would be considered part of the included material and bypassed if it is not loaded because of some `\includeonly` statement.⁵⁴

`include/excluded, include/<name>/excluded` The above hooks for `\include` files are only executed when the file is loaded (or more exactly the load is attempted). If, however, the `\include` file is explicitly excluded (through an `\includeonly` statement) the above hooks are bypassed and instead the `include/excluded` hook followed by the `include/<name>/excluded` hook are executed. This happens after $\text{\LaTeX}$ has loaded the `.aux` file for this include file, i.e., after $\text{\LaTeX}$ has updated its counters to pretend that the file was seen.

All `include` hooks involving the name of the included file are implemented as one-time hooks (whereas all other such hooks are normal hooks).

If you want to execute code that is run for every `\include` regardless of whether or not it is excluded, use the `cmd/include/before` or `cmd/include/after` hooks.

1.5 High-level interfaces for $\text{\LaTeX}$

We do not provide any additional wrappers around the hooks (like `filehook` or `scrfile` do) because we believe that for package writers the high-level commands from the hook management, e.g., `\AddToHook`, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

⁵⁴For that reason another `\clearpage` is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

1.6 Kernel, class, and package interfaces for L^AT_EX

<code>\declare@file@substitution</code>	<code>\declare@file@substitution {<file>} {<replacement-file>}</code>
<code>\undeclare@file@substitution</code>	<code>\undeclare@file@substitution {<file>}</code>

If `<file>` is requested for loading replace it with `<replacement-file>`. `\CurrentFile` remains pointing to `<file>` but `\CurrentFileUsed` will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of L^AT_EX kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases. As such it is mainly meant for use in the L^AT_EX kernel but other use cases are conceivable.

The `\undeclare@file@substitution` declaration undoes a substitution made earlier.

Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!

<code>\disable@package@load</code>	<code>\disable@package@load {<package>} {<alternate-code>}</code>
<code>\reenable@package@load</code>	<code>\reenable@package@load {<package>}</code>

If `<package>` is requested, do not load it but instead run `<alternate-code>` which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files and again it should only be applied if there are good reasons for doing this.⁵⁵

1.7 A sample package for structuring the log output

As an application we provide the package `structuredlog` that adds lines to the `.log` when a file is opened and closed for reading keeping track of nesting level as well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd
= (LEVEL 1 STOP) t1lmr.fd
= (LEVEL 1 START) supp-pdf.mkii
= (LEVEL 1 STOP) supp-pdf.mkii
= (LEVEL 1 START) nameref.sty
== (LEVEL 2 START) refcount.sty
== (LEVEL 2 STOP) refcount.sty
== (LEVEL 2 START) gettitlestring.sty
== (LEVEL 2 STOP) gettitlestring.sty
= (LEVEL 1 STOP) nameref.sty
= (LEVEL 1 START) ltfilehook-doc.out
```

⁵⁵Just to be sure: “I don’t like this package by somebody else” is not a good one :-)

```

= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) otllmr.fd
== (LEVEL 2 STOP) otllmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) tsllmr.fd
== (LEVEL 2 STOP) tsllmr.fd
== (LEVEL 2 START) tllmss.fd
== (LEVEL 2 STOP) tllmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx

```

Thus if you inspect an issue in the .log it is easy to figure out in which file it occurred, simply by searching back for LEVEL and if it is a STOP then remove 1 from the level value and search further for LEVEL with that value which should then be the START level of the file you are in.

2 The Implementation

```

1 <*2kernel>
2 <@@=filehook>

```

2.1 Document and package-level commands

\CurrentFile User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested \input).
\CurrentFilePath The versions \...Used hold the *actual* file name and path that is loaded by L^AT_EX, whereas the other two hold the name as requested. They will differ in case there's a file substitution.
\CurrentFileUsed
\CurrentFilePathUsed

```

3 </2kernel>
4 <*2kernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2020/10/01}%
6 <latexrelease>          {\CurrentFile}{Hook management file}%
7 \ExplSyntaxOn
8 \tl_new:N \CurrentFile
9 \tl_new:N \CurrentFilePath
10 \tl_new:N \CurrentFileUsed
11 \tl_new:N \CurrentFilePathUsed

```



```

12 \ExplSyntaxOff
13 </2ekernel | latexrelease>
14 <latexrelease>\EndIncludeInRelease

15 <latexrelease>\IncludeInRelease{0000/00/00}%
16 <latexrelease>                {\CurrentFile}{Hook management file}%
17 <latexrelease>
18 <latexrelease>\let \CurrentFile      \@undefined
19 <latexrelease>\let \CurrentFilePath  \@undefined
20 <latexrelease>\let \CurrentFileUsed  \@undefined
21 <latexrelease>\let \CurrentFilePathUsed \@undefined
22 <latexrelease>
23 <latexrelease>\EndIncludeInRelease
24 <*2ekernel>

```

(End of definition for `\CurrentFile` and others. These functions are documented on page 1161.)

2.2 expl3 helpers

```

25 </2ekernel>
26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/10/01}%
28 <latexrelease>                {\_filehook_file_parse_full_name:nN}{File helpers}%
29 \ExplSyntaxOn

```

`\_filehook_file_parse_full_name:nN`
`\_filehook_full_name:nn`

A utility macro to trigger expl3's file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn't exist, no normalization takes place. The output of `\_filehook_file_parse_full_name:nN` is passed on to the #2—a 3-argument macro that takes the `<path>`, `<base>`, and `<ext>` parts of the file name.

```

30 \cs_new:Npn \_filehook_file_parse_full_name:nN #1
31 {
32   \exp_args:Nf \file_parse_full_name_apply:nN
33   {
34     \exp_args:Nf \_filehook_full_name:nn
35     { \file_full_name:n {#1} } {#1}
36   }
37 }
38 \cs_new:Npn \_filehook_full_name:nn #1 #2
39 {
40   \tl_if_empty:nTF {#1}
41   { \tl_trim_spaces:n {#2} }
42   { \tl_trim_spaces:n {#1} }
43 }

```

(End of definition for `\_filehook_file_parse_full_name:nN` and `\_filehook_full_name:nn`.)

`\_filehook_if_no_extension:nTF`
`\_filehook_drop_extension:N`

Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use `\_filehook_file_parse_full_name:nN` to split up the file name and either check if `<ext>` (#3) is empty, or discard it.

```

44 \cs_new:Npn \_filehook_if_no_extension:nTF #1
45 {
46   \exp_args:Ne \tl_if_empty:nTF
47   { \file_parse_full_name_apply:nN {#1} \use_iii:nnn }
48 }

```

```

49 \cs_new_protected:Npn \__filehook_drop_extension:N #1
50 {
51   \tl_gset:Ne #1
52   {
53     \exp_args:NV \__filehook_file_parse_full_name:nN #1
54     \__filehook_drop_extension_aux:nnn
55   }
56 }
57 \cs_new:Npn \__filehook_drop_extension_aux:nnn #1 #2 #3
58 { \tl_if_empty:nF {#1} { #1 / } #2 }

```

(End of definition for __filehook_if_no_extension:nTF and __filehook_drop_extension:N.)

\g__filehook_input_file_seq
 \l__filehook_internal_tl
 __filehook_file_push:
 __filehook_file_pop:
 __filehook_file_pop_assign:nnnn

Yet another stack, to keep track of \CurrentFile and \CurrentFilePath with nested \inputs. At the beginning of \InputIfFileExists, the current value of \CurrentFilePath and \CurrentFile is pushed to \g__filehook_input_file_seq, and at the end, it is popped and the value reassigned. Some other places don't use \InputIfFileExists directly (\include) or need \CurrentFile earlier (\@onefilewithoptions), so these are manually used elsewhere as well.

```

59 \tl_new:N \l__filehook_internal_tl
60 \seq_if_exist:NF \g__filehook_input_file_seq
61 { \seq_new:N \g__filehook_input_file_seq }
62 \cs_new_protected:Npn \__filehook_file_push:
63 {
64   \seq_gpush:Ne \g__filehook_input_file_seq
65   {
66     { \CurrentFilePathUsed } { \CurrentFileUsed }
67     { \CurrentFilePath      } { \CurrentFile      }
68   }
69 }
70 \cs_new_protected:Npn \__filehook_file_pop:
71 {
72   \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
73   { \exp_after:wN \__filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
74   {
75     \msg_error:nnn { latex2e } { should-not-happen }
76     { Tried-to-pop-from-an-empty-file-name-stack. }
77   }
78 }
79 \cs_new_protected:Npn \__filehook_file_pop_assign:nnnn #1 #2 #3 #4
80 {
81   \tl_set:Nn \CurrentFilePathUsed {#1}
82   \tl_set:Nn \CurrentFileUsed     {#2}
83   \tl_set:Nn \CurrentFilePath     {#3}
84   \tl_set:Nn \CurrentFile         {#4}
85 }
86 \ExplSyntaxOff

```

(End of definition for \g__filehook_input_file_seq and others.)

```

87 </2ekernel|latexrelease>
88 <latexrelease>\EndIncludeInRelease

```

When rolling forward the following expl3 functions may not be defined. If we roll back the code does nothing.

```

89 <latexrelease>\IncludeInRelease{2020/10/01}%
90 <latexrelease>          {\file_parse_full_name_apply:nN}{Roll forward help}%
91 <latexrelease>
92 <latexrelease>\ExplSyntaxOn
93 <latexrelease>\cs_if_exist:NF\file_parse_full_name_apply:nN
94 <latexrelease>{
95 <latexrelease>\cs_new:Npn \file_parse_full_name_apply:nN #1
96 <latexrelease> {
97 <latexrelease>   \exp_args:Ne \__file_parse_full_name_auxi:nN
98 <latexrelease>   { \__kernel_file_name_sanitiz:n {#1} }
99 <latexrelease> }
100 <latexrelease>\cs_new:Npn \__file_parse_full_name_auxi:nN #1
101 <latexrelease> {
102 <latexrelease>   \__file_parse_full_name_area:nw { } #1
103 <latexrelease>   / \s__file_stop
104 <latexrelease> }
105 <latexrelease>\cs_new:Npn \__file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
106 <latexrelease> {
107 <latexrelease>   \tl_if_empty:nTF {#3}
108 <latexrelease>   { \__file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
109 <latexrelease>   { \__file_parse_full_name_area:nw { #1 / #2 }
110 <latexrelease>                                     #3 \s__file_stop }
111 <latexrelease> }
112 <latexrelease>\cs_new:Npn \__file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
113 <latexrelease> {
114 <latexrelease>   \tl_if_empty:nTF {#3}
115 <latexrelease>   {
116 <latexrelease>     \tl_if_empty:nTF {#1}
117 <latexrelease>     {
118 <latexrelease>       \tl_if_empty:nTF {#2}
119 <latexrelease>       { \__file_parse_full_name_tidy:nnnN { } { } }
120 <latexrelease>       { \__file_parse_full_name_tidy:nnnN { .#2 } { } }
121 <latexrelease>     }
122 <latexrelease>     { \__file_parse_full_name_tidy:nnnN {#1} { .#2 } }
123 <latexrelease>   }
124 <latexrelease>   { \__file_parse_full_name_base:nw { #1 . #2 }
125 <latexrelease>                                     #3 \s__file_stop }
126 <latexrelease> }
127 <latexrelease>\cs_new:Npn \__file_parse_full_name_tidy:nnnN #1 #2 #3 #4
128 <latexrelease> {
129 <latexrelease>   \exp_args:Nee #4
130 <latexrelease>   {
131 <latexrelease>     \str_if_eq:nnF {#3} { / } { \use_none:n }
132 <latexrelease>     #3 \prg_do_nothing:
133 <latexrelease>   }
134 <latexrelease>   { \use_none:n #1 \prg_do_nothing: }
135 <latexrelease>   {#2}
136 <latexrelease> }
137 <latexrelease>}
138 <latexrelease>\ExplSyntaxOff
139 <latexrelease>
140 <latexrelease>\EndIncludeInRelease
141 <*2ekernel>
142 <@@=>

```

2.3 Declaring the file-related hooks

These hooks have names with three-parts that start with `file/`, `include/`, `class/` or `package/` and end with `/before` or `/after` (or `/end` in the case of `include/`). They are all generic hooks so will be declared only if code is added to them; this declaration is done for you automatically and, indeed, they should not be declared explicitly.

Those named `.../after` and `include/.../end` are, when code is added, declared as reversed hooks.

2.4 Patching L^AT_EX’s `\InputIfFileExists` command

Most of what we have to do is adding `\UseHook` into several L^AT_EX 2_ε core commands, because of some circular dependencies in the kernel we do this only now and not in `ltfiles`.

`\InputIfFileExists` `\InputIfFileExists` loads any file if it is available so we have to add the hooks `file/before` and `file/after` in the right places. If the file doesn’t exist no hooks should be executed.

```

143 </2ekernel>
144 <latexrelease>\IncludeInRelease{2020/10/01}%
145 <latexrelease>          {\InputIfFileExists}{Hook management (files)}%
146 <*2ekernel|latexrelease>

147 \let\InputIfFileExists\@undefined
148 \DeclareRobustCommand \InputIfFileExists[2]{%
149   \IfFileExists{#1}%
150   {%
151     \@expl@@@filehook@file@push@@
152     \@filehook@set@CurrentFile

```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don’t change their value meanwhile. This isn’t a worry with `\CurrentFile...` because they are kept in a stack.

```

153   \expandafter\@swaptwoargs\expandafter
154   {\expandafter\@input@file@exists@with@hooks
155     \expandafter{\@filef@und}}%
156   {#2}%
157   \@expl@@@filehook@file@pop@@
158   }%
159 }
160 \def\@input@file@exists@with@hooks#1{%

```

If the file exists then `\CurrentFile` holds its name. But we can’t rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/.../after` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

`expl3`’s `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension

are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```

161 \edef\reserved@a{%
162   \@expl@@@filehook@file@pop@assign@@nnnn
163   {\CurrentFilePathUsed}%
164   {\CurrentFileUsed}%
165   {\CurrentFilePath}%
166   {\CurrentFile}}%
167 \expandafter\@swaptwoargs\expandafter{\reserved@a}%

```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```

\filename@parse{<filename>}
\ifx\filename@ext\@clsextension
...
\fi

```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\@clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a T_EX string like `\jobname`) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in T_EX Live didn't bring up any obvious candidate for breaking with this catcode change).

```

168 {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
169  \@addtofilelist{\string@makeletter\reserved@a}%
170  \UseHook{file/before}%

```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```

171  \UseHook{file/\CurrentFile/before}%
172  \@input #1% <- trailing space comes from \@filef@und
173  }%

```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@@nnnn`) so we can use it once more.

```

174  \UseHook{file/\CurrentFile/after}%
175  \UseHook{file/after}}
176 \def\unqu@tefilef@und"#1" \@nil{#1}

```

Now declare the non-generic file hooks used above:

```

177 \NewHook{file/before}
178 \NewReversedHook{file/after}
179 \<latexrelease>\EndIncludeInRelease
180 \</2ekernel|latexrelease>

```

Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute '#3'.

```

181 \<latexrelease>\IncludeInRelease{2019/10/01}%
182 \<latexrelease>      {\InputIfFileExists}{Hook management (files)}%
183 \<latexrelease>
184 \<latexrelease>\DeclareRobustCommand \InputIfFileExists[2]{%
185 \<latexrelease>  \IfFileExists{#1}%
186 \<latexrelease>    {%

```

```

187 <latexrelease> \expandafter\@swaptwoargs\expandafter
188 <latexrelease> {\@filef@und}{#2\@addtofilelist{#1}\@input}}
189 <latexrelease>\let\@input@file@exists@with@hooks\@undefined
190 <latexrelease>\let\unqu@tefilef@und\@undefined
191 <latexrelease>\EndIncludeInRelease

192 <latexrelease>\IncludeInRelease{0000/00/00}%
193 <latexrelease> {\InputIfFileExists}{Hook management (files)}%
194 <latexrelease>\long\def \InputIfFileExists#1#2{%
195 <latexrelease> \IfFileExists{#1}%
196 <latexrelease> {#2\@addtofilelist{#1}\@input \@filef@und}}

```

Also undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

197 <latexrelease>\expandafter\let\csname InputIfFileExists \endcsname\@undefined
198 <latexrelease>\let\@input@file@exists@with@hooks\@undefined
199 <latexrelease>\let\unqu@tefilef@und\@undefined
200 <latexrelease>\EndIncludeInRelease
201 <*2ekernel>

```

(End of definition for `\InputIfFileExists`, `\@input@file@exists@with@hooks`, and `\unqu@tefilef@und`.)

2.5 Declaring a file substitution

```

202 <@=filehook>
203 </2ekernel>
204 <*2ekernel | latexrelease>
205 <latexrelease>\IncludeInRelease{2020/10/01}%
206 <latexrelease> {\_filehook_subst_add:nn}{Declaring file substitution}%
207 \ExplSyntaxOn

```

`\_filehook_subst_add:nn` `\_filehook_subst_add:nn` declares a file substitution by doing a (global) definition of the form `\def\@file-subst@{file}{replacement}`. The file names are properly sanitized, and normalized with the same treatment done for the file hooks. That is, a file replacement is declared by using the file name (and extension, if any) only, and the file path should not be given. If a file name is empty it is replaced by `.tex` (the empty csname is used to check that).

```

208 \cs_new_protected:Npn \_filehook_subst_add:nn #1 #2
209 {
210   \group_begin:
211     \cs_set:cpe { } { \exp_not:o { \cs:w\cs_end: } }
212     \int_set:Nn \tex_escapechar:D { -1 }
213     \cs_gset:cpe
214     {
215       @file-subst@
216       \_filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
217     }
218     { \_filehook_subst_file_normalize:Nn \_filehook_file_name_compose:nnn
219       {#2} }
220   \group_end:
221 }
222 \cs_new_protected:Npn \_filehook_subst_remove:n #1
223 {

```

```

224 \group_begin:
225 \cs_set:cpe { } { \exp_not:o { \cs:w\cs_end: } }
226 \int_set:Nn \tex_escapechar:D { -1 }
227 \cs_undefine:c
228 {
229 @file-subst@
230 \__filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
231 }
232 \group_end:
233 }
234 \cs_new:Npn \__filehook_subst_file_normalize:Nn #1 #2
235 {
236 \exp_after:wN \__filehook_subst_empty_name_chk:NN
237 \cs:w \exp_after:wN \cs_end:
238 \cs:w \__filehook_file_parse_full_name:nN {#2} #1 \cs_end:
239 }
240 \cs_new:Npn \__filehook_subst_empty_name_chk:NN #1 #2
241 { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }

```

(End of definition for `\__filehook_subst_add:nn` and others.)

`\use_ii_iii:nnn` A variant of `\use_...` to discard the first of three arguments.

Todo: this should move to `expl3`

```

242 \cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}

```

(End of definition for `\use_ii_iii:nnn`.)

```

243 \ExplSyntaxOff
244 </2ekernel | latexrelease>
245 <latexrelease>\EndIncludeInRelease
246 <*2ekernel>

```

`\declare@file@substitution` For two internals we provide L^AT_EX 2_ε names so that we can use them elsewhere in the kernel (and so that they can be used in packages if really needed, e.g., `scrfile`).

`\undecclare@file@substitution`

```

247 </2ekernel>
248 <*2ekernel | latexrelease>
249 <latexrelease>\IncludeInRelease{2020/10/01}%
250 <latexrelease> \declare@file@substitution{File substitution}%
251 \ExplSyntaxOn
252 \cs_new_eq:NN \declare@file@substitution \__filehook_subst_add:nn
253 \cs_new_eq:NN \undecclare@file@substitution \__filehook_subst_remove:n
254 \ExplSyntaxOff
255 </2ekernel | latexrelease>
256 <latexrelease>\EndIncludeInRelease

```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```

257 <latexrelease>\IncludeInRelease{0000/00/00}%
258 <latexrelease> \declare@file@substitution{File substitution}%
259 <latexrelease>
260 <latexrelease>\let \declare@file@substitution \@gobbletwo
261 <latexrelease>\let \undecclare@file@substitution \@gobble
262 <latexrelease>

```

```
263 <latexrelease>\EndIncludeInRelease
```

```
264 <*2ekernel>
```

(End of definition for \declare@file@substitution and \undeclear@file@substitution. These functions are documented on page 1164.)

```
265 <@@=>
```

2.6 Selecting a file (\set@curr@file)

```
\set@curr@file
\set@curr@file@nosearch
  \curr@file
  \curr@file@reqd
```

Now we hook into \set@curr@file to resolve a possible file substitution, and add \expl@@@filehook@set@curr@file@@nNN at the end, after \curr@file is set.

A file name is built using \expandafter\string\csname<filename>\endcsname to avoid expanding utf8 active characters. The \csname expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when <filename> is empty, the generated control sequence is \csname\endcsname, and doing \string on that results in the file csnameendcsname.tex. To guard against that we \ifx-compare the generated control sequence with the empty csname. To do so, \csname\endcsname has to be defined, otherwise it would be equal to \relax and we would have false positives. Here we define \csname\endcsname to expand to itself to avoid it matching the definition of some other control sequence.

```
266 </2ekernel>
```

```
267 <*2ekernel | latexrelease>
```

```
268 <latexrelease>\IncludeInRelease{2022/06/01}%
```

```
269 <latexrelease>          {\set@curr@file}{Setting current file name}%
```

```
270 \def\set@curr@file{%
```

```
271   \begingroup
```

```
272     \set@curr@file@aux}
```

```
273 \edef\set@curr@file@nosearch{%
```

```
274   \begingroup
```

```
275     \let\noexpand\input@path\noexpand\@empty
```

```
276     \csname seq_clear:N\endcsname
```

```
277     \expandafter\noexpand\csname l_file_search_path_seq\endcsname
```

```
278     \noexpand\set@curr@file@aux}
```

```
279 \def\set@curr@file@aux#1{%
```

```
280   \escapechar\m@ne
```

```
281   \let\protect\string
```

```
282   \edef~{\string~}%
```

```
283   \expandafter\def\csname\expandafter\endcsname
```

```
284     \expandafter{\csname\endcsname}%
```

Two file names are set here: \curr@file@reqd which is the file requested by the user, and \curr@file which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. \curr@file is resolved first, to check if a substitution happens. If it doesn't, \expl@@@filehook@if@file@replaced@@TF short-cuts and just copies \curr@file, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default .tex is *not* added to \curr@file because for applications other than \input (graphics, for example) the default extension may not be .tex. First check if the input has an extension, then if the input had no extension, call

\@expl@@@filehook@drop@extension@@N. In case of a file substitution, \@curr@file will have an extension.

```

285 \exp1@@@filehook@if@no@extension@@nTF{#1}%
286 {\@tempwattrue}{\@tempwafalse}%
287 \@kernel@make@file@csname\@curr@file
288 \exp1@@@filehook@resolve@file@subst@@w {#1}%
289 \@exp1@@@filehook@if@file@replaced@@TF
290 {\@kernel@make@file@csname\@curr@file@reqd
291 \exp1@@@filehook@normalize@file@name@@w{#1}%
292 \if@tempswa \@exp1@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
293 {\if@tempswa \@exp1@@@filehook@drop@extension@@N\@curr@file \fi
294 \global\let\@curr@file@reqd\@curr@file}%
295 \@exp1@@@filehook@clear@replacement@flag@@
296 \endgroup}
297 /2kernel|latexrelease)
298 (latexrelease)\EndIncludeInRelease

299 (latexrelease)\IncludeInRelease{2021/06/01}%
300 (latexrelease) {\set@curr@file}{Setting current file name}%
301 (latexrelease)\def\set@curr@file#1{%
302 (latexrelease) \begingroup
303 (latexrelease) \escapechar\m@ne
304 (latexrelease) \let\protect\string
305 (latexrelease) \edef~{\string~}%
306 (latexrelease) \expandafter\def\csname\expandafter\endcsname
307 (latexrelease) \expandafter{\csname\endcsname}%
308 (latexrelease) \@exp1@@@filehook@if@no@extension@@nTF{#1}%
309 (latexrelease) {\@tempwattrue}{\@tempwafalse}%
310 (latexrelease) \@kernel@make@file@csname\@curr@file
311 (latexrelease) \exp1@@@filehook@resolve@file@subst@@w {#1}%
312 (latexrelease) \@exp1@@@filehook@if@file@replaced@@TF
313 (latexrelease) {\@kernel@make@file@csname\@curr@file@reqd
314 (latexrelease) \exp1@@@filehook@normalize@file@name@@w{#1}%
315 (latexrelease) \if@tempswa \@exp1@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
316 (latexrelease) {\if@tempswa \@exp1@@@filehook@drop@extension@@N\@curr@file \fi
317 (latexrelease) \global\let\@curr@file@reqd\@curr@file}%
318 (latexrelease) \@exp1@@@filehook@clear@replacement@flag@@
319 (latexrelease) \endgroup}
320 (latexrelease)\let\set@curr@file@nosearch\@undefined
321 (latexrelease)\EndIncludeInRelease

322 (latexrelease)\IncludeInRelease{2020/10/01}%
323 (latexrelease) {\set@curr@file}{Setting current file name}%
324 (latexrelease)\def\set@curr@file#1{%
325 (latexrelease) \begingroup
326 (latexrelease) \escapechar\m@ne
327 (latexrelease) \expandafter\def\csname\expandafter\endcsname
328 (latexrelease) \expandafter{\csname\endcsname}%
329 (latexrelease) \@exp1@@@filehook@if@no@extension@@nTF{#1}%
330 (latexrelease) {\@tempwattrue}{\@tempwafalse}%
331 (latexrelease) \@kernel@make@file@csname\@curr@file
332 (latexrelease) \exp1@@@filehook@resolve@file@subst@@w {#1}%
333 (latexrelease) \@exp1@@@filehook@if@file@replaced@@TF
334 (latexrelease) {\@kernel@make@file@csname\@curr@file@reqd
335 (latexrelease) \exp1@@@filehook@normalize@file@name@@w{#1}%

```

```

336 <latexrelease> \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
337 <latexrelease> {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
338 <latexrelease> \global\let\@curr@file@reqd\@curr@file}%
339 <latexrelease> \@expl@@@filehook@clear@replacement@flag@@
340 <latexrelease> \endgroup}
341 <latexrelease>\let\set@curr@file@nosearch\@undefined
342 <latexrelease>\EndIncludeInRelease

343 <latexrelease>\IncludeInRelease{2019/10/01}%
344 <latexrelease> {\set@curr@file}{Setting current file name}%
345 <latexrelease>\def\set@curr@file#1{%
346 <latexrelease> \begingroup
347 <latexrelease> \escapechar\m@ne
348 <latexrelease> \xdef\@curr@file{%
349 <latexrelease> \expandafter\expandafter\expandafter\unquote@name
350 <latexrelease> \expandafter\expandafter\expandafter{%
351 <latexrelease> \expandafter\string
352 <latexrelease> \csname\@firstofone#1\@empty\endcsname}}%
353 <latexrelease> \endgroup
354 <latexrelease> }
355 <latexrelease>\let\set@curr@file@nosearch\@undefined
356 <latexrelease>\EndIncludeInRelease

357 <latexrelease>\IncludeInRelease{0000/00/00}%
358 <latexrelease> {\set@curr@file}{Setting current file name}%
359 <latexrelease>\let\set@curr@file\@undefined
360 <latexrelease>\let\set@curr@file@nosearch\@undefined
361 <latexrelease>\EndIncludeInRelease
362 <*2ekernel>

```

(End of definition for \set@curr@file and others.)

```

\@filehook@set@CurrentFile
\@kernel@make@file@csname
\@set@curr@file@aux

```

Todo: This should get internalized using @expl@ names

```

363 </2ekernel>
364 <*2ekernel | latexrelease>
365 <latexrelease>\IncludeInRelease{2020/10/01}%
366 <latexrelease> {\@kernel@make@file@csname}{Make file csname}%
367 \def\@kernel@make@file@csname#1#2#3{%
368 \xdef#1{\expandafter\@set@curr@file@aux
369 \csname\expandafter#2\@firstofone#3\@nil\endcsname}}

```

This auxiliary compares \<filename> with \csname\endcsname to check if the empty .tex file was requested.

```

370 \long\def\@set@curr@file@aux#1{%
371 \expandafter\ifx\csname\endcsname#1%
372 .tex\else\string#1\fi}

```

Then we call \@expl@@@filehook@set@curr@file@@nNN once for \@curr@file to set \CurrentFile(Path)Used and once for \@curr@file@reqd to set \CurrentFile(Path). Here too the slower route is only used if a substitution happened, but here \@expl@@@filehook@if@file@replaced@@TF can't be used because the flag is reset at the \endgroup above, so we check if \@curr@file and \@curr@file@reqd differ. This macro is issued separate from \set@curr@file because it changes \CurrentFile, and side-effects would quickly get out of control.

```

373 \def\@filehook@set@CurrentFile{%

```

```

374 \@@expl@@@filehook@set@curr@file@@nNN{\@curr@file}%
375 \CurrentFileUsed\CurrentFilePathUsed
376 \ifx\@curr@file@reqd\@curr@file
377 \let\CurrentFile\CurrentFileUsed
378 \let\CurrentFilePath\CurrentFilePathUsed
379 \else
380 \@@expl@@@filehook@set@curr@file@@nNN{\@curr@file@reqd}%
381 \CurrentFile\CurrentFilePath
382 \fi}
383 </2ekernel | latexrelease>
384 <latexrelease>\EndIncludeInRelease
385 <*2ekernel>

(End of definition for \@filehook@set@CurrentFile, \@kernel@make@file@csname, and
\@set@curr@file@aux.)

386 <@@=filehook>

```

`\_filehook_set_curr_file:nNN`
`\_filehook_set_curr_file_assign:nnnNN`

When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (`<base>` plus `<ext>`) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```

387 </2ekernel>
388 <*2ekernel | latexrelease>
389 <latexrelease>\IncludeInRelease{2020/10/01}%
390 <latexrelease> \{\_filehook_set_curr_file:nNN\}{Set curr file}%
391 \ExplSyntaxOn
392 \cs_new_protected:Npn \_filehook_set_curr_file:nNN #1
393 {
394 \exp_args:Nf \_filehook_file_parse_full_name:nN {#1}
395 \_filehook_set_curr_file_assign:nnnNN
396 }
397 \cs_new_protected:Npn \_filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
398 {
399 \str_set:Nn #5 {#1}
400 \str_set:Nn #4 {#2#3}
401 }
402 \ExplSyntaxOff
403 </2ekernel | latexrelease>
404 <latexrelease>\EndIncludeInRelease
405 <*2ekernel>

(End of definition for \_filehook_set_curr_file:nNN and
\_filehook_set_curr_file_assign:nnnNN.)

```

2.7 Replacing a file and detecting loops

`\_filehook_resolve_file_subst:w`
`\_filehook_normalize_file_name:w`
`\_filehook_file_name_compose:nnn`

Start by sanitizing the file with `\_filehook_file_parse_full_name:nN` then do `\_filehook_file_subst_begin:nnn{\<path>}{\<name>}{\<ext>}`.

```

406 </2ekernel>
407 <*2ekernel | latexrelease>
408 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

409 <latexrelease>      {\_filehook_resolve_file_subst:w}{Replace files detect loops}%
410 \ExplSyntaxOn
411 \cs_new:Npn \_filehook_resolve_file_subst:w #1 \@nil
412   { \_filehook_file_parse_full_name:nN {#1} \_filehook_file_subst_begin:nnn }
413 \cs_new:Npn \_filehook_normalize_file_name:w #1 \@nil
414   { \_filehook_file_parse_full_name:nN {#1} \_filehook_file_name_compose:nnn }
415 \cs_new:Npn \_filehook_file_name_compose:nnn #1 #2 #3
416   { \tl_if_empty:nF {#1} { #1 / } #2#3 }

```

```

\_filehook_file_replaced
\_filehook_if_file_replaced:TF
\_filehook_clear_replacement_flag:

```

Since the file replacement is done expandably in a `\csname`, use a flag to remember if a substitution happened. We use this in `\setcurr@file` to short-circuit some of it in case no substitution happened (by far the most common case, so it's worth optimizing). The flag raised during the file substitution algorithm must be explicitly cleared after the `\_filehook_if_file_replaced:TF` conditional is no longer needed, otherwise further uses of `\_filehook_if_file_replaced:TF` will wrongly return true.

```

417 \flag_new:n { \_filehook_file_replaced }
418 \cs_new:Npn \_filehook_if_file_replaced:TF #1 #2
419   { \flag_if_raised:nTF { \_filehook_file_replaced } {#1} {#2} }
420 \cs_new_protected:Npn \_filehook_clear_replacement_flag:
421   { \flag_clear:n { \_filehook_file_replaced } }

```

```

\_filehook_file_subst_begin:nnn

```

First off, start by checking if the current file (`\langle name \rangle + \langle ext \rangle`) has a declared substitution. If not, then just put that as the name (including a possible `\langle path \rangle` in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary `\_filehook_file_subst_tortoise_hare:nn` sees that there's no replacement for `#2#3` and does nothing else.

```

422 \cs_new:Npn \_filehook_file_subst_begin:nnn #1 #2 #3
423   {
424     \_filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
425     { \_filehook_file_name_compose:nnn {#1} {#2} {#3} }
426   }
427 \ExplSyntaxOff
428 </2ekernel|latexrelease>
429 <latexrelease>\EndIncludeInRelease
430 <*2ekernel>

```

2.7.1 The Tortoise and Hare algorithm

```

\_filehook_file_subst_tortoise_hare:nn
\_filehook_file_subst_loop:NN
\_filehook_file_subst_loop:cc

```

If there is a substitution (`\langle true \rangle` in the first `\cs_if_exist:cTF` below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the `\flag \_filehook_file_replaced` is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put $\text{\TeX}$ in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, `\_filehook_file_subst_loop:cc` is called with `\@file-subst@<file>` and

`\@file-subst@\@file-subst@<file>`; that is, the substitution of `<file>` and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within `\__filehook_file_subst_loop:NN` the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, `\__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go `<false>` and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```

431 </2ekernel>
432 <*2ekernel|latexrelease>
433 <latexrelease>\IncludeInRelease{2020/10/01}%
434 <latexrelease> {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
435 \ExplSyntaxOn
436 \cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
437 {
438   \cs_if_exist:cTF { @file-subst@ #2 }
439   {
440     \flag_if_raised:nF { __filehook_file_replaced }
441     { \flag_raise:n { __filehook_file_replaced } }
442     \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
443     {
444       \__filehook_file_subst_loop:cc
445       { @file-subst@ #1 }
446       { @file-subst@ \use:c { @file-subst@ #2 } }
447     }
448     { \use:c { @file-subst@ #2 } }
449   }
450   { #3 }
451 }

```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `\__filehook_file_subst_cycle_error:cN` is called to report the error.

```

452 \cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
453 {
454   \token_if_eq_meaning:NNTF #1 #2
455   {
456     .tex
457     \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
458   }
459   { \__filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
460 }
461 \cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }

```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```

462 \cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
463 {
464   \msg_expandable_error:nnff { latex2e } { file-cycle }

```

```

465     {#1} { \use:c { @file-subst@ #1 } }
466     \token_if_eq_meaning:NNF #1 #2
467     { \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
468   }
469   \cs_generate_variant:Nn \__filehook_file_subst_cycle_error:NN { c }

```

And the error message:

```

470   \msg_new:nnn { latex2e } { file-cycle }
471   { File~loop!~#1~replaced~by~#2... }

```

(End of definition for `\__filehook_resolve_file_subst:w` and others.)

```

472   \ExplSyntaxOff
473   </2ekernel | latexrelease>
474   <latexrelease>\EndIncludeInRelease
475   <*2ekernel>
476   <@@=>

```

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

`\disable@package@load` defines `\@pkg-disable@<package>` to expand to some code #2 instead of loading the package.

```

\reenable@package@load
\@disable@packageload@do
477 </2ekernel>
478 <*2ekernel | latexrelease>
479 <latexrelease>\IncludeInRelease{2020/10/01}%
480 <latexrelease>          {\disable@package@load}{Disable packages}%
481 \def\disable@package@load#1#2{%
482   \global\@namedef{\@pkg-disable@#1.\@pkgextension}{#2}}

```

Here we check if a control sequence named `\@pkg-disable@<name>.sty` is defined, and if so don't use the package loading code #2, but use the replacement code stored in that control sequence, write something to the log, and then prevent `\@onefilewithoptions` from sanity-checking the requested package date (the `\expandafter` here triggers one in `\@onefilewithoptions` that ends a conditional there, and the `\@gobbletwo` removes the date checking code from the input stream).

```

483 \def\@disable@packageload@do#1#2{%
484   \ifundefined{\@pkg-disable@#1}%
485     {#2}%
486     {\@nameuse{\@pkg-disable@#1}%
487       \@latex@info{Package '1' has been disabled.%
488         \MessageBreak Load request ignored}%
489       \expandafter\@gobbletwo}}

```

`\reenable@package@load` undefines `\@pkg-disable@<package>` to realow loading a package.

```

490 \def\reenable@package@load#1{%
491   \global\expandafter\let
492   \csname \@pkg-disable@#1.\@pkgextension \endcsname \@undefined}

```

```

493 </2ekernel | latexrelease>
494 <latexrelease>\EndIncludeInRelease
495 <latexrelease>\IncludeInRelease{0000/00/00}%
496 <latexrelease>      {\disable@package@load}{Disable packages}%
497 <latexrelease>
498 <latexrelease>\let\disable@package@load \undefined
499 <latexrelease>\let\@disable@packageload@do\undefined
500 <latexrelease>\let\reenable@package@load \undefined
501 <latexrelease>\EndIncludeInRelease
502 <*2ekernel>

```

(End of definition for \disable@package@load, \reenable@package@load, and
\@disable@packageload@do. These functions are documented on page 1164.)

2.9 High-level interfaces for L^AT_EX

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX 2_ε names to allow for internal commands to be used outside this module (and in parts that still use L^AT_EX 2_ε syntax. We have to unset the @@ since we want double “at” sign in place of double underscores.

```

503 <@@= >
504 </2ekernel>
505 <*2ekernel | latexrelease>
506 <latexrelease>\IncludeInRelease{2020/10/01}%
507 <latexrelease>      {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
508 \ExplSyntaxOn
509 \cs_new_eq:NN \@expl@@@filehook@if@no@extension@@nTF
510               \__filehook_if_no_extension:nTF
511 \cs_new_eq:NN \@expl@@@filehook@set@curr@file@@nNN
512               \__filehook_set_curr_file:nNN
513 \cs_new_eq:NN \@expl@@@filehook@resolve@file@subst@@w
514               \__filehook_resolve_file_subst:w
515 \cs_new_eq:NN \@expl@@@filehook@normalize@file@name@@w
516               \__filehook_normalize_file_name:w
517 \cs_new_eq:NN \@expl@@@filehook@if@file@replaced@@TF
518               \__filehook_if_file_replaced:TF
519 \cs_new_eq:NN \@expl@@@filehook@clear@replacement@flag@@
520               \__filehook_clear_replacement_flag:
521 \cs_new_eq:NN \@expl@@@filehook@drop@extension@@N
522               \__filehook_drop_extension:N
523 \cs_new_eq:NN \@expl@@@filehook@file@push@@
524               \__filehook_file_push:
525 \cs_new_eq:NN \@expl@@@filehook@file@pop@@
526               \__filehook_file_pop:

```

```

527 \cs_new_eq:NN \@expl@@@filehook@file@pop@assign@@nnnn
528         \__filehook_file_pop_assign:nnnn
529 \ExplSyntaxOff

```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when `latexrelease` is loaded. It cannot be `\let` to `\undefined` otherwise it would error as well, so it is `\let` to `\relax` to be silently ignored when loading `\latexrelease`.

```

530 </2ekernel | latexrelease>
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>
533 <latexrelease>\IncludeInRelease{0000/00/00}%
534 <latexrelease>    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
535 <latexrelease>\let\@expl@@@filehook@file@pop@@\relax
536 <latexrelease>\EndIncludeInRelease
537 <*2ekernel>

```

This ends the kernel code in this file.

```

538 </2ekernel>

```

3 A sample package for structuring the log output

```

539 <*structuredlog>
540 <@@=filehook>

541 \ProvidesExplPackage
542     {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
543     {Structuring the TeX transcript file}

```

`\g_filehook_nesting_level_int` Stores the current package nesting level.

```

544 \int_new:N \g__filehook_nesting_level_int

```

Initialize the counter with the number of files in the `\@currnamestack` (the number of items divided by the number of items we place on the stack for each file (which right now is 4)) minus one, because this package is skipped when printing to the log.

```

545 \int_gset:Nn \g__filehook_nesting_level_int
546     { ( \tl_count:N \@currnamestack ) / 4 - 1 }

```

(End of definition for `\g__filehook_nesting_level_int`.)

`\__filehook_log_file_record:n` This macro is responsible for increasing and decreasing the file nesting level, as well as printing to the log. The argument is either `STOPTART` or `STOP` and the action it takes on the nesting integer depends on that.

```

547 \cs_new_protected:Npn \__filehook_log_file_record:n #1
548     {
549         \str_if_eq:nnT {#1} {START} { \int_gincr:N \g__filehook_nesting_level_int }
550         \iow_term:e
551         {
552             \prg_replicate:nn { \g__filehook_nesting_level_int } { = } ~
553             ( LEVEL ~ \int_use:N \g__filehook_nesting_level_int \c_space_tl #1 ) ~
554             \CurrentFileUsed

```


If there was a file replacement, show that as well:

```

555     \str_if_eq:NNF \CurrentFileUsed \CurrentFile
556     { ~ ( \CurrentFile \c_space_tl requested ) }
557     \iow_newline:
558   }
559   \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g__filehook_nesting_level_int }
560 }

```

Now just hook the macro above in the generic file/before...

```

561 \AddToHook{file/before}{ \__filehook_log_file_record:n { START } }
...and file/after hooks. We don't want to install the file/after hook immediately,
because that would mean it is the first time executed when the package finishes. We
therefore put the declaration inside \AddToHookNext so that it gets only installed when
we have left this package.
562 \AddToHookNext{file/after}
563 { \AddToHook{file/after}{ \__filehook_log_file_record:n { STOP } } }
(End of definition for \__filehook_log_file_record:n.)
564 <@@=
565 </structuredlog>

```

4 Package emulations

4.1 Package atveryend emulation

With the new hook management and the hooks in \enddocument all of atveryend is taken care of. We can make an emulation only here after the substitution functionality is available:

```

566 <*2ekernel>
567 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
568 </2ekernel>

```

Here is the package file we point to:

```

569 <*atveryend-ltx>
570 \ProvidesPackage{atveryend-ltx}
571 [2020/08/19 v1.0a
572 Emulation of the original atveryend package^^Jwith kernel methods]

```

Here are new definitions for its interfaces now pointing to the hooks in \enddocument

```

573 \newcommand\AfterLastShipout {\AddToHook{enddocument/afterlastpage}}
574 \newcommand\AtVeryEndDocument {\AddToHook{enddocument/afteraux}}

```

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in enddocument/info.

```

575 \newcommand\AtEndAfterFileList{\AddToHook{enddocument/info}}
576 \newcommand\AtVeryVeryEnd {\AddToHook{enddocument/end}}

```

\BeforeClearDocument This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```

577 \ExplSyntaxOn
578 \newcommand\BeforeClearDocument[1]
579 { \AtEndDocument{#1}
580   \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
581 }

```

```

582 \cs_new:Npn\atveryend@DEPRECATED #1
583     {\iow_term:e{=====~DEPRECATED~USAGE~#1~=====}}
584 \ExplSyntaxOff

(End of definition for \BeforeClearDocument.)

585 \atveryend-ltx)

```

File 53

ltshipout.dtx

1 Introduction

The code provides an interface to the `\shipout` primitive of $\text{\TeX}$ which is called when a finished pages is finally “shipped out” to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages `atbegshi` and `atenddvi` even though the interfaces are somewhat different.⁵⁶

1.1 Overloading the `\shipout` primitive

`\shipout` With this implementation $\text{\TeX}$ ’s shipout primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each shipout that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a $\text{\LaTeX}$ counter.

`\RawShipout` This command implements a simplified shipout that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total shipout counters are incremented.

The command doesn’t use `\ShipoutBox` but its own private box register so that it can be used inside of shipout hooks to do some additional shipouts while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

⁵⁶Heiko’s interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

<code>\ShipoutBox</code> <code>\l_shipout_box</code>	This box register is called <code>\ShipoutBox</code> (alternatively available via the L3 name <code>\l_shipout_box</code>).
---	--

This box is a “local” box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

<code>\l_shipout_box_ht_dim</code> <code>\l_shipout_box_dp_dim</code> <code>\l_shipout_box_wd_dim</code> <code>\l_shipout_box_ht_plus_dp_dim</code>	
--	--

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no L^AT_EX 2_ε names).⁵⁷ These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

1.2 Provided hooks

<code>shipout/before</code> <code>shipout/after</code> <code>shipout/foreground</code> <code>shipout/background</code> <code>shipout/firstpage</code> <code>shipout/lastpage</code>	The code for <code>\shipout</code> offers a number of hooks into which packages (or the user) can add code to support different use cases. These are: <code>shipout/before</code> This hook is executed after the finished page has been stored in <code>\ShipoutBox</code> / <code>\l_shipout_box</code> . It can be used to alter that box content or to discard it completely (see <code>\DiscardShipoutBox</code> below).
--	---

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn’t yet include the background and foreground material).

Note: It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

`shipout/background` This hook adds a picture environment into the background of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of 1pt. It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

⁵⁷Might need changing, but HO’s version as strings is not really helpful I think).

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

shipout/foreground This hook adds a picture environment into the foreground of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of 1pt. Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its (0,0) point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

shipout This hook is executed after foreground and/or background material has been added, i.e., just in front of the actual shipout operation. Its purpose is to allow manipulation of the finalized box (stored in `\ShipoutBox`) with the extra material also in place (which is not yet the case in `shipout/before`).

It cannot be used to cancel the shipout operation via `\DiscardShipoutBox` (that has to happen in `shipout/before`, if desired!

shipout/firstpage The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the .dvi or .pdf output.⁵⁸

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

shipout/lastpage The corresponding hook to add `\specials` at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that $\text{\LaTeX}$ believes is the last one. Again it is executed regardless of the shipout method.

It may not be possible for $\text{\LaTeX}$ to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then $\text{\LaTeX}$ will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

shipout/after This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

Note: Just like `shipout/before` this hook is not meant to be used for adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout` and `shipout/after`) are added inside `hboxes` to the box being shipped out in the following order:

⁵⁸In $\text{\LaTeX 2}_\epsilon$ that was already existing, but implemented using a box register with the name `\@begindivibox`.

<code>shipout/firstpage</code>	only on the first page
<code>shipout/background</code>	
<code><boxed content of \ShipoutBox></code>	
<code>shipout/foreground</code>	
<code>shipout/lastpage</code>	only on the last page

If any of the hooks has no code then the corresponding box is added at that point.

Once the (page) box has got the above extra content it can again be manipulated using the `shipout` hook and then is shipped out for real.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes `\typeouts`, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

1.3 Legacy L^AT_EX commands

<hr/> <code>\AtBeginDvi</code>	<code>\AtBeginDvi {<code>}</code>
<code>\AtEndDvi</code>	<code>\AtBeginDvi</code> is the existing L ^A T _E X 2 _ε interface to fill the <code>shipout/firstpage</code> hook. This is not really a good name as it is not just supporting <code>.dvi</code> but also <code>.pdf</code> output or <code>.xdv</code> . <code>\AtEndDvi</code> is the counterpart that was not available in the kernel but only through the package <code>atenddvi</code> . It fills the <code>shipout/lastpage</code> hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of “randomly-named” wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other “new” wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

1.4 Special commands for use inside the hooks

<code>\DiscardShipoutBox</code> <code>\shipout_discard:</code>	<code>\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}</code>
---	---

The `\DiscardShipoutBox` declaration (L3 name `\shipout_discard:`) requests that on the next shipout the page box is thrown away instead of being shipped to the `.dvi` or `.pdf` file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that L^AT_EX output routine is called in an asynchronous manner! Thus normally one would use this only as part of the `shipout/before` code.

Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it — not done (yet)

`\DiscardShipoutBox` cannot be used in any of the `shipout/...` hooks other than `shipout/before`.

In the `atbegshi` package there are a number of additional commands for use inside the `shipout/before` hook. They should normally not be needed any more as one can instead simply add code to the hooks `shipout/before`, `shipout`, `shipout/background` or `shipout/foreground`.⁵⁹ If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

1.5 Provided LuaT_EX callbacks

<code>pre_shipout_filter</code>	Under LuaT _E X the <code>pre_shipout_filter</code> Lua callback is provided which gets called directly after the <code>shipout</code> hook, immediately before the shipout primitive gets invoked. The signature is
---------------------------------	--

```
function(<node> head)
  return true
end
```

The `head` is the list node corresponding to the box to be shipped out. The return value should always be `true`.

⁵⁹If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.6 Information counters

<code>\ReadonlyShipoutCounter</code>	<code>\ifnum\ReadonlyShipoutCounter=...</code>
<code>\g_shipout_readonly_int</code>	<code>\int_use:N \g_shipout_readonly_int % expl3 usage</code>

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a $\text{\LaTeX}$ counter but as a $\text{\TeX}$ counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

<code>totalpages</code>	<code>\arabic{totalpages}</code>
<code>\g_shipout_totalpages_int</code>	<code>\int_use:N \g_shipout_totalpage_int % expl3 usage</code>

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a $\text{\LaTeX}$ counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by $\text{\LaTeX}$. It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by $\text{\LaTeX}$ but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

<code>\PreviousTotalPages</code>	<code>\PreviousTotalPages</code>
----------------------------------	----------------------------------

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to 0. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

1.7 Debugging shipout code

<code>\DebugShipoutsOn</code>	<code>\DebugShipoutsOn</code>
<code>\DebugShipoutsOff</code>	
<code>\shipout_debug_on:</code>	Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.
<code>\shipout_debug_off:</code>	

Todo: This needs some rationalizing and may not stay this way.

2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the L^AT_EX kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

2.1 Emulating atbegshi

<code>\AtBeginShipoutUpperLeft</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutUpperLeft{<code>}...}</code>
<code>\AtBeginShipoutUpperLeftForeground</code>	

This adds a `picture` environment into the background of the shipout box expecting `<code>` to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

`\AddToHook{shipout/background}{<code>}`

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all `<code>` going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

<code>\AtBeginShipoutAddToBox</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutAddToBox{<code>}...}</code>
<code>\AtBeginShipoutAddToBoxForeground</code>	

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that `<code>` is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap `<code>` into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

<code>\AtBeginShipoutBox</code>	This is the name of the shipout box as <code>atbegshi</code> knows it.
---------------------------------	--

<code>\AtBeginShipoutOriginalShipout</code>

This is the name of the `\shipout` primitive as `atbegshi` knows it. This bypasses all the mechanisms set up by the L^AT_EX kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy `atbegshi` code but not in newly developed applications.

The kernel alternative is `\RawShipout` which is integrated with the L^AT_EX mechanisms and updates, for example, the `\ReadonlyShipoutCounter` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

<hr/> <hr/>	<code>\AtBeginShipoutInit</code>	By default <code>atbegshi</code> delayed its action until <code>\begin{document}</code> . This command was forcing it in an earlier place. With the new concept it does nothing.
<hr/> <hr/>	<code>\AtBeginShipout</code> <code>\AtBeginShipoutNext</code>	<code>\AtBeginShipout{<code>} ≡ \AddToHook{shipout/before}{<code>}</code> <code>\AtBeginShipoutNext{<code>} ≡ \AddToHookNext{shipout/before}{<code>}</code> This is equivalent to filling the <code>shipout/before</code> hook by either using <code>\AddToHook</code> or <code>\AddToHookNext</code> , respectively.
<hr/> <hr/>	<code>\AtBeginShipoutFirst</code> <code>\AtBeginShipoutDiscard</code>	The <code>atbegshi</code> names for <code>\AtBeginDvi</code> and <code>\DiscardShipoutBox</code> .

2.2 Emulating everyshi

The `everyshi` package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as `\ShipoutBox` for inspection of change, one should not manipulate box 255 directly inside `shipout/before`, so old code doing this would change to use `\ShipoutBox` instead of 255 or `\@cclv`.

<hr/> <hr/>	<code>\EveryShipout</code>	<code>\EveryShipout{<code>} ≡ \AddToHook{shipout/before}{<code>}</code>
<hr/> <hr/>	<code>\AtNextShipout</code>	<code>\AtNextShipout{<code>} ≡ \AddToHookNext{shipout/before}{<code>}</code>

However, most use cases for `everyshi` are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the `shipout/background` and `shipout/foreground` hooks without any need to coding.

2.3 Emulating atenddvi

The `atenddvi` package implemented only a single command: `\AtEndDvi` and that is now available out of the box so the emulation makes the package a no-op.

2.4 Emulating everypage

This package patched the original `\@begindvi` hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

<hr/> <hr/>	<code>\AddEverypageHook</code>	<code>\AddEverypageHook{<code>} ≡</code> <code>\AddToHook{shipout/background}{\put(1in,-1in){<code>}}</code> <code>\AddEverypageHook</code> is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the <code>\put</code> statement above.
<hr/> <hr/>	<code>\AddThispageHook</code>	<code>\AddThispageHook{<code>} ≡</code> <code>\AddToHookNext{shipout/background}{\put(1in,-1in){<code>}}</code> The <code>\AddThispageHook</code> wrapper is similar but uses <code>\AddToHookNext</code> .

3 The Implementation

```
1 <@@=shipout>
```

At the moment most the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

```
2 <*2ekernel|latexrelease>
```

```
3 <latexrelease>\IncludeInRelease{2026/11/01}%
```

```
4 <latexrelease>          {\_shipout_picture_overlay:n}{Link handling in overlay pictures}
```

```
5 \ExplSyntaxOn
```

`\_shipout_picture_overlay:n` Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.

We want to avoid that links in the main galley which break over a page spills over into the picture (gh/2210). The needed code is different for `lualatex` and other engines and also must be inserted in different places: in `vmode` for `lualatex` and in `hmode` else. The code only works if the PDF management is loaded.

```
6 \cs_new:Npn \_shipout_picture_overlay_linkon: {}
```

```
7 \cs_new:Npn \_shipout_picture_overlay_linkoff: {}
```

```
8 \cs_new:Npn \_shipout_picture_overlay_linkstate: {}
```

We need to wait until we know if the PDF management has been loaded.

```
9 \AddToHookNext{class/before}
```

```
10 {
```

```
11   \IfPDFManagementActiveT
```

```
12   {
```

```
13     \sys_if_engine luatex:TF
```

```
14     {
```

```
15       \cs_set:Npn \_shipout_picture_overlay_linkstate: {\pdfextension linkstate-3}
```

```
16     }
```

```
17     {
```

```
18       \cs_set_protected:Npn \_shipout_picture_overlay_linkon: {\pdfannot_link_on:}
```

```
19       \cs_set_protected:Npn \_shipout_picture_overlay_linkoff: {\pdfannot_link_off:}
```

```
20     }
```

```
21   }
```

```
22 }
```

```
23 \cs_new:Npn \_shipout_picture_overlay:n #1 {
```

The very first time this is executed we have to initialize (and freeze) the origins.

```
24   \_shipout_init_page_origins:
```

```
25   \kern -\c__shipout_horigin_tl \scan_stop:
```

```
26   \vbox_to_zero:n {
```

```
27     \kern -\c__shipout_vorigin_tl \scan_stop:
```

```
28     \unitlength 1pt \scan_stop:
```

```
29     \_shipout_picture_overlay_linkstate:
```

This mimics a simple zero-sized picture environment. The `\hss` is needed in case there is horizontal material (without using `\put` with a positive width).

```
30   \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
```

```
31   {
```

```
32     \_shipout_picture_overlay_linkoff:
```

```

33         \ignorespaces #1 \hss
34         \_shipout_picture_overlay_linkon:
35     }
36     \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
37     \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
38     \box_use:N \l__shipout_tmp_box
39     \tex_vss:D
40 }
41 }

(End of definition for \_shipout_picture_overlay:n.)

42 \ExplSyntaxOff
43 <latexrelease>
44 <latexrelease>\EndIncludeInRelease

45 <latexrelease>\IncludeInRelease{2020/10/01}%
46 <latexrelease>         {\_shipout_picture_overlay:n}{Link handling in overlay pictures}
47 <latexrelease>\ExplSyntaxOn
48 <latexrelease>\cs_new:Npn \_shipout_picture_overlay:n #1 {
49 <latexrelease>     \_shipout_init_page_origins:
50 <latexrelease>     \kern -\c__shipout_horigin_tl \scan_stop:
51 <latexrelease>     \vbox_to_zero:n {
52 <latexrelease>         \kern -\c__shipout_vorigin_tl \scan_stop:
53 <latexrelease>         \unitlength 1pt \scan_stop:
54 <latexrelease>         \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
55 <latexrelease>         { \ignorespaces #1 \hss }
56 <latexrelease>         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
57 <latexrelease>         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
58 <latexrelease>         \box_use:N \l__shipout_tmp_box
59 <latexrelease>         \tex_vss:D
60 <latexrelease>     }
61 <latexrelease>}
62 <latexrelease>\ExplSyntaxOff
63 <latexrelease>\EndIncludeInRelease

64 <latexrelease>\IncludeInRelease{2020/10/01}%
65 <latexrelease>         {\shipout}{Hook management (shipout)}%
66 \ExplSyntaxOn

```

3.1 Debugging

`\g__shipout_debug_bool` Holds the current debugging state.

```

67 \bool_new:N \g__shipout_debug_bool

(End of definition for \g__shipout_debug_bool.)

\shipout_debug_on: Turns debugging on and off by redefining \_shipout_debug:n.
\shipout_debug_off:
\_shipout_debug:n
\_shipout_debug_gset:
68 \cs_new_eq:NN \_shipout_debug:n \use_none:n
69 \cs_new_protected:Npn \shipout_debug_on:
70 {
71     \bool_gset_true:N \g__shipout_debug_bool
72     \_shipout_debug_gset:
73 }
74 \cs_new_protected:Npn \shipout_debug_off:

```

```

75 {
76   \bool_gset_false:N \g__shipout_debug_bool
77   \__shipout_debug_gset:
78 }
79 \cs_new_protected:Npn \__shipout_debug_gset:
80 {
81   \cs_gset_protected:Npe \__shipout_debug:n ##1
82   { \bool_if:NT \g__shipout_debug_bool {##1} }
83 }

```

(End of definition for `\shipout_debug_on:` and others. These functions are documented on page 1189.)

`\ShipoutBox` The box filled with the page to be shipped out (both L3 and L^AT_EX 2_ε name).
`\l_shipout_box`

```

84 \box_new:N \l_shipout_box
85 \cs_set_eq:NN \ShipoutBox \l_shipout_box

```

(End of definition for `\ShipoutBox` and `\l_shipout_box`. These functions are documented on page 1185.)

`\l__shipout_raw_box` The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```

86 \box_new:N \l__shipout_raw_box

```

(End of definition for `\l__shipout_raw_box`.)

`\__shipout_finalize_box:` For Lua_TE_X invoke the `pre_shipout_filter` callback.

```

87 \sys_if_engine luatex:TF
88 {
89   \newprotectedluacmd \__shipout_finalize_box:
90   \exp_args:Ne \everyjob {
91     \exp_not:V \everyjob
92     \exp_not:N \lua_now:n {
93       luatexbase.create_callback('pre_shipout_filter', 'list')
94       local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
95       lua.get_functions_table()[\the \allocationnumber] = function()
96         local~head = getbox(\the \l_shipout_box)
97         local~result = call('pre_shipout_filter', head)
98         if~not (result == head) then~
99           setbox(\the \l_shipout_box, result~or~nil)
100         end~
101       end
102     }
103   }
104 } {
105   \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
106 }

```

(End of definition for `\__shipout_finalize_box:.`)

`\__shipout_execute:` This is going to be the code run by `\shipout`. The code follows closely the ideas from `atbegshi`, so not documenting that here for now.

```

107 \cs_set_protected:Npn \__shipout_execute: {
108   \tl_set:Nc \l__shipout_group_level_tl
109   { \int_value:w \tex_currentgrouplevel:D }

```

```

110 \tex_afterassignment:D \__shipout_execute_test_level:
111 \tex_setbox:D \l_shipout_box
112 }

```

(End of definition for __shipout_execute:.)

\shipout Overloading the \shipout primitive:

```

113 \cs_gset_eq:NN \shipout \__shipout_execute:

```

(End of definition for \shipout. This function is documented on page 1184.)

\l__shipout_group_level_tl Helper token list to record the group level at which __shipout_execute: is encountered.

```

114 \tl_new:N \l__shipout_group_level_tl

```

(End of definition for \l__shipout_group_level_tl.)

__shipout_execute_test_level: If the group level has changed then we are still constructing \l_shipout_box and to continue we need to wait until the current group has finished, hence the \tex_aftergroup:D.

```

115 \cs_new:Npn \__shipout_execute_test_level: {
116   \int_compare:nNnT
117     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
118     \tex_aftergroup:D \__shipout_execute_cont:
119 }

```

(End of definition for __shipout_execute_test_level:.)

__shipout_execute_cont: This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to __shipout_execute_main_cont:Nnnn; the first argument is the box to be shipped out.

```

120 \cs_new:Npn \__shipout_execute_cont: {
121   \__shipout_execute_main_cont:Nnnn
122   \l_shipout_box
123   { \hook_use:n {shipout/before} }
124   { \hook_if_empty:nF {shipout/foreground}
125     { \__shipout_add_foreground_picture:n
126       { \hook_use:n {shipout/foreground} } }

```

If the user hook for the background (shipout/background) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the \@kernel@before@shipout@background though. If the \@kernel@after@shipout@background needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```

127     \bool_lazy_and:nnF
128       { \hook_if_empty_p:n {shipout/background} }
129       { \tl_if_empty_p:N \@kernel@before@shipout@background }
130     { \__shipout_add_background_picture:n
131       { \@kernel@before@shipout@background
132         \hook_use:n {shipout/background}
133         \@kernel@after@shipout@background }
134     }
135   }
136   { \hook_use:n {shipout/after} }
137 }

```

(End of definition for __shipout_execute_cont:.)

`\_shipout_execute_main_cont:Nnnn`

When we have reached this point the shipout box has been processed and is available in `\l_shipout_box` and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (#1), within the shipout adding background and foreground material (#2) and after the shipout has happened (#3). These are passed as arguments because the same code without those hooks is also used when doing a “raw” shipout implemented by `\RawShipout`. The only hook that is always executed is that for the very last page, i.e., `shipout/lastpage`.

First we quickly check if it is void (can’t happen in the standard L^AT_EX output routine but `\shipout` might be called from a package that has some special processing logic). If it is void we aren’t shipping anything out and processing ends.⁶⁰

```

138 \cs_new:Npn \_shipout_execute_main_cont:Nnnn #1#2#3#4 {
139   \box_if_empty:NTF #1
140   { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
141   {

```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of `\protect` while we are running the hook code). We also save the current `\protect` state to restore it later.

```

142 %       \bool_gset_false:N \g__shipout_discard_bool % setting this would disable
143                                     % \DiscardShipoutBox on doc-level
144 \cs_set_eq:NN \_shipout_saved_protect: \protect
145 \set@typeset@protect

```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.⁶¹

```

146   \_shipout_get_box_size:N #1

```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```

147   #2

```

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`).

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_readonly_int` are in sync while the hook is executed (in the case that `totalpages` isn’t manually altered or through discarding pages that is).

```

148   \int_gincr:N \g_shipout_totalpages_int

```

The above hook might contain code that requests the page to be discarded so we now test for it.

```

149   \bool_if:NTF \g__shipout_discard_bool
150   { \@latex@info@no@line{Completed~ page~ discarded}
151     \bool_gset_false:N \g__shipout_discard_bool

```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset T_EX’s deadcycles so that it doesn’t complain about too many calls to the OR without any shipout.

```

152   \tex_deadcycles:D \c_zero_int

```

⁶⁰In that case we don’t reset the deadcycles, that would be up to the OR processing logic to do.

⁶¹This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

Todo: In atbegshi the box was dropped but is that actually needed? Or the resetting of \protect to its kernel value?

```

153 %          \group_begin:
154 %          \box_set_eq_drop:NN #1 #1
155 %          \group_end:
156 %          \cs_set_eq:NN \protect \exp_not:N
157      }

```

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```

158      { \box_if_empty:NTF #1
159        { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
160          \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
161        }

```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.⁶²

```

162      {
163        \int_gincr:N \g_shipout_readonly_int
164        \__shipout_debug:n {
165          \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
166            \space (target:~ \@abspage@last)}
167        }

```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```

168      \__shipout_get_box_size:N #1

```

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `\__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\specials` for integration at the top of the page they will be temporarily stored in a safe place and added later with `\__shipout_add_firstpage_specials:`.

```

169      \__shipout_run_firstpage_hook:

```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l_shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l_shipout_box`).

```

170      #3

```

We then run `\__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```

171      \__shipout_add_firstpage_specials:

```

⁶²Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```

172         \int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
173         { \bool_lazy_and:nnF
174           { \hook_if_empty_p:n {shipout/lastpage} }
175           { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
176           { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
177             on~ page~ \int_use:N \g_shipout_readonly_int } } }
178           \__shipout_add_foreground_box:n
179           { \UseHook{shipout/lastpage}
180             \@kernel@after@shipout@lastpage }

```

We record that we have handled the `shipout/lastpage` hook but only if we really did.

```

181         \bool_gset_true:N \g__shipout_lastpage_handled_bool
182       }
183     }
184     \hook_use:n {shipout}
185     \__shipout_finalize_box:

```

Finally we run the actual T_EX primitive for `shipout`. As that will expand delayed `\write` statements inside the page in which protected commands should not expand we first change `\protect` to the appropriate definition for that case.

```

186         \cs_set_eq:NN \protect \exp_not:N
187         \tex_shipout:D \box_use:N \l_shipout_box

```

The `\l_shipout_box` may contain the firstpage material if this was the very first `shipout`. That makes it unsuitable for reuse in another `shipout`, so as a safety measure the next command resets `\l_shipout_box` to its earlier state if that is necessary. On later pages this is then a no-op.

```

188         \__shipout_drop_firstpage_specials:

```

The `shipout/after` hook (if in #4) needs to run with `\protected` commands again being executed, because that hook will “typeset” material added at the top of the next page.

```

189         \set@typeset@protect
190         #4
191       }
192     }

```

Restore the value of `\protect` in case `\shipout` is called outside of the output routine (where it is automatically restored because of the implicit group).

```

193         \cs_set_eq:NN \protect \__shipout_saved_protect:
194       }
195     }

```

(End of definition for __shipout_execute_main_cont:Nnnn.)

`\__shipout_execute_raw:` This implements the “raw” `shipout` which bypasses the before, foreground, background and after hooks. It follows the same pattern than `\__shipout_execute_raw:` except that it finally calls `\__shipout_execute_main_cont:Nnnn` with three empty arguments. instead of the hook code.

```

196 \cs_set_protected:Npn \__shipout_execute_raw: {
197   \tl_set:Nc \l__shipout_group_level_tl
198     { \int_value:w \tex_currentgrouplevel:D }

```

```

199 \tex_afterassignment:D \__shipout_execute_test_level_raw:
200 \tex_setbox:D \l__shipout_raw_box
201 }
202 \cs_new:Npn \__shipout_execute_test_level_raw: {
203   \int_compare:nNt
204     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
205     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
206 }

```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```

207 \cs_new:Npn \__shipout_execute_nohooks_cont: {
208   \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
209   {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} } }
210   \box_set_eq:NN \l_shipout_box \l__shipout_raw_box {} {}
211 }

```

(End of definition for __shipout_execute_raw: and __shipout_execute_test_level_raw:.)

\RawShipout The interface name for raw shipout.

```

212 \cs_gset_eq:NN \RawShipout \__shipout_execute_raw:

```

(End of definition for \RawShipout. This function is documented on page 1184.)

__shipout_saved_protect: Remember the current \protect state.

```

213 \cs_new_eq:NN \__shipout_saved_protect: \protect

```

(End of definition for __shipout_saved_protect:.)

shipout/before Declaring all hooks for the shipout code.

```

      shipout 214 \hook_new:n{shipout/before}
      shipout 215 \hook_new:n{shipout}
      shipout/after 216 \hook_new:n{shipout/after}
      shipout/foreground 217 \hook_new:n{shipout/foreground}
      shipout/background 218 \hook_new:n{shipout/background}
      shipout/firstpage 219 \hook_new:n{shipout/firstpage}
      shipout/lastpage 220 \hook_new:n{shipout/lastpage}

```

(End of definition for shipout/before and others. These functions are documented on page 1185.)

\@kernel@after@shipout@lastpage And here are the internal kernel hooks going before or after the public ones where needed.

```

\@kernel@before@shipout@background
\@kernel@after@shipout@background
221 \let\@kernel@after@shipout@lastpage\@empty
222 \let\@kernel@before@shipout@background\@empty
223 \let\@kernel@after@shipout@background\@empty

```

(End of definition for \@kernel@after@shipout@lastpage, \@kernel@before@shipout@background, and \@kernel@after@shipout@background.)

`\_shipout_run_firstpage_hook:` There are three commands to handle the `shipout/firstpage` hook: `\__shipout_run_firstpage_hook:`, `\__shipout_add_firstpage_specials:` and `\__shipout_drop_firstpage_specials:`.

That hook is supposed to contain `\specials` and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\specials` show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed “first”, e.g., visible from code in `shipout/background` and this conflicts with adding the `\specials` late.

Therefore the processing is split into different parts: `\__shipout_run_firstpage_hook:` is done early and checks if there is any material in the hook.

```
224 \cs_new:Npn \__shipout_run_firstpage_hook: {
225   \hook_if_empty:NTF {shipout/firstpage}
```

If not then we define the other two commands to do nothing.

```
226   {
227     \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
228     \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
229   }
```

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```
230   {
231     \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
232   }
```

Once we are here we change the definition to do nothing next time and we also change the command used to implement `\AtBeginDvi` to become a warning and not add further material to a hook that is never used again.

```
233   \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
234   \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
235     \@latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
236                   \MessageBreak \string##1 }
237   }
238 }
```

(End of definition for `\__shipout_run_firstpage_hook:`.)

`\_shipout_add_firstpage_specials:` The `\__shipout_add_firstpage_specials:` then adds the `\specials` stored in `\l__shipout_firstpage_box` to the page to be shipped out when the time is ready. Note that if there was no material in the `shipout/firstpage` hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript header, or some metadata declaration, etc. so by default we assume there is something to do.

```
239 \cs_new:Npn \__shipout_add_firstpage_specials: {
```

First we make a copy of the `\l__shipout_box` that we can restore it later on.

```
240   \box_set_eq:NN \l__shipout_raw_box \l__shipout_box
```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```
241 \__shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }
```

After the actual shipout `\__shipout_drop_firstpage_specials:` is run to restore the earlier content of `\l_shipout_box` and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```
242 \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
243 }
```

The `\__shipout_drop_firstpage_specials:` is run after the shipout has occurred but before the shipout/afterpage hook is executed. That is the point where we have to restore the `\ShipoutBox` to its state without the shipout/firstpage material.

```
244 \cs_new:Npn \__shipout_drop_firstpage_specials: {
245   \box_set_eq:NN \l_shipout_box \l__shipout_raw_box
```

If there was no such material then `\__shipout_run_firstpage_hook:` will have changed the definition to a no-op already. Otherwise this is what we do here.

```
246   \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
247 }
```

(End of definition for `\__shipout_add_firstpage_specials:` and
`\__shipout_drop_firstpage_specials:.`)

`\l__shipout_firstpage_box` The box to hold any firstpage `\specials`.

```
248 \box_new:N \l__shipout_firstpage_box
```

(End of definition for `\l__shipout_firstpage_box`.)

`\g_shipout_lastpage_handled_bool` A boolean to signal if we have already handled the shipout/lastpage hook.

```
249 \bool_new:N \g__shipout_lastpage_handled_bool
```

(End of definition for `\g__shipout_lastpage_handled_bool`.)

`\__shipout_add_firstpage_material:Nn` This command adds material to the shipout/firstpage hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which it is called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

```
250 \cs_new:Npn \__shipout_add_firstpage_material:Nn #1#2 {
251   \AddToHook{shipout/firstpage}{#2}
252 }
```

(End of definition for `\__shipout_add_firstpage_material:Nn`.)

`\__shipout_get_box_size:N` Store the box dimensions in dimen registers.

Todo: This could/should perhaps be generalized to set height depth and width given an arbitrary box.

```
253 \cs_new:Npn \__shipout_get_box_size:N #1 {
254   \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
255   \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
256   \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
257   \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
258     { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
259 }
```

(End of definition for `\_shipout_get_box_size:N`.)

`\l_shipout_box_ht_dim` And here are the variables set by `\_shipout_get_box_size:N`.

```

\l_shipout_box_dp_dim 260 \dim_new:N \l_shipout_box_ht_dim
\l_shipout_box_wd_dim 261 \dim_new:N \l_shipout_box_dp_dim
\l_shipout_box_ht_plus_dp_dim 262 \dim_new:N \l_shipout_box_wd_dim
263 \dim_new:N \l_shipout_box_ht_plus_dp_dim

```

(End of definition for `\l_shipout_box_ht_dim` and others. These functions are documented on page 1185.)

`\g_shipout_discard_bool` Indicate whether or not the current page box should be discarded

```
264 \bool_new:N \g_shipout_discard_bool
```

(End of definition for `\g_shipout_discard_bool`.)

`\l__shipout_tmp_box` We need a box for the background and foreground material and a token register to
`\l__shipout_saved_badness_tl` remember badness settings as we disable them during the buildup below.

```

265 \box_new:N \l__shipout_tmp_box
266 \tl_new:N \l__shipout_saved_badness_tl

```

(End of definition for `\l__shipout_tmp_box` and `\l__shipout_saved_badness_tl`.)

`\_shipout_add_background_box:n` In standard L^AT_EX the shipout box is always a `\vbox` but here we are allow for other usage as well, in case some package has its own output routine.

```

267 \cs_new:Npn \_shipout_add_background_box:n #1
268 { \_shipout_get_box_size:N \l_shipout_box

```

But we start testing for a vertical box as that should be the normal case.

```

269 \box_if_vertical:NTF \l_shipout_box
270 {

```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```

271 \tl_set:Nc \l__shipout_saved_badness_tl
272 { \vfuzz=\the\vfuzz\relax
273 \vbadness=\the\vbadness\relax }
274 \vfuzz=\c_max_dim
275 \vbadness=\c_max_int

```

Then we reconstruct `\l_shipout_box` ...

```

276 \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
277 {

```

... the material in #1 is placed into a horizontal box with zero dimensions.

```

278 \hbox_set:Nn \l__shipout_tmp_box
279 { \l__shipout_saved_badness_tl #1 }
280 \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
281 \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
282 \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim

```

The we typeset that box followed by whatever was in `\l_shipout_box` before (unpacked).

```

283 \skip_zero:N \baselineskip
284 \skip_zero:N \lineskip
285 \skip_zero:N \lineskiplimit
286 \box_use:N \l__shipout_tmp_box
287 \vbox_unpack:N \l_shipout_box

```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```

288       \kern \c_zero_dim
289     }
290     \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
291     \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim

```

Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.

```

292     \l__shipout_saved_badness_tl
293   }
294   {

```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing F branch.

```

295     \box_if_horizontal:NT \l_shipout_box
296     {
297       \tl_set:Nc \l__shipout_saved_badness_tl
298       { \hfuzz=\the\hfuzz\relax
299         \hbadness=\the\hbadness\relax }
300       \hfuzz=\c_max_dim
301       \hbadness=\c_max_int
302       \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
303       {
304         \hbox_set:Nn \l__shipout_tmp_box
305         { \l__shipout_saved_badness_tl #1 }
306         \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
307         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
308         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
309         \box_move_up:nn
310         \l_shipout_box_ht_dim
311         { \box_use:N \l__shipout_tmp_box }
312         \hbox_unpack:N \l_shipout_box
313       }
314       \l__shipout_saved_badness_tl
315     }
316   }
317 }

```

(End of definition for `\__shipout_add_background_box:n`.)

`\__shipout_add_foreground_box:n` Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```

318 \cs_new:Npn \__shipout_add_foreground_box:n #1
319 {
320   \box_if_vertical:NTF \l_shipout_box
321   {
322     \tl_set:Nc \l__shipout_saved_badness_tl
323     { \vfuzz=\the\vfuzz\relax
324       \vbadness=\the\vbadness\relax }
325     \vfuzz=\c_max_dim
326     \vbadness=\c_max_int
327     \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
328     {

```

```

329         \hbox_set:Nn \l__shipout_tmp_box
330         { \l__shipout_saved_badness_tl #1 }
331         \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
332         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
333         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
334         \skip_zero:N \baselineskip
335         \skip_zero:N \lineskip
336         \skip_zero:N \lineskiplimit
337         \vbox_unpack:N \l_shipout_box
338         \kern -\l_shipout_box_ht_plus_dp_dim
339         \box_use:N \l__shipout_tmp_box
340         \kern \l_shipout_box_ht_plus_dp_dim
341     }
342     \l__shipout_saved_badness_tl
343     \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
344     \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
345 }
346 {
347     \box_if_horizontal:NT \l_shipout_box
348     {
349         \tl_set:Ne \l__shipout_saved_badness_tl
350         { \hfuzz=\the\hfuzz\relax
351           \hbadness=\the\hbadness\relax }
352         \hfuzz=\c_max_dim
353         \hbadness=\c_max_int
354         \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
355         {
356             \hbox_unpack:N \l_shipout_box
357             \kern -\box_wd:N \l_shipout_box
358             \hbox_set:Nn \l__shipout_tmp_box
359             { \l__shipout_saved_badness_tl #1 }
360             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
361             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
362             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
363             \box_move_up:nn { \box_ht:N \l_shipout_box }
364             { \box_use:N \l__shipout_tmp_box }
365             \kern \box_wd:N \l_shipout_box
366         }%
367         \l__shipout_saved_badness_tl
368     }
369 }
370 }

```

(End of definition for __shipout_add_foreground_box:n.)

__shipout_init_page_origins:
\c__shipout_horigin_tl
\c__shipout_vorigin_tl

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the begindocument hook that affects their setup.

```

371 \cs_new:Npn \__shipout_init_page_origins: {
372     \tl_const:Ne \c__shipout_horigin_tl
373     {
374         \cs_if_exist_use:NTF \pdfvariable { horigin }
375         { \cs_if_exist_use:NF \pdfhorigin { 1in } }

```

```

376     }
377     \tl_const:Ne \c__shipout_vorigin_tl
378     {
379         \cs_if_exist_use:NTF \pdfvariable { vorigin }
380         { \cs_if_exist_use:NF \pdfvorigin { 1in } }
381     }

```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```

382     \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
383 }

```

(End of definition for __shipout_init_page_origins:, \c__shipout_horigin_tl, and \c__shipout_vorigin_tl.)

`\__shipout_add_background_picture:n` Put a `picture` env in the background of the shipout box with its reference point in the top-left corner.

```

384 \cs_new:Npn \__shipout_add_background_picture:n #1 {
385     \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
386 }

```

(End of definition for __shipout_add_background_picture:n.)

`\__shipout_add_foreground_picture:n` Put a `picture` env in the foreground of the shipout box with its reference point in the top-left corner.

```

387 \cs_new:Npn \__shipout_add_foreground_picture:n #1 {
388     \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
389 }

```

(End of definition for __shipout_add_foreground_picture:n.)

`\shipout_discard:` Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case L^AT_EX looks ahead and is not using the position for on the next page).

```

390 \cs_new_protected:Npn \shipout_discard: {
391     \bool_gset_true:N \g__shipout_discard_bool
392 }

```

(End of definition for \shipout_discard:. This function is documented on page 1188.)

3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

`\g_shipout_readonly_int` We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

```

393 \int_new:N \g_shipout_readonly_int

```

For L^AT_EX 2_ε it is available as a command (i.e., a T_EX counter only).

```

394 \cs_new_eq:NN \ReadonlyShipoutCounter \g_shipout_readonly_int

```


(End of definition for `\g_shipout_readonly_int` and `\ReadonlyShipoutCounter`. These functions are documented on page 1189.)

`\g_shipout_totalpages_int` We count every shipout attempt (even those that are discarded) in this counter. It is not
`\c@totalpages` used in the code but may get used in user code.

```
395 \int_new:N \g_shipout_totalpages_int
```

For L^AT_EX 2_ε this is offered as a L^AT_EX counter so can be easily typeset inside the output routine to display things like “`\thepage/\thetotalpages`”, etc.

```
396 \cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
```

```
397 \cs_new:Npn \thetotalpages { \arabic{totalpages} }
```

(End of definition for `\g_shipout_totalpages_int` and `\c@totalpages`. These functions are documented on page 1189.)

`\@abspage@last` In `\@abspage@last` record the number of pages from the last run. This is written to the `.aux` and this way made available to the next run. In case there is no `.aux` file or the statement is missing from it we initialize it with the largest possible number in T_EX. We use this as the default because then we are inserting the `shipout/lastpage` on the last page (or after the last page) but not on page 1 for a multipage document.

```
398 \xdef\@abspage@last{\number\maxdimen}
```

(End of definition for `\@abspage@last`.)

`\enddocument` Instead of using the hooks `enddocument` and `enddocument/afterlastpage` we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside `\enddocument` there is a `\clearpage`. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don’t know the value from the last run, we assume that the current page is the right one. So we set `\@abspage@last` and as a result the next shipout will run the `shipout/lastpage` code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

```
\@kernel@after@enddocument 399 \g@addto@macro \@kernel@after@enddocument {
400   \int_compare:nNnT \@abspage@last = \maxdimen
401   {
```

We use L^AT_EX 2_ε coding as `\@abspage@last` is not an L3 name.

```
402   \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
403   }
404 }
```

After the last page has been shipped out, we force further `\write` calls to be always `\immediate` because we aren’t going to ship out any more pages. This goes before the `enddocument/afterlastpage` hook because that may contain such `\write` commands.

```
405 \g@addto@macro \@kernel@before@enddocument@afterlastpage {
406   \__shipout_force_immediate_writes:
407   % This is also the point where \__tag_lastpage_label: could be executed
408   % instead of going into enddocument/afterlastpage
409 }
```

\@kernel@after@enddocument@afterlastpage

Once the `\clearpage` has done its work inside `\enddocument` we know for sure how many pages this document has, so we record that in the `.aux` file for the next run.

```
410 \g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to run the `shipout/lastpage` is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
411 \int_compare:nNf \g_shipout_readonly_int = 0
412 {
```

This ends up in the `.aux` so we use L^AT_EX 2_ε names here.

Todo: This needs an interface for `\nofiles` in `expl3`, doesn't at the moment!

```
413 \if@filesw
414 \iow_now:Ne \@auxout {
415 \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
416 \fi
```

But we may have guessed wrongly earlier and have run it too early or we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy. In either case we should put out an appropriate “rerun” warning.

```
417 \bool_if:Ntf \g__shipout_lastpage_handled_bool
418 {
```

If the hook was already executed, we have to test if that total shipouts match the shipouts from last run (because that corresponds to the page it was executed). If not we output a warning.

```
419 \int_compare:nNf \@abspage@last = \g_shipout_readonly_int
420 {
421 \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed~
422 on~ wrong~ page~ (\@abspage@last\space not~
423 \int_use:N \g_shipout_readonly_int).\MessageBreak
424 Rerun~ to~ correct~ this}%
425 }
426 }
427 {
```

If the hook was not run, we need to add an extra page and place it there. However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
428 \bool_lazy_and:nnF
429 { \hook_if_empty_p:n {shipout/lastpage} }
430 { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
431 {
432 \global\advance\c@page\@ne
433 \tex_shipout:D\vbox to\textheight
434 {
435 \hbox:n { \UseHook{shipout/lastpage}
436 \@kernel@after@shipout@lastpage }
```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```

437         \_shipout_excuse_extra_page:
438         \null
439     }

```

At this point we also signal to L^AT_EX's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```

440         \cs_gset_eq:NN \@extra@page@added \relax
441     }
442 }
443 }
444 }

```

(End of definition for \enddocument and others.)

`\_shipout_excuse_extra_page:` Say mea culpa ...

```

445 \cs_new:Npn \_shipout_excuse_extra_page: {
446     \vfil
447     \begin{center}
448         \bfseries Temporary~ page!
449     \end{center}
450     \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
451     correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
452     should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
453     page~ has~ been~ added~ to~ receive~ it.
454     \par
455     If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
456     surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
457     how~ many~ pages~ to~ expect~ for~ this~ document.
458     \vfil
459 }

```

(End of definition for _shipout_excuse_extra_page:.)

\PreviousTotalPages

In the preamble before the aux file was read `\PreviousTotalPages` is always zero.

`\@kernel@before@begindocument`

```

460 \def\PreviousTotalPages{0}

```

In the aux file there should be an update for `\@abspage@last` recording the number of pages from the previous run. If not that macro holds the value of `\maxdimen`. So we test for it and update `\PreviousTotalPages` if there was a real value. This should happen just before the `begindocument` hook is executed so that the value can be used inside that hook.

```

461 \g@addto@macro\@kernel@before@begindocument
462     {\ifnum\@abspage@last<\maxdimen
463         \xdef\PreviousTotalPages{\@abspage@last}\fi}

```

(End of definition for \PreviousTotalPages and \@kernel@before@begindocument. These functions are documented on page 1189.)

4 Legacy L^AT_EX 2_ε interfaces

\DiscardShipoutBox Request that the next shipout box is to be discarded.

```
464 \cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(End of definition for \DiscardShipoutBox. This function is documented on page 1188.)

\AtBeginDvi If we roll forward from an earlier kernel \AtBeginDvi is defined so we better not use \cs_new_protected:Npn here.

```
465 \cs_set_protected:Npn \AtBeginDvi
466 { \__shipout_add_firstpage_material:Nn \AtBeginDvi }
```

(End of definition for \AtBeginDvi. This function is documented on page 1187.)

\DebugShipoutsOn
\DebugShipoutsOff

```
467 \cs_new_eq:NN \DebugShipoutsOn \shipout_debug_on:
468 \cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(End of definition for \DebugShipoutsOn and \DebugShipoutsOff. These functions are documented on page 1189.)

5 Internal commands needed elsewhere

These internal commands use double and triple @ signs so we need to stop getting them translated to the module name.

```
469 <@@=>
```

Some internals needed elsewhere.

```
\@expl@@@shipout@add@firstpage@material@@Nn
\@expl@@@shipout@add@background@box@@n
\@expl@@@shipout@add@foreground@box@@n
\@expl@@@shipout@add@background@picture@@n
\@expl@@@shipout@add@foreground@picture@@n

470 \cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@Nn
471 \__shipout_add_firstpage_material:Nn
472 \cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
473 \__shipout_add_background_box:n
474 \cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
475 \__shipout_add_foreground_box:n
476 \cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
477 \__shipout_add_background_picture:n
478 \cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
479 \__shipout_add_foreground_picture:n
```

(End of definition for \@expl@@@shipout@add@firstpage@material@@Nn and others.)

```
480 \ExplSyntaxOff
481 </2ekernel | latexrelease>
482 <latexrelease>\EndIncludeInRelease
```

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
483 <latexrelease>\IncludeInRelease{0000/00/00}%
484 <latexrelease> \{ \shipout \} { Hook management (shipout) } %
485 <latexrelease>
```

If we roll forward then `\tex_shipout:D` may not be defined in which case `\shipout` does have it original definition and so we must not `\let` it to something else which is `\relax`!

```

486 <latexrelease>\ifcsname tex_shipout:D\endcsname
487 <latexrelease>\expandafter\let\expandafter\shipout
488 <latexrelease> \csname tex_shipout:D\endcsname
489 <latexrelease>\fi
490 <latexrelease>
491 <latexrelease>\let \RawShipout\@undefined
492 <latexrelease>\let \ShipoutBox\@undefined
493 <latexrelease>\let \ReadonlyShipoutCounter \@undefined
494 <latexrelease>\let \c@totalpages \@undefined
495 <latexrelease>\let \thetotalpages \@undefined
496 <latexrelease>
497 <latexrelease>\let \DiscardShipoutBox \@undefined
498 <latexrelease>\let \DebugShipoutsOn \@undefined
499 <latexrelease>\let \DebugShipoutsOff \@undefined
500 <latexrelease>
501 <latexrelease>\DeclareRobustCommand \AtBeginDvi [1]{%
502 <latexrelease> \global \setbox \@begindvibox
503 <latexrelease> \vbox{\unvbox \@begindvibox #1}%
504 <latexrelease>}
505 <latexrelease>
506 <latexrelease>\let \AtBeginShipout \@undefined
507 <latexrelease>\let \AtBeginShipoutNext \@undefined
508 <latexrelease>
509 <latexrelease>\let \AtBeginShipoutFirst \@undefined
510 <latexrelease>
511 <latexrelease>\let \ShipoutBoxHeight \@undefined
512 <latexrelease>\let \ShipoutBoxDepth \@undefined
513 <latexrelease>\let \ShipoutBoxWidth \@undefined
514 <latexrelease>

```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```

515 <latexrelease>
516 <latexrelease>\let \AtEndDvi \@undefined

```

We do not reenale a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```

517 %\reenable@package@load{atenddvi}
518 <latexrelease>
519 <latexrelease>\EndIncludeInRelease
520 <*2kernel>

```

`\_shipout_force_immediate_writes:` Change all writes to be immediate. To be used after the final page has been shipped out.

```

521 </2ekernel>
522 <*2ekernel | latexrelease>
523 <latexrelease>\IncludeInRelease{2025/06/01}%
524 <latexrelease> {\_shipout_force_immediate_writes:}{Force immediate writes}%
525 \ExplSyntaxOn
526 \cs_new:Npn \_shipout_force_immediate_writes: {
    Need to check if any variants are manually defined and if so adjust them too — not
    done!

```

```

527 \cs_gset_eq:NN \iow_shipout:Nn \iow_now:Nn
528 \cs_gset_eq:NN \lua_shipout:n \lua_now:n
529 \cs_gset:Npn \write {\tex_immediate:D \tex_write:D}

```

As the writes will happen without a page break, reset the page number so they reference the last page.

```

530 \global\advance\c@page\m@ne
531 }
532 \ExplSyntaxOff
533 </2ekernel | latexrelease>
534 <latexrelease>\EndIncludeInRelease
535 <latexrelease>\IncludeInRelease{0000/00/00}%
536 <latexrelease> {\_shipout_force_immediate_writes:}{Force immediate writes}%

```

We want a definition for this even when doing rollback so that it can stay in the kernel hook without forcing a rollback there as well.

```

537 <latexrelease>\ExplSyntaxOn
538 <latexrelease> \cs_new_eq:NN \_shipout_force_immediate_writes: \prg_do_nothing:
539 <latexrelease>\ExplSyntaxOff
540 <latexrelease>\EndIncludeInRelease
541 <*2ekernel>

```

(End of definition for _shipout_force_immediate_writes:.)

6 Package emulation for compatibility

6.1 Package `atenddvi` emulation

\AtEndDvi This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```

542 </2ekernel>
543 <*2ekernel | latexrelease>
544 <latexrelease>\IncludeInRelease{2020/10/01}%
545 <latexrelease> {\AtEndDvi}{atenddvi emulation}%
546 \ExplSyntaxOn
547 \cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
548 \ExplSyntaxOff

```

As the package is integrate we prevent loading (no need to roll that back):

```

549 \disable@package@load{atenddvi}
550 {\PackageWarning{atenddvi}
551 {Functionality of this package is already\MessageBreak
552 provided by LaTeX.\MessageBreak\MessageBreak
553 It is there no longer necessary to load it\MessageBreak

```

```

554         and you can safely remove it.\MessageBreak
555         Found on}}
556 \
```

2kernel | latexrelease)

```

557 \latexrelease\EndIncludeInRelease
558 \latexrelease\IncludeInRelease{0000/00/00}%
559 \latexrelease\AtEndDvi{atendddvi emulation}%
560 \latexrelease\let \AtEndDvi \@undefined
561 \latexrelease\EndIncludeInRelease
562 \*2kernel)

(End of definition for \AtEndDvi. This function is documented on page 1187.)
563 \
```

2kernel)

6.2 Package atbegshi emulation

```

564 \*atbegshi-ltx)
565 \ProvidesPackage{atbegshi-ltx}
566 [2021/01/10 v1.0c
567 Emulation of the original atbegshi^^Jpackage with kernel methods]

```

\AtBeginShipoutBox

```

568 \let \AtBeginShipoutBox \ShipoutBox

```

(End of definition for \AtBeginShipoutBox. This function is documented on page 1190.)

\AtBeginShipoutInit

Compatibility only, we aren't delaying ...

```

569 \let \AtBeginShipoutInit \@empty

```

(End of definition for \AtBeginShipoutInit. This function is documented on page 1191.)

\AtBeginShipout

Filling hooks

\AtBeginShipoutNext

```

570 \protected\long\def\AtBeginShipout #1{\AddToHook{shipout/before}{#1}}
571 \protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}

```

(End of definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 1191.)

\AtBeginShipoutFirst

Slightly more complex as we need to know the name of the command under which the shipout/firstpage hook is filled.

```

572 \protected \def \AtBeginShipoutFirst
573 {\@expl@@@shipout@add@firstpage@material@@Nn \AtBeginShipoutFirst}

```

(End of definition for \AtBeginShipoutFirst. This function is documented on page 1191.)

\AtBeginShipoutDiscard

Just a different name.

```

574 \let \AtBeginShipoutDiscard \DiscardShipoutBox

```

(End of definition for \AtBeginShipoutDiscard. This function is documented on page 1191.)

\AtBeginShipoutAddToBox

We don't expose them.

\AtBeginShipoutAddToBoxForeground

```

575 \let \AtBeginShipoutAddToBox

```

\AtBeginShipoutUpperLeft

```

576 \@expl@@@shipout@add@background@box@@N

```

\AtBeginShipoutUpperLeftForeground

```

577 \let \AtBeginShipoutAddToBoxForeground
578 \@expl@@@shipout@add@foreground@box@@N

```

```

579 \let \AtBeginShipoutUpperLeft
580       \@@expl@@@shipout@add@background@picture@@n
581 \let \AtBeginShipoutUpperLeftForeground
582       \@@expl@@@shipout@add@foreground@picture@@n

```

(End of definition for \AtBeginShipoutAddToBox and others. These functions are documented on page 1190.)

\AtBeginShipoutOriginalShipout This offers the raw \shipout primitive of the engine. A page shipped out with this is not counted by \ReadonlyShipoutCounter counter and thus the mechanism to place \specials at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```

583 \ExplSyntaxOn
584 \cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D

```

(End of definition for \AtBeginShipoutOriginalShipout. This function is documented on page 1190.)

\ShipoutBoxHeight **\ShipoutBoxWidth** **\ShipoutBoxDepth** This is somewhat different from the original in atbegshi where \ShipoutBoxHeight etc. only holds the \the\ht<box> value. This may have some implications in some use cases and if that is a problem then it might need changing.

```

585 \cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
586 \cs_new:Npn \ShipoutBoxDepth { \dim_use:N \l_shipout_box_dp_dim }
587 \cs_new:Npn \ShipoutBoxWidth { \dim_use:N \l_shipout_box_wd_dim }
588 \ExplSyntaxOff

```

(End of definition for \ShipoutBoxHeight, \ShipoutBoxWidth, and \ShipoutBoxDepth.)

```

589 </atbegshi-ltx>

```

If the package is requested we substitute the one above:

```

590 <*2kernel>
591 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
592 </2kernel>

```

6.3 Package everyshi emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

```

593 <@@=>

```


File 54

ltoutput.dtx

1 Output Routine and float handling

```
1 <*2ekernel>
2 \message{output,}
```

1.1 Historical notes on the algorithm and commands

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
*****
*                               *
*                               *
*****
```

PAGE LAYOUT PARAMETERS

`\topmargin` : Extra space added to top of page.
`@twoside` : boolean. T if two-sided printing
`\oddsidemargin` : IF `@twoside = T`
THEN extra space added to left of odd-numbered
pages.
ELSE extra space added to left of all pages.
`\evensidemargin` : IF `@twoside = T`
THEN extra space added to left of even-numbered
pages.
`\headheight` : height of head
`\headsep` : separation between head and text
`\footskip` : distance separation between baseline of last
line of text and baseline of foot.
Note difference between `\footSKIP` and `\headSEP`.
`\textheight` : height of text on page, excluding head and foot
`\textwidth` : width of printing on page
`\columnsep` : IF `@twocolumn = T`
THEN width of space between columns
`\columnseprule` : IF `@twocolumn = T`
THEN width of rule between columns (0 if none).
`\columnwidth` : IF `@twocolumn = T`
THEN $(\text{\textwidth} - \text{\columnsep})/2$
ELSE `\textwidth`
It is set by the `\twocolumn` and
`\onecolumn` commands.
`\@textbottom` : Command executed at bottom of vbox holding text of
page (including figures). The `\raggedbottom`
command almost \let's this to `\vfil` (actually sets
it to `\vskip \z@ plus.0001fil`).
Should have depth 0pt.

`\@texttop` : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages. Let to `\relax` by `\raggedbottom` and `\flushbottom`.

Page layout must initialize `\@colht` and `\@colroom` to `\textheight`.

PAGE STYLE PARAMETERS:

`\floatsep` : Space left between floats.
`\textfloatsep` : Space between last top float or first bottom float and the text.
`\topfigrule` : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertical mode right before the `\textfloatsep` skip separating the floats from the text. Must occupy zero vertical space. (See `\footnoterule`.)
`\botfigrule` : Same as `\topfigrule`, but put after the `\textfloatsep` skip separating text from the floats at bottom of page.
`\intextsep` : Space left on top and bottom of an in-text float.
`\dblfloatsep` : Space between double-column floats.
`\dbltextfloatsep` : Space between top double-column floats and text.
`\dblfigrule` : Similar to `\topfigrule`, but for double-column floats.
`\@fptop` : Glue to go at top of float column – must be 0pt + stretch
`\@fpsep` : Glue to go between floats in a float column.
`\@fpbot` : Glue to go at bottom of float column – must be 0pt + stretch
`\@dblfpsep`, `\@dblfpbot` : Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use `\insert\footins`.

PAGE LAYOUT SWITCHES AND MACROS

`@twocolumn` : Boolean. T if two columns per page globally.

PAGE STYLE MACROS AND SWITCHES

`\@oddhead` : IF `@twoside = T`
 THEN macro to generate head of odd-numbered pages.
 ELSE macro to generate head of all pages.
`\@evenhead` : IF `@twoside = T`
 THEN macro to generate head of even-numbered

pages.

`\@oddfoot` : IF @twoside = T
THEN macro to generate foot of odd-numbered
pages.
ELSE macro to generate foot of all pages.

`\@evenfoot` : IF @twoside = T
THEN macro to generate foot of even-numbered
pages.

`@specialpage` : boolean. T if current page is to have a special
format.

`\@specialstyle` : If its value is foo then
IF @specialpage = T
THEN the command `\ps@foo` is executed to
temporarily reset the page style parameters
before composing the current page.
This command should execute only `\def`'s and
`\edef`'s, making only local definitions.

FLOAT PLACEMENT PARAMETERS

The following parameters are set by the macro `\@floatplacement`.
When `\@floatplacement` is called,

`\@colht` is the height of the page or column being built. I.e.:
* For single-column page it equals `\textheight`.
* For double-column page it equals `\textheight` - height
of double-column floats on page.

Note that some are set globally and some locally:

`\@topnum` :=G Maximum number of floats allowed on the top of a
column.

`\@toproom` :=G Maximum amount of top of column devoted to floats—
excluding `\textfloatsep` separation below the floats
and `\floatsep` separation between them. For
two-column output, should be computed as a function
of `\@colht`.

`\@botnum`, `\@botroom`
: Analogous to above.

`\@colnum` :=G Maximum number of floats allowed in a column,
including in-text floats.

`\@textmin` :=L Minimum amount of text (excluding footnotes) that
must appear on a text page.
%% 27 Sep 85 : made local to
%% `\@addtocurcol` and `\@addtonextcol`
It is now also used locally in processing double
floats.

`\@fpmin` :=L Minimum height of floats in a float column.

The macro `\dblfloatplacement` sets the following parameters.

`\dbltopnum` :=G Maximum number of double-column floats allowed at
the top of a two-column page.

`\dbltoproom` :=G Maximum height of double-column floats allowed at

top of two-column page.

`\@fpm` :=L Minimum height of floats in a float column.

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

`\@fptop` :=L `\@dblftop`

`\@fpsep` :=L `\@dblfpsep`

`\@fpbot` :=L `\@dblfpbot`

OUTPUT ROUTINE VARIABLES

`\@colht` : The total height of the current column. In single column style, it equals `\textheight`. In two-column style, it is `\textheight` minus the height of the double-column floats on the current page. MUST BE INITIALIZED TO `\textheight`.

`\@colroom` : The height available in the current column for text and footnotes. It equals `\@colht` minus the height of all floats committed to the top and bottom of the current column.

`\@textfloatsheight` : The total height of in-text floats on the current page.

`\footins` : Footnote insertion number.

`\@maxdepth` : Saved value of TeX's `\maxdepth`. Must be set when any routine sets `\maxdepth`.

CALLING THE OUTPUT ROUTINE

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty $<$ or $= -10000$ in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

penalty	reason
-10000	<code>\pagebreak</code> <code>\newpage</code>
-10001	<code>\clearpage</code> (<code>\penalty -10000 \vbox{}</code> <code>\penalty -10001</code>)
-10002	float insertion, called from horizontal mode
-10003	float insertion, called from vertical mode.
-10004	float insertion.

Note: A float or marginpar puts the following sequence in the output list:

- (i) a penalty of -10004,
- (ii) a null `\vbox`
- (iii) a penalty of -10002 or -10003.

This solves two special problems:

1. If the float comes right after a `\newpage` or `\clearpage`, then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

THE OUTPUT ROUTINE

FUNCTIONS USED IN THE OUTPUT ROUTINE:

`\@outputpage` : Produces an output page with the contents of box `\@outputbox` as the text part.
 Also sets `\@colht :=G \textheight`.
 The page style is determined as follows.
 IF `@thispagestyle = true`
 THEN use `\thispagestyle` style
 ELSE use ordinary page style.

`\@tryfcolumn\FLIST` : Tries to form a float column composed of floats from `\FLIST` (if nonempty) with the following parameters:
 `\@colht` : height of box
 `\@fpmin` : minimum height of floats in the box
 `\@fpsep` : interfloat space
 `\@fptop` : glue at top of box
 `\@fpbot` : glue at bottom of box.
 If it succeeds, then it does the following:
 * `\@outputbox :=L` the composed float box.
 * `@fcolmade :=G true`
 * `\FLIST :=G \FLIST - floats put in box`
 * `\@freelist :=G \@freelist + floats put in box`
 If it fails, then:
 * `@fcolmade :=G false`

NOTE: BIT MUST BE A SINGLE TOKEN!

`\@makefcolumn \FLIST` : Same as `\@tryfcolumn` except that it fails to make a float column only if `\FLIST` is empty. Otherwise, it makes a float column containing at least the first box in `\FLIST`, disregarding `\@fpmin`.

`\@startcolumn` :
 Calls `\@tryfcolumn\@deferlist`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:
 * Globally sets `\@toplist` and `\@botlist` to floats from `\@deferlist` to go at top and bottom of column, deleting them from `\@deferlist`. It does this using `\@colht` as the total height, the page style parameters `\@floatsep` and `\@textfloatsep`, and the float placement parameters `\@topnum`, `\@toproom`, `\@botnum`, `\@botroom`, `\@colnum` and `\textfraction`.

* Globally sets `\@colroom` to `\@colht` minus the height of the added floats.

`\@startdblcolumn` :

Calls `\@tryfcolumn\@dbldeferlist{8}`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

- * Globally sets `\@dbltoplist` to floats from `\@dbldeferlist` to go at top and bottom of column, deleting them from `\@dbldeferlist`. It does this using `\textheight` as the total height, and the parameters `\@dblfloatsep`, etc.
- * Globally sets `\@colht` to `\textheight` minus the height of the added floats.

`\@combinefloats` : Combines the text from box

`\@outputbox` with the floats from `\@toplist` and `\@botlist`, putting the new box in `\@outputbox`. It uses `\floatsep` and `\textfloatsep` for the appropriate separations. It puts the elements of `\TOPLIST` and `\BOTLIST` onto `\freelist`, and makes those lists null.

`\@makecol` : Makes the contents of `\box255` plus the accumulated footnotes, plus the floats in `\@toplist` and `\@botlist`, into a single column of height `\@colht` (unless the page height has been locally changed), which it puts into box `\@outputbox`. It puts boxes in `\@midlist` back onto `\freelist` and restores `\maxdepth`.

`\@opcol` : Outputs a column whose text is in box `\@outputbox`

If `@twocolumn = false`, then it calls `\@outputpage`, sets `\@colht :=G \textheight`, and calls `\@floatplacement`.

If `@twocolumn = true`, then:

If `@firstcolumn = true`, then it puts box `\@outputbox` into `\@leftcolumn` and sets `@firstcolumn :=G false`.

If `@firstcolumn = false`, then it puts out the current two-column page, any possible two-column float pages, and determines `\@dbltoplist` for the next page.

USER COMMANDS THAT CALL OR AFFECT THE OUTPUT

ROUTINE

`\newpage == BEGIN \par\vfil\penalty -10000 END`

`\clearpage == BEGIN \newpage
\write -1{} % Part of hack to make sure no`

```

\hbox{} % \write's get lost.
\penalty -10001
END

\cleardoublepage == BEGIN \clearpage
                    if @twoside = true and c@page is even
                    then \hbox{} \newpage fi
END

\twocolumn[BOX] : starts a new page, changing to twocolumn setting
                    and puts BOX in a parbox of width \textwidth across the top.
                    Useful for full-width titles for double-column pages.
                    SURPRISE: The stretch from \@dbltextfloatsep will be inserted
                    between the BOX and the top of the two columns.

```

FLOAT-HANDLING MECHANISMS

The float environment obtains an insertion number B from the `\@freelist` (see below for a description of list manipulation), puts the float into box B and sets `\count B` to a FLOAT SPECIFIER. For a normal (not double-column) float, it then causes a page break in one of the following two ways:

- In outer hmode: `\vadjust{\penalty -10002}`
- In vmode : `\penalty -10003`.

For a double-column float, it puts B onto the `\@dbldeferlist`. The float specifier has two components:

- * A PLACEMENT SPECIFICATION, describing where the float may be placed.
- * A TYPE, which is a power of two—e.g., figures might be type 1 floats, tables type 2 floats, programs type 4 floats, etc.

The float specifier is encoded as follows, where bit 0 is the least significant bit.

Bit	Meaning
0	1 iff the float may go where it appears in the text.
1	1 iff the float may go on the top of a page.
2	1 iff the float may go on the bottom of a page.
3	1 iff the float may go on a float page.
4	1 unless the PLACEMENT includes a !
5	1 iff a type 1 float
6	1 iff a type 2 float
etc.	

A negative float specifier is used to indicate a marginal note.

MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

A FLOAT LIST consisting of the floats in boxes `\boxa ... \boxN` has the form:

`\@elt \boxa ... \@elt \boxN`

where `\boxI` is defined by

`\newinsert\boxI`

Normally, `\@elt` is `\let` to `\relax`. A test can be performed on the entire float list by locally `\def`'ing `\@elt` appropriately and executing the list.

This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
\@next \CS \LIST {NONEMPTY}{EMPTY} == %% NOTE: ASSUME \@elt
= \relax
  BEGIN  assume that \LIST == \@elt \B1 ... \@elt \Bn
        if n = 0
        then  EMPTY
        else  \CS      :=L \B1
              \LIST :=G \@elt \B2 ... \@elt \Bn
              NONEMPTY
        fi
  END
```

`\@bitor\NUM\LIST` : Globally sets switch `@test` to the disjunction for all `I` of bit `log2 \NUM` of the float specifiers of all the floats in `\LIST`.
 I.e., `@test` is set to true iff there is at least one float in `\LIST` having bit `log2 \NUM` of its float specifier equal to 1.

Note: `log2 [(\count I)/32]` is the bit number corresponding to the type of float `I`. To see if there is any float in `\LIST` having the same type as float `I`, you run `\@bitor` with
`\NUM = [(\count I)/32] * 32`.

```
\@bitor\NUM\LIST ==
  BEGIN
    @test :=G false
    { \@elt \CTR == if \NUM <> 0 then
                      if \count\CTR / \NUM is odd
                      then @test := true      fi fi
    \LIST
  }
  END
```

`\@cons\LIST\NUM` : Globally sets `\LIST := \LIST * \@elt \NUM`


```

\@cons\LIST\NUM ==
  BEGIN { \@elt == \relax
    \LIST :=G \LIST \@elt \NUM
  }

```

BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

```

\@freelist      : List of empty boxes for placing new floats.
\@toplist       : List of floats to go at top of current column.
\@midlist       : List of floats in middle of current column.
\@botlist       : List of floats to go at bottom of current column.
\@deferlist     : List of floats to go after current column.
\@dbltoplist    : List of double-col. floats to go at top of current
                  page.
\@dbldeferlist  : List of double-column floats to go on subsequent
                  pages.

```

FLOAT-PLACEMENT ALGORITHMS

```

\@addtobot : Tries to put insert \@currbox on \@botlist.
             Called only when:
               * \ht BOX < \@colroom
               * type of \@currbox not on \@deferlist
               * \@colnum > 0
               * @insert = false
             If it succeeds, then:
               * sets @insert true
               * decrements \@botroom by \ht BOX
               * decrements \@botnum and \@colnum by 1
               * decrements \@colroom by \ht BOX + either \floatsep
                 or \textfloatsep, as appropriate.
               * sets \maxdepth to 0pt      % <- as of 2025/06/01
                                           %      no longer the case

\@addtotoporbot : Tries to put insert \@currbox on \@toplist or
                  \@botlist.
                  Called only under same conditions as \@addtobot.
                  If it succeeds, then:
                    * sets @insert true
                    * decrements \@toproom or \@botroom by \ht BOX
                    * decrements \@colnum and either \@topnum or
                      \@botnum by 1
                    * decrements \@colroom by \ht BOX + \floatsep
                      or \textfloatsep, as appropriate.

\@addtocurcol : Tries to add \@currbox to current column, setting
                 @insert true if it succeeds, false otherwise.
                 It will add \@currbox to top only if bit 0 of

```

```

\count \@currbox is 0, and to the bottom only if
bit 0 = 0 or an earlier float of the same type is
put on the bottom.
If the float is put in the text, then
\penalty\interlinepenalty is put
right after the float, before the following \vskip,
and \outputpenalty :=L 0.

\@addtonextcol : Tries to add \@currbox to the next column, setting
                 @insert true if it succeeds, false otherwise.

\@addtodblcol : Tries to add \@currbox to the next double-column page,
                 adding it to \@dbltoplist if it succeeds and
                 \@dbldeferlist if it fails.

\@addmarginpar ==
BEGIN
  if \@currlist nonempty
    then remove \@marbox from \@currlist
      add \@marbox and \@currbox to \@freelist
      %% NOTE: \@currbox = left box
    else LaTeX error: ? %% shouldn't happen
  fi
  \@tempcnta := 1      %% 1 = right, -1 = left
  if @twocolumn = true
    then if @firstcolumn = true
      then \@tempcnta := -1
    fi
    else if @mparswitch = true
      then if count0 odd
        else \@tempcnta := -1
      fi
    fi
    if @reversemargin = true
      then \@tempcnta := -\@tempcnta
    fi
  fi
  if \@tempcnta < 0 then \box\@marbox :=G \box\@currbox
  fi
  \@tempdima :=L maximum(\@mparbottom - \@pageht
                        + ht of \@marbox, 0)
  if \@tempdima > 0 then LaTeX warning: 'marginpar moved' fi
  \@mparbottom :=G \@pageht + \@tempdima + depth of \@marbox
                + \marginparpush
  \@tempdima :=L \@tempdima - ht of \@marbox
  \box\@marbox :=G \box\@currbox
                    \vbox { \vskip \@tempdima
                          \box\@marbox
                          }

```

```

height of \@marbox :=G depth of \@marbox :=G 0
\kern -\@pagedp
\nointerlineskip
\hbox{ if @tempcnta > 0 then \hskip \columnwidth
                                \hskip \marginparsep
                                else \hskip -\marginparsep
                                \hskip -\marginparwidth
                                fi
      \box\@marbox \hss
    }
\nobreak
\nointerlineskip
\hbox{\vrule height 0 width 0 depth \@pagedp}
END

```

End of historical L^AT_EX 2.09 comments.

1.2 Core definitions

Floats and marginpars add a lot of dead cycles.

```

3 \maxdeadcycles = 100
4 \let\@elt\relax
5 \def\@next#1#2#3#4{\ifx#2\@empty #4\else
6   \expandafter\@xnext #2\@@#1#2#3\fi}
7 \def\@xnext \@elt #1#2\@@#3#4{\def#3{#1}\gdef#4{#2}}
8 \def\@testfalse{\global\let@if@test\iffalse}
9 \def\@testtrue {\global\let@if@test\iftrue}
10 \@testfalse
11 \def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
12   \@tempcnta #1\relax #2}}

```

RmS 91/11/22: Added test for \count#1 = 0. Suggested by Chris Rowley.

```

13 \def\@xbitor #1{\@tempcntb \count#1
14   \ifnum \@tempcnta =\z@
15   \else
16     \divide\@tempcntb\@tempcnta
17     \ifodd\@tempcntb \@testtrue\fi
18   \fi}

```

1.2.1 Definition of float boxes

```

19 \let\@elt\newinsert
20 \def\@freelist{%
21   \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
22   \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
23   \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
24   \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}
25 \@freelist
26 \def\reserved@a{%
27   \@elt\bx@S\@elt\bx@T\@elt\bx@U\@elt\bx@V

```

```

28 \elt\bx@W\elt\bx@X\elt\bx@Y\elt\bx@Z
29 \elt\bx@AA\elt\bx@BB\elt\bx@CC\elt\bx@DD\elt\bx@EE
30 \elt\bx@FF\elt\bx@GG\elt\bx@HH\elt\bx@II\elt\bx@JJ
31 \elt\bx@KK\elt\bx@LL\elt\bx@MM\elt\bx@NN
32 \elt\bx@OO\elt\bx@PP\elt\bx@QQ\elt\bx@RR
33 \elt\bx@SS\elt\bx@TT\elt\bx@UU\elt\bx@VV
34 \elt\bx@WW\elt\bx@XX\elt\bx@YY\elt\bx@ZZ}
35 \reserved@a
36 \def\elt{\noexpand\elt\noexpand}
37 \edef\freelist{\freelist\reserved@a}
38 \let\reserved@a\relax
39 \let\elt\relax
40 \gdef\@toplist{}
41 \gdef\@botlist{}
42 \gdef\@midlist{}
43 \gdef\@currlist{}
44 \gdef\@deferlist{}
45 \gdef\@dbltoplist{}

```

The new algorithm stores page wide floats together with column floats in a single \@deferlist list. We keep \@dbldeferlist initialized as empty so that packages that are testing for deferred floats can use the same code for old or new float handling.

```
46 \gdef\@dbldeferlist{}
```

1.2.2 Page layout parameters

```

47 \newdimen\topmargin
48 \newdimen\oddsidemargin
49 \newdimen\evensidemargin
50 \let\themargin=\oddsidemargin
51 \newdimen\headheight
52 \newdimen\headsep
53 \newdimen\footskip
54 \newdimen\textheight
55 \newdimen\textwidth
56 \newdimen\columnwidth
57 \newdimen\columnsep
58 \newdimen\columnseprule
59 \newdimen\marginparwidth
60 \newdimen\marginparsep
61 \newdimen\marginparpush

```

\AtBeginDvi We use a box register in which to put stuff that must appear before anything else in the .dvi file.

\@begindvibox The stuff in the box should not add any typeset material to the page when it is unboxed.

This interface is no longer used. Instead a new one is inside `ltshipout.dtx`. We only keep the box in case some old code refers to it directly (or we do some rollback).

```

62 \newbox\@begindvibox
63 %\DeclareRobustCommand \AtBeginDvi [1]{%
64 % \global \setbox \@begindvibox
65 % \vbox{\unvbox \@begindvibox #1}%
66 %}

```

(End of definition for \AtBeginDvi and \@begindvibox. These functions are documented on page 1187.)

`\@maxdepth` This is not the right place to set this; it needs to be set in a class/style file when `\maxdepth` is set.

Also, many settings to `\maxdepth` should be to `\@maxdepth`, probably?

```
67 \newdimen\@maxdepth
68 \@maxdepth = \maxdepth
```

(End of definition for \@maxdepth.)

`\paperheight` New `\paper...` registers.

```
\paperwidth 69 \newdimen\paperheight
70 \newdimen\paperwidth
```

(End of definition for \paperheight and \paperwidth.)

`\stockheight` New `\stock...` registers.

```
\stockwidth 71 \newdimen\stockheight
72 \newdimen\stockwidth
```

(End of definition for \stockheight and \stockwidth.)

`\if@insert` Local switches first:

```
\if@fcolmade 73 \newif \if@insert
```

`\if@specialpage` These should definitely be global:

```
\if@firstcolumn 74 \newif \if@fcolmade
\if@twocolumn 75 \newif \if@specialpage \@specialpagefalse
\if@twoside
```

`\if@reversemargin` These should be global but are not always set globally in other files.

```
\if@mparswitch 76 \newif \if@firstcolumn \@firstcolumntrue
\col@number 77 \newif \if@twocolumn \@twocolumnfalse
```

Not sure about these: two questions. Should things which must apply to a whole document be local or global (they probably should be ‘preamble only’ commands)? Are these three such things?

```
78 \newif \if@twoside \@twosidefalse
79 \newif \if@reversemargin \@reversemarginfalse
80 \newif \if@mparswitch \@mparswitchfalse
```

This counter has been imported from ‘multicol’.

```
81 \newcount \col@number
82 \col@number \@ne
```

(End of definition for \if@insert and others.)

1.2.3 Internal registers

```
83 \newcount\@topnum
84 \newdimen\@toproom
85 \newcount\@dbltopnum
86 \newdimen\@dbltoproom
87 \newcount\@botnum
88 \newdimen\@botroom
89 \newcount\@colnum
90 \newdimen\@textmin
91 \newdimen\@fpmin
92 \newdimen\@colht
93 \newdimen\@colroom
94 \newdimen\@pageht
95 \newdimen\@pagedp
96 \newdimen\@mparbottom \@mparbottom\z@
97 \newcount\@currtype
98 \newbox\@outputbox
99 \newbox\@leftcolumn
100 \newbox\@holdpg
101 \def\@thehead{\@oddhead} % initialization
102 \def\@thefoot{\@oddfoot}
```

1.2.4 Page break commands

\clearpage The tests at the beginning are an experimental attempt to avoid a completely empty page after a `\twocolumn[...]`. This prevents the text from the argument vanishing into a float box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```
103 \def\clearpage{%
104   \ifvmode
105     \ifnum \@dbltopnum =\m@ne
106       \ifdim \pagetotal <\topskip
107         \hbox{}%
108       \fi
109     \fi
110   \fi
111   \newpage
```

After the above `\newpage` there may be some still unwritten `\write`'s issued in the output routine and we need to make sure that they are not lost when `\@docclearpage` produces float pages. Therefore the following trick is applied: We add a harmless `\write` followed by an empty `\vbox` and then call the output routine again through a special penalty. That triggers `\@docclearpage` in the OR which splits off a zero-sized chunk from box 255 (which should go up to, but not including, the `\vbox`), then `\unvbox` that (which should contain any `\write`'s, and then throw away the rest of 255 (which should just contain the `\vbox`). That explains the strange `\write-1{}`: that statement introduces a necessary breakpoint so that the split is done before the `\vbox`. If it would not be there and *there are real \write's*, then the split would grab the whole of 255 and so would put the `\vbox` back on the page and we would end with an extra page only containing that empty `\vbox`.

Deep breath—the above is my (Frank's) attempt of a slightly longer version of Leslie's cryptic comment earlier in the file that this `\write` is “part of a hack to make sure that no `\write`'s are lost”.

This `\write` needs to be non-immediate even if we are supposedly already past the last page because it has to end up on the page sent to the output routine if for some reason that happens.

```

112 \@@write\m@ne{}%
113 \vbox{}%
114 \penalty -\@Mi
115 }

116 \let\@@write\write

```

(End of definition for \clearpage.)

`\cleardoublepage`

```

117 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
118 \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
119 </2ekernel>

```

(End of definition for \cleardoublepage.)

`\onecolumn`

```

120 <*2ekernel | fltrace>
121 \def\onecolumn{%
122 \clearpage
123 \global\columnwidth\textwidth
124 \global\hsize\columnwidth
125 \global\linewidth\columnwidth
126 \global\@twocolumnfalse
127 \col@number \@ne
128 \@floatplacement}

```

(End of definition for \onecolumn.)

`\newpage` The two checks at the beginning ensure that an item label or run-in section title immediately before a `\newpage` get printed on the correct page, the one before the page break.

All three tests are largely to make error processing more robust; that is why they all reset the flags explicitly, even when it would appear that this would be done by a `\leavevmode`.

```

129 </2ekernel | fltrace>
130 <latexrelease>\IncludeInRelease{2025/06/01}%
131 <latexrelease> \{\newpage\}{Check depth of page}%
132 <*2ekernel | latexrelease | fltrace>
133 \def \newpage {%
134 \if@noskipsec
135 \ifx \@nodocument\relax
136 \leavevmode
137 \global \@noskipsecfalse
138 \fi
139 \fi
140 \if@inlabel
141 \leavevmode
142 \global \@inlabelfalse
143 \fi
144 \if@nobreak \@nobreakfalse \everypar{}\fi

```

The `\vfil` at the end of the macro before the break penalty will normally result in the page being run short, even with `\flushbottom` in effect (in contrast to the behavior of `\pagebreak`). However, if there is some explicit stretch on the page, say, a `\vfill`, it has the undesired side-effect, that the last line will not align at its baseline if it contains characters going below the baseline, as the value of `\prevdepth` is no longer taken into account by T_EX. So we have to back up by that amount (or by `\maxdepth` if it is really huge), to mimic the normal behavior without the `\newpage`.

In 2017 this was fixed by doing the backing-up here. However, the problem with that is that if the page contains footnotes the `\skip\ý` is measured from the baseline of the last line and not (as on all other pages) from the bottom of the last line. One can argue that measuring from the baseline is better, but if that is done it should be done everywhere not just on some pages.

Thus this backing up should only happen when there are no footnotes. The problem is that, at the point when `\newpage` is called, it is not known whether or not there are footnotes; that is only determined in the output routine. We have therefore moved this code to the newly introduced output routine sockets.

An explicit `\par` token is required so that after environments like lists `\par` is reset to its standard meaning and does not suppress indentation on the next page.

```

145 \par
146 % \ifdim\prevdepth>\z@
147 % \vskip -%
148 % \ifdim\prevdepth>\maxdepth
149 % \maxdepth
150 % \else
151 % \prevdepth
152 % \fi
153 % \fi
154 \vfil
155 \penalty -\@M}
156 </2ekernel | latexrelease | fltrace>
157 <latexrelease>\EndIncludeInRelease

158 <latexrelease>\IncludeInRelease{2017/04/15}%
159 <latexrelease> \newpage{Check depth of page}%
160 <latexrelease>\def \newpage {%
161 <latexrelease> \if@noskipsec
162 <latexrelease> \ifx \@nodocument\relax
163 <latexrelease> \leavevmode
164 <latexrelease> \global \@noskipsecfalse
165 <latexrelease> \fi
166 <latexrelease> \fi
167 <latexrelease> \if@inlabel
168 <latexrelease> \leavevmode
169 <latexrelease> \global \@inlabelfalse
170 <latexrelease> \fi
171 <latexrelease> \if@nobreak \@nobreakfalse \everypar{}\fi
172 <latexrelease> \par
173 <latexrelease> \ifdim\prevdepth>\z@
174 <latexrelease> \vskip -%
175 <latexrelease> \ifdim\prevdepth>\maxdepth
176 <latexrelease> \maxdepth
177 <latexrelease> \else
178 <latexrelease> \prevdepth

```



```

179 <latexrelease> \fi
180 <latexrelease> \fi
181 <latexrelease> \vfil
182 <latexrelease> \penalty -\@M}

183 <latexrelease>\EndIncludeInRelease
184 <latexrelease>\IncludeInRelease{0000/00/00}%
185 <latexrelease> \newpage}{Check depth of page}%
186 <latexrelease>\def \newpage {%
187 <latexrelease> \if@noskipsec
188 <latexrelease> \ifx \@nodocument\relax
189 <latexrelease> \leavevmode
190 <latexrelease> \global \@noskipsecfalse
191 <latexrelease> \fi
192 <latexrelease> \fi
193 <latexrelease> \if@inlabel
194 <latexrelease> \leavevmode
195 <latexrelease> \global \@inlabelfalse
196 <latexrelease> \fi
197 <latexrelease> \if@nobreak \@nobreakfalse \everypar{}\fi
198 <latexrelease> \par
199 <latexrelease> \vfil
200 <latexrelease> \penalty -\@M}
201 <latexrelease>\EndIncludeInRelease
202 <*2ekernel|fltrace>

```

(End of definition for \newpage.)

`\@emptycol` It may be better to use an invisible rule rather than an empty box here.

```
203 \def \@emptycol {\vbox{}\penalty -\@M}
```

(End of definition for \@emptycol.)

`\twocolumn` There are several bug fixes to the two-column stuff here.
`\@topnewpage`

```

204 \def \twocolumn {%
205   \clearpage
206   \global\columnwidth\textwidth
207   \global\advance\columnwidth-\columnsep
208   \global\divide\columnwidth\tw@
209   \global\hsize\columnwidth
210   \global\linewidth\columnwidth
211   \global\@twocolumntrue
212   \global\@firstcolumntrue
213   \col@number \tw@

```

There is no reason to put a `\@dblfloatplacement` here since `\@topnewpage` ignores these settings. The `\@floatplacement` is needed in case this comes after some changes.

```

214   \ifnextchar [\@topnewpage\@floatplacement
215 }

```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```

216 \long\def \@topnewpage [#1]{%
217   \@nodocument
218   \@next\@currbox\@freelist{}\}%

```

```

219 \global \setbox\@currbox
220 \color@vbox
221 \normalcolor
222 \vbox {%
223 \hsize\textwidth
224 \@parboxrestore
225 \col@number \@ne
226 #1%
227 \vskip -\dbltextfloatsep
228 }%
229 \color@endbox

```

Added size test and warning message; perhaps we should use an error message.

```

230 \ifdim \ht\@currbox>\textheight
231 \ht\@currbox \textheight
232 \fi

```

This next line is not essential but it is more robust to make this value non-zero, in case of weird errors.

This next bit is what is needed from `\@addtodblcol`, plus some extra checks for error trapping.

```

233 \global \count\@currbox \tw@
234 \@tempdima -\ht\@currbox
235 \advance \@tempdima -\dbltextfloatsep
236 \global \advance \@colht \@tempdima
237 \ifx \@dbltoplist \@empty
238 \else
239 \latexerror{Float(s) lost}\@ehb
240 \let \@dbltoplist \@empty
241 \fi
242 \@cons \@dbltoplist \@currbox

```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```

243 \global \@dbltopnum \m@ne
244 (*trace)
245 \fl@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
246 (/trace)

```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present; but note that this value is larger than that used when checking that page is too full of normal floats.

If there is little room left we just force a page-break, OK? This involves producing two empty columns. The second empty column may be produced by `\output`, in which case an extra, misleading, warning will be generated, OK? (This happens only when there is too little room left on the page for any float.) Otherwise (i.e. if the size is such that it is allowed as a normal float) the extra `\@emptycol` will be invoked in the second column by the conditional code guarded by the `\if@firstcolumn` test.

I now think that the cut-off point here should be `3\baselineskip`, but we make it a bit less so that 3 lines of text will be allowed, OK?

Since this happens only when there is nothing on the page but the ‘top-box’, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

Here we need two page-ends since both columns need to be empty.

```

247 \ifdim \@colht<2.5\baselineskip
248 \latex@warning@no@line {Optional argument of \noexpand\twocolumn
249 too tall on page \thepage}%
250 \emptycol
251 \if@firstcolumn
252 \else
253 \emptycol
254 \fi
255 \else
256 \global \vsize \@colht
257 \global \@colroom \@colht
258 \@floatplacement
259 \fi
260 }

```

(End of definition for \twocolumn and \topnewpage.)

2 The L^AT_EX output routine

2.1 Hooks and replaceable code blocks

To support packages that want to augment aspects of the output routine we offer a number of hooks (where several packages can add code) as well as some sockets (that can only be changed by one package or through options like those of the `footmisc` package).

2.1.1 Output routine hooks

build/page/before, build/page/after These two hooks enable packages to prepend or append code to the page processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal L^AT_EX 2_ε `\@outputpage` command, respectively.

build/page/reset Packages that set up special conventions for text in the main galley (such as catcode changes, etc.) can use this hook to undo these changes within the output routine, so that they aren't applied to unrelated material, e.g., the text for running header or footers.

build/column/before, build/column/after These two hooks enable packages to prepend or append code to the column processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal L^AT_EX 2_ε `\@makecol` command, respectively.

2.1.2 Replaceable code blocks (sockets)

To cater for different layouts with respect to text, footnotes, and bottom-floats placements there are sockets for now. They are sockets not hooks, because the overall layout can only be controlled by one package, i.e., the last setting wins.

build/column/outputbox (0 arguments) In code for this socket the `\@outputbox` (holding the galley text for the current column or page) is augmented by attaching floats and footnote areas together with appropriate spacing.

Prior to calling the socket the output routine has already decided which floats go into which area and which get deferred. Therefore, the assumption is that the code in the socket attaches all areas that contain floats. If this is not done, then the order of floats is likely to be screwed up unless unused floats are moved to the defer list in an appropriate way (for now we don't offer any interface for that scenario).

Before the code in the socket is run, any existing glue at the bottom of the `\@outputbox` is removed and stored in a safe place. If needed, it can be reinserted with one of the helper commands.

To support setting this up the following helper commands are available:

`\@outputbox@append (1 argument)`

This general purpose command alters the `\@outputbox` box by appending material to it.

`\@outputbox@appendfil (0 arguments)` Append a `\vfil` to the `\@outputbox`.

Based on the plug in `build/column/baselineattach` either measured from the bottom or from the baseline of the last line in the box.

`\@outputbox@appendfootnotes (0 arguments)`

This command appends the footnotes to the `\@outputbox` (if there are any). If not, then it does nothing.

`\@outputbox@attachfloats (0 arguments)`

`\@outputbox@attachtopfloats (0 arguments)`

`\@outputbox@attachbottomfloats (0 arguments)`

Attaching top and bottom floats can usually be done in one go, but for special layouts we might want more control so we provide also separate commands.

`\@outputbox@reinsertbskip (0 arguments)`

Reinsert the bottom skip of the `\@outputbox` that was saved before in `\@makecol`.

Testing for existence of material

There are a number of helpers to run conditional code depending on whether or not there are footnotes or bottom floats. They are `\@if@footnotes@TF` and `\@if@bottomfloats@TF` (names are likely to change).

This socket cannot be empty but needs appropriate code; a set of suitable plugs for it are already given in the kernel. These are

space-footnotes-floats After the galley text there is a vertical `\vfil` followed by any footnotes followed by the bottom floats, if any.

footnotes-space-floats As before but the `\vfil` is between footnotes and floats.

floats-space-footnotes Floats are directly after the text, then a space and then footnotes at the bottom.

space-floats-footnotes Both floats and footnotes are pushed to the bottom with footnotes last.⁶³

⁶³There are two more permutations, but neither of them has ever been requested so they aren't set up by default — doing that in a class would be trivial though.

floats-footnotes All excess space is distributed across the existing glue on the page, e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

footnotes-floats As the previous one but floats and footnotes are swapped. This is the L^AT_EX default for newer document, i.e., this plug is assigned to the socket when `\DocumentMetadata` is used.

footnotes-floats-legacy As the previous one but L^AT_EX's bottom skip bug is not corrected, i.e., in ragged bottom designs where footnotes are supposed to be directly attached to the text, they suddenly appear at the bottom of the page when the page is ended with `\newpage` or `\clearpage`. While this is clearly a bug, it was the case since the days of L^AT_EX 2.09; thus for compatibility we continue to support this behavior.

build/column/footnotes (0 arguments) This socket is used to manipulate the footnote material inside `\box\footins`. If it contains code, it is supposed to do some processing of that box and then write the result back into it (and nothing else!). By default it does nothing, i.e., has the `noop` assigned.

If (short) footnotes are run as a paragraph this socket gets the plug `para` assigned which is defined elsewhere.

build/column/baselineattach (0 arguments) By default, footnotes are attached after the last line of text by adding a `\vskip\skip\footins`, i.e., the space is measured from the bottom of `\@outputbox` (plug `off`). Bottom floats are attached in a similar way by using `\vskip\textfloatsep`.

The socket can get the plug `on` assigned, in which case the skip starts at the baseline of the last line in `\@outputbox`, if there is one; otherwise it starts still from the bottom of the box.

2.1.3 Tagging sockets

The following sockets are used to implement tagging. They are used via `\UseTaggingSocket` which turns them off when tagging is disabled. For each of them a `default` plug is implemented that holds the tagging code.

build/column/outputbox (0 arguments) This socket is used to add any missing tagging structures to the `\@outputbox` box, if necessary.

build/column/footins (0 arguments) This socket is used to add any missing tagging structures to `\footins` box, if necessary.

build/page/header (2 arguments) This socket receives the content of the formatted page header as its second argument and adds the necessary tagging around it. The first argument is empty as no special setup is necessary.

build/page/footer (2 arguments) This socket receives the content of the formatted page footer as its second argument and adds the necessary tagging around it. The first argument is empty as no special setup is necessary.

2.1.4 Output routine commands

`\output` This needs some small adjustments. We cannot guarantee that the float mechanism will
`\@specialoutput` interact correctly with this stuff, but that mechanism does not always work properly with footnotes already.

RmS 91/09/29:

added reset of `\par` to the output routine. This avoids problems when the output routine is called within a list where `\par` may be a no-op.

```
261 \output {%
262   \let \par \@par
263   \ifnum \outputpenalty<-\@M
264     \@specialoutput
265   \else
266     \@makecol
267     \@opcol
```

Moved to `\@opcol: \@floatplacement`.

```
268   \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```
269   \@whilesw \if@fcolmade \fi
270   {%
271   (*trace)
272     \fl@trace{PAGE: float \if@twocolumn column \else page \fi
273               completed}%
274   (/trace)
275     \@opcol\@startcolumn}%
276   \fi
277   \ifnum \outputpenalty>-\@Miv
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved; but this is the natural place at which to set the `vsize` and a test would need to be done anyway. A check has been added to ensure that there really has been a change in the value of `\@colroom`.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

The `twocolumn` case does not need any extra code here since this is the `\output` itself; in the second column there will still not be enough room left so `\@emptycol` will be executed again when the OR is called by the page builder when it gets to the penalty inserted by the first execution. (The page-builder is never invoked whilst the OR is being executed since it builds a inner `vlist`; thus any conditional code for the two-column case within `\output` may not get executed with the correct value of `\if@firstcolumn`.

```
278   \ifdim \@colroom<1.5\baselineskip
279     \ifdim \@colroom<\textheight
280       \@latex@warning@no@line {Text page \thepage\space
281                               contains only floats}%
282     \@emptycol
283   %     \if@twocolumn
284   %     \if@firstcolumn
```

```

285 %           \else
286 %           \@emptycol
287 %           \fi
288 %           \fi
289     \else
290       \global \vsize \@colroom
291     \fi
292   \else
293     \global \vsize \@colroom
294   \fi
295 \else
296   \global \vsize \maxdimen
297 \fi
298 }

```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CHANGES TO \@specialoutput:

* \penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments.
(Changed 23 Oct 86)

* Definition of \@specialoutput changed 26 Feb 88 so \@pageht and \@pagedp aren't changed for a marginal note.
(Change suggested by Chris Rowley.)

End of historical L^AT_EX 2.09 comments.

```

299 \gdef\@specialoutput{%
300   \ifnum \outputpenalty>-\@Mii
301     \@docclearpage
302   \else
303     \ifnum \outputpenalty<-\@Miii
304       \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
305       \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
306     \else

```

Note that \boxmaxdepth should not be set here since we wish to record the natural depth of the holdpg box.

This is changed so as to not lose anything, such as writes and marks, which may get into box 255 and should be returned to the list. This should only happen when the first penalty in the mechanism is discarded and therefore \@holdpg should always be void in this case. This can happen because a penalty is discarded whenever there is no box on the list.

It was just: \setbox\@tempboxa \box \@cclv.

The last box which is removed is the box put there by the double-penalty mechanism. The \unskip then removes the \topskip which is put there since the box is the first on the page.

```

307   \global \setbox\@holdpg \vbox{%
308     \unvbox\@holdpg
309     \unvbox\@cclv

```

We must now remove the box added by the float mechanism and the \topskip glue therefore added above it by T_EX.

```

310     \setbox\@tempboxa \lastbox
311   \unskip

```

312 }%

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```
313 \pagedp \dp\@holdpg
314 \pageht \ht\@holdpg
315 \unvbox \@holdpg
316 \@next\@currbox\@currlist{%
317 \ifnum \count\@currbox>\z@
```

Putting the whole size into `\@pageht` (see above).

```
318 \advance \@pageht \@pagedp
319 \ifvoid\footins \else
320 \advance \@pageht \ht\footins
321 \advance \@pageht \skip\footins
322 \advance \@pageht \dp\footins
323 \fi
324 \ifvbox \@kludgeins
```

We want to make the adjustment due to this insert only if the non-star form is used. The *-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```
325 \ifdim \wd\@kludgeins=\z@
326 \advance \@pageht \ht\@kludgeins
327 <{*trace}
328 \fl@trace {Extra size added: \the \ht\@kludgeins}%
329 </trace>
330 \fi
331 \fi
```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```
332 \@reinserts
333 \@addtocurcol
334 \else
335 \@reinserts
336 \@addmarginpar
337 \fi
338 }\@latexbug
```

A 2e change: use `\addpenalty` instead of `\penalty` here. Some penalty is needed to create a potential break-point immediately after the reinserts (or the marginal). Otherwise there can be no possibility to break here and this can cause the reinserts or the marginal to appear on the next page (which is often incorrect). However, if the nobreak flag is true, a `\nobreak` must be correct.

```
339 \ifnum \outputpenalty<\z@
340 \if@nobreak
341 \nobreak
342 \else
343 \addpenalty \interlinepenalty
344 \fi
345 \fi
346 \fi
347 \fi
```



```

348 }
349 </2ekernel | fltrace>

```

(End of definition for \output and \@specialoutput.)

\@testwrongwidth Test if the float box has the wrong width when trying to place it into some area. (Actually the test is for a conventional depth setting rather than for the width of the float. For that reason the box depth was explicitly tailored when the float was created).

\f@depth

```

350 <latexrelease>\IncludeInRelease{2015/01/01}%
351 <latexrelease>          {\@testwrongwidth}{float order in 2-column}%
352 <*2ekernel | latexrelease | fltrace>

353 \def\@testwrongwidth #1{%
354   \ifdim\dp#1=\f@depth
355   <*trace>
356     \fl@trace{\string#1
357               \ifdim\f@depth=\z@ single \else double \fi
358               column float -- ok}%
359   </trace>
360   \else
361     \global\@testtrue
362   <*trace>
363     \fl@trace{\string#1
364               \ifdim\f@depth=\z@ double \else single \fi
365               column float -- wrong}%
366   </trace>
367   \fi}%

```

Normally looking for single column floats, which have zero depth.

```

368 \let\f@depth\z@

369 </2ekernel | latexrelease | fltrace>
370 <latexrelease>\EndIncludeInRelease
371 <latexrelease>\IncludeInRelease{0000/00/00}%
372 <latexrelease>          {\@testwrongwidth}{float order in 2-column}%
373 <latexrelease>\let\@testwrongwidth\@undefined
374 <latexrelease>\let\f@depth\@undefined
375 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@testwrongwidth and \f@depth.)

\@docclearpage This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed; but even more urgent is a bug-free version (see, for example, pr/3528).

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a very bad bug since, for example, a kludge insert will be in quite the wrong place and, worse, be irremovable and uncancelable.

All the remaining changes are replacing the double column defer list or inserting the extra test \@testwrongwidth{<box>} at suitable places. That is at places where a box is taken off the deferlist.

```

376 <latexrelease>\IncludeInRelease{2015/01/01}{\@docclearpage}%
377 <latexrelease>          {float order in 2-column}%
378 <*2ekernel | latexrelease>
379 \def \@docclearpage {%
380   \ifvoid\footins

```

```

381     \ifvbox\@kludgeins
382     {\setbox \@tempboxa \box \@kludgeins}%
383 < *trace>
384     \fl@trace {kludgeins box made void}%
385 < /trace>
386     \fi
387     \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
388     \setbox\@tempboxa\box\@cclv
389     \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
390     \global \let \@toplist \@empty
391     \global \let \@botlist \@empty
392     \global \@colroom \@colht
393     \ifx \@currlist\@empty
394     \else
395         \@latexerror{Float(s) lost}\@ehb
396         \global \let \@currlist \@empty
397     \fi
398     \@makefcolumn\@deferlist
399     \@whiles\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
400     \if@twocolumn
401         \if@firstcolumn
402             \xdef\@deferlist{\@dbltoplist\@deferlist}%
403             \global \let \@dbltoplist \@empty
404             \global \@colht \textheight
405             \begingroup
406                 \@dblfloatplacement
407                 \@makefcolumn\@deferlist
408                 \@whiles\if@fcolmade \fi{\@outputpage
409                     \@makefcolumn\@deferlist}%
410             \endgroup
411         \else
412             \vbox{}\clearpage
413         \fi
414     \fi

```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\doclearpage` will now no longer output all floats.

```

415     \ifx\@deferlist\@empty \else\clearpage \fi
416     \else
417         \setbox\@cclv\vbox{\box\@cclv\vfil}%
418         \@makecol\@opcol
419         \clearpage
420     \fi
421 }%
422 < /2ekernel | latexrelease>
423 < latexrelease>\EndIncludeInRelease
424 < latexrelease>\IncludeInRelease{0000/00/00}{\@doclearpage}%
425 < latexrelease>                                     {float order in 2-column}%
426 < latexrelease>\def \@doclearpage {%
427 < latexrelease>     \ifvoid\footins

```

We empty any left over kludge insert box here; this is a temporary fix. It should perhaps be applied to one page of cleared floats, but who cares? The whole of this stuff needs

completely redoing for many such reasons.

```

428 <latexrelease>      \ifvbox\@kludgeins
429 <latexrelease>      {\setbox \@tempboxa \box \@kludgeins}%
430 <*trace>
431 <latexrelease>      \fl@trace {kludgeins box made void}%
432 </trace>
433 <latexrelease>      \fi
434 <latexrelease>      \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
435 <latexrelease>      \setbox\@tempboxa\box\@cclv
436 <latexrelease>      \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

437 <latexrelease>      \global \let \@toplist \@empty
438 <latexrelease>      \global \let \@botlist \@empty
439 <latexrelease>      \global \@colroom \@colht
440 <latexrelease>      \ifx \@currlist\@empty
441 <latexrelease>      \else
442 <latexrelease>      \latexerr{Float(s) lost}\@ehb

443 <latexrelease>      \global \let \@currlist \@empty
444 <latexrelease>      \fi
445 <latexrelease>      \@makefcolumn\@deferlist
446 <latexrelease>      \@whiles\if@fcolmade \fi
447 <latexrelease>      {\@opcol\@makefcolumn\@deferlist}%
448 <latexrelease>      \if@twocolumn
449 <latexrelease>      \if@firstcolumn
450 <latexrelease>      \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

451 <latexrelease>      \global \let \@dbltoplist \@empty
452 <latexrelease>      \global \@colht \textheight
453 <latexrelease>      \begingroup
454 <latexrelease>      \dblfloatplacement
455 <latexrelease>      \@makefcolumn\@dbldeferlist
456 <latexrelease>      \@whiles\if@fcolmade \fi
457 <latexrelease>      {\@outputpage\@makefcolumn\@dbldeferlist}%
458 <latexrelease>      \endgroup
459 <latexrelease>      \else
460 <latexrelease>      \vbox{}\clearpage
461 <latexrelease>      \fi
462 <latexrelease>      \fi
463 <latexrelease>      \else
464 <latexrelease>      \setbox\@cclv\vbox{\box\@cclv\vfil}%
465 <latexrelease>      \@makecol\@opcol
466 <latexrelease>      \clearpage
467 <latexrelease>      \fi
468 <latexrelease>      }%
469 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@doclearpage.)

\@opcol Several changes in detail here.

```

470 <*2ekernel | fltrace>
471 \def \@opcol {%
472   \if@twocolumn

```

The funny-looking internal commands are interfacing with the new marks mechanism. We make sure (elsewhere) that those are always defined, even when we roll back, so here we add them unconditionally. This still need turning into a hook or config point eventually:

```

473 \expl@@@mark@update@dblcol@structures@@
474 \outputdblcol
475 \else
476 \expl@@@mark@update@singlecol@structures@@
477 \outputpage
478 <*trace>
479 \fl@trace{PAGE: one column (float? see above) page completed}%
480 </trace>

```

Not needed since it comes after `\@outputpage`:

```

481 % \global\@colht\textheight
482 \fi

```

These do not need to be done every time `\@opcol` is used: they should be grouped together since they all need to be done at the end of the non-special output routine, or at the end of a clearpage one.

```

483 \global \@mparbottom \z@ \global \@textfloatsheight \z@
484 \@floatplacement
485 }
486 </2ekernel | fltrace>

```

(End of definition for \@opcol.)

```

487 <*2ekernel | latexrelease>
488 <latexrelease>\IncludeInRelease{2025/06/01}%
489 <latexrelease> {\@makecol}{\@makecol adding hooks/sockets}%

```

`\@makecol` `\@makecol` is shortened a lot, basically all the hardwired code in the middle has moved into a socket.

```

490 \def \@makecol {%

```

A number of packages want to prepend code to `\@makecol`; this is the hook for that:

```

491 \UseHook {build/column/before}%

```

Save away box 255 as `\@outputbox` to make it available for further adjustments.

```

492 \setbox\@outputbox \box\@cclv

```

The only real addition is the next command which either does nothing or removes an infinite glue from the bottom of the `\@outputbox`.

```

493 \@outputbox@removebskip

```

Now a kernel hook for tagging that adjusts the content of `\@outputbox`, if necessary. At this point it just contains the material from the galley.

```

494 \UseTaggingSocket{build/column/outputbox}%

```

When this code is run any “here” floats in the `\@outputbox` are already handled, so we recycle their registers and put them back to the `\@freelist`.

```

495 \let\@elt\relax
496 \xdef\@freelist{\@freelist\@midlist}%
497 \global \let \@midlist \@empty

```

Here we have the configurable part. This socket is supposed to add floats, footnotes and stretchable vertical space as appropriate to the `\@outputbox`. It is used by packages such as `footmisc` to implement different layout, e.g., footnotes above or below bottom floats, etc.

```
498 \UseSocket {build/column/outputbox}%
```

Then we deal with any `\enlargethispage` or run the normal code to build a column.

```
499 \ifvbox\@kludgeins
500   \@make@specialcolbox
501 \else
502   \@make@normalcolbox
503 \fi
504 \global \maxdepth \@maxdepth
```

Finally, another hook for external packages or classes that want to augment or alter the output routine by appending to `\@makecol`.

```
505 \UseHook {build/column/after}%
506 }
```

(End of definition for \@makecol.)

`\@outputbox@depth` We need to know the depth of `\@outputbox` once in a while. Rather than using a temp dimen (as it was done in the past), we give it a proper register.

The value of this register is only correct inside of `\outputbox@append`, elsewhere you can't rely on it!

```
507 \newdimen\@outputbox@depth
```

(End of definition for \@outputbox@depth.)

`\@make@normalcolbox` Taken out of `\@makecol` for readability.

```
508 \def \@make@normalcolbox {%
509   \setbox\@outputbox \vbox to\@colht {%
510     \@texttop
511     \@outputbox@depth \dp\@outputbox
512     \unvbox \@outputbox
```

The `\vskip -\@outputbox@depth` ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the maxdepths at appropriate times.

If `\@textbottom` ends with a box or rule of non-zero depth then this skip adjustment should be done again after it.

```
513   \vskip -\@outputbox@depth
514   \@textbottom
515 }%
516 }
```

(End of definition for \@make@normalcolbox.)

`\@make@specialcolbox` Make the colbox when `\enlargethispage` was used.

```
517 </2kernel | latexrelease>
518 <*2kernel | latexrelease | fltrace>
519 \def \@make@specialcolbox {%
520 <*trace>
```

```

521 \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
522         dp \the\dp\@kludgeins\space
523         wd \the\wd\@kludgeins}%
524 
```

Note that in this case (the *-version), the height of the `\@kludgeins` box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size `\@colht` using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the $\TeX$ level!).

This needs $\TeX$ 3 otherwise `\pageshrink` is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of 0pt?, on the extra space added to force shrinking.

```

528 \advance \@tempdima -\ht\@outputbox
529 \advance \@tempdima \pageshrink
530 (*trace)
531 \fl@trace {Natural ht of col: \the \ht\@outputbox}%
532 \fl@trace {\string \@colht: \the \@colht}%
533 \fl@trace {Pageshrink added: \the \pageshrink}%
534 \fl@trace {Hence, space added: \the \@tempdima}%
535 
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```

541 \else
542 \advance \@tempdima -\ht\@kludgeins
543 (*trace)
544 \fl@trace {Natural ht of col: \the \ht\@outputbox}%
545 \fl@trace {\string \@colht: \the \@colht}%
546 \fl@trace {Extra size added: -\the \ht \@kludgeins}%
547 \fl@trace {Hence, height of inner box: \the \@tempdima}%
548 \fl@trace {Max? pageshrink available: \the \pageshrink}%
549 
```

This type of final packaging could be done always; this may simplify all of this page-makeup.

It is not necessary to set `\boxmaxdepth` here since the `\@outputbox` ends with glue.

```

550     \setbox \@outputbox \vbox to \@colht {%
551         \vbox to \@tempdima {%
552             \unvbox \@outputbox
553             \@textbottom}%
554         \vss}%
555     \fi
556     {\setbox \@tempboxa \box \@kludgeins}%
557     <*trace>
558     \fl@trace {kludgeins box made void}%
559     </trace>
560 }
561 </2ekernel | latexrelease | fltrace>
562 <*2ekernel | latexrelease>

```

(End of definition for \@make@specialcolbox.)

\@outputbox@removebskip This is really a bug fix for the kernel (from the 2.09 days) but one we only make by default in new documents that are using `\DocumentMetadata`. If `\raggedbottom` is in force, footnotes get attached to the main galley at a distance of `\footskip` on all pages except on those that are ended by `\newpage` or `\clearpage` where the `\vfil` from `\newpage` pushes the footnotes to the very bottom.

This is kind of a weird difference to a page ending with `\pagebreak`—in that case the page is also run short, but the footnotes are not pushed to the bottom.

In `footmisc` `\@outputbox@removebskip` is only applied when `footmisc` is called with an option specifying the footnote placement, i.e., not in the default case. In new documents we apply it always.

```

563 \def\@outputbox@removebskip{%

```

In some special circumstances the `\@outputbox` might be void. As `\@outputbox@append` below would change this, we handle this case by leaving it unchanged, because otherwise we can no longer detect that situation (needed, for example, in `ftnright`).

```

564     \ifvoid \@outputbox
565         \global\let\@outputbox@reinsertbskip\relax
566     \else

```

If it is not void then we drop the final skip at the end of `\@outputbox` provided there is one and it has a glue stretch order of 1 or more (i.e., contains a `fil` or `fill` part).

```

567     \@outputbox@append{%
568         \@tempskipa\lastskip
569         \ifnum \gluestretchorder\@tempskipa>\z@
570             \unskip

```

We also prepare for reinserting the skip we removed elsewhere on the page. As we have to do this globally, we also need to explicitly reset `\@outputbox@reinsertbskip` if we don't find any such glue.

\@outputbox@reinsertbskip

```

571         \xdef\@outputbox@reinsertbskip
572         {%

```

Everything is done inside `\@outputbox@append` so that a call to `\@outputbox@reinsertbskip` will update the box. If there are no footnotes and no bottom floats that still needs to be attached we first backup by the depth of the `\@outputbox` so that the skip we add starts from the baseline of the last line inside (if there is one). As it is a `fil`, `fill`, or `filll` glue this normally makes no differences, but if the box itself contains a fill skip of higher order our fill is ignored and text is pushed to the bottom of the page. In that case and without backing up first, the last line would not be with its baseline on the bottom, but slightly too high if it has characters with descenders.

```

573         \noexpand\@outputbox@append{%
574         \noexpand\@if@footnotes@TF
575         {}%
576         {%
577         \noexpand\@if@bottomfloats@TF
578         {}%
579         {\noexpand\@backup@outputbox@depth}%
580         }%
581         \vskip\the\@tempskipa}}%
582     \else
583     \global\let\@outputbox@reinsertbskip\relax
584     \fi
585 }%
586 \fi
587 }

```

We need a trivial top-level definition for `\@outputbox@reinsertbskip` in case the first page has no bottom glue and the command gets called.

```

588 \let \@outputbox@reinsertbskip \relax

```

(End of definition for \@outputbox@removebskip and \@outputbox@reinsertbskip.)

`\@backup@outputbox@depth`

In some places we have to make the depth of the `\@outputbox` zero by backing up by its current depth. This is done by `\@backup@outputbox@depth`. The macro can only be used inside of `\@outputbox@append` (or after you have manually set the `\@outputbox@depth` to the right value).

If the depth is zero, there is no point in doing anything, and if the depth is negative it must have been deliberately set to this value, so in this case we also do not back up and change it.

If it is positive we back up by the minimum of `\@outputbox@depth` and `\maxdepth`, i.e., if the depth is unusually large we do not perform a full backup.

```

589 \def\@backup@outputbox@depth{%
590 % \typeout{--> \@backup@outputbox@depth : \the \@outputbox@depth}%
591 \ifdim\@outputbox@depth>\z@
592 \vskip -%
593 \ifdim\@outputbox@depth>\maxdepth
594 \maxdepth
595 \else
596 \@outputbox@depth
597 \fi

```

After the skip was added the depth of the box will become zero but this is not immediately reflected in `\@outputbox@depth`. Thus, if there is further code in the same `\@outputbox@append` testing the register it would make a wrong conclusion. We therefore set it locally to zero.


```

598     \@outputbox@depth\z@
599     \fi
600 }

```

(End of definition for \@backup@outputbox@depth.)

2.2 The output routine configuration components

Here we provide the commands that are used to define code for the socket `build/column/outputbox`. The all manipulate the `\@outputbox` box in one way or another.

\@outputbox@append This general purpose command alters the `\@outputbox` box by appending material to it. As this is a box typesetting operation we make sure that the last line of the box reflects the true depth of the last line (in case that is needed later). We also expose the current depth of `\@outputbox` as `\@outputbox@depth` before unboxing so that its value can be used by `#1` if wanted.

```

601 \def\@outputbox@append #1{%
602     \setbox\@outputbox \vbox {%

```

This `\boxmaxdepth` setting is to ensure that deep footnotes do not overwrite the footer (on account of the negative skip added later): it should use `\@maxdepth` otherwise the change is pointless when there are footnotes.

But see also its use when combining floats.

```

603     \boxmaxdepth \@maxdepth
604     \@outputbox@depth\dp\@outputbox      % if needed in #1
605     \unvbox \@outputbox
606     #1%
607 }%
608 }

```

(End of definition for \@outputbox@append.)

\@outputbox@appendfil Append a `\vfil` to the `\@outputbox`. Depending on the plug in the `build/column/baselineattach` either from the bottom of the box or the baseline of the last line.

```

609 \def\@outputbox@appendfil{\@outputbox@append
610     {\UseSocket{build/column/baselineattach}\vfil}}

```

(End of definition for \@outputbox@appendfil.)

\@outputbox@appendfootnotes This command appends the footnotes to the `\@outputbox` (if there are any). If not, then it does nothing.

```

611 \def\@outputbox@appendfootnotes {%
612     \ifvoid\footins \else

```

Handling split footnotes need further work so this is for the moment just a dummy that does nothing. It might vanish and get replaced by a socket eventually.

```

613     \@makecol@handlesplitfootnotes

```

The following socket can be used to manipulate the material in `\footins` box, for example, if the footnotes are to be presented all together in a single paragraph. By default it does nothing.

```

614     \UseSocket{build/column/footnotes}%

```

Then the footnotes are appended:

```

615     \@outputbox@append{%

```

First we call a socket which controls whether or not we back up by the depth of the `\@outputbox` so that the skip that separates the footnotes from the text starts at the baseline of the last text line.

```
616      \UseSocket{build/column/baselineattach}%
617      \vskip \skip\footins
```

The next socket adjusts the tagging of the `\footins` box, if necessary.

```
618      \UseTaggingSocket{build/column/footins}%
619      \color@begingroup
620      \normalcolor
621      \footnoterule
```

Support for `pdfcolfoot`; eventually this can go once color is properly supported. The `csname` is constructed in case the command is not defined, i.e., the package not loaded.

```
622      \csname pdfcolfoot@current\endcsname
623      \unvbox \footins
624      \color@endgroup
625  }%
626 \fi
627 }
```

(End of definition for \@outputbox@appendfootnotes.)

`\@makecol@handlesplitfootnotes` For future extensions ...

```
628 \let \@makecol@handlesplitfootnotes \@empty
```

(End of definition for \@makecol@handlesplitfootnotes.)

`\@outputbox@attachfloats`
`\@outputbox@attachtopfloats`
`\@outputbox@attachbottomfloats`

Attaching top and bottom floats can usually be done in one go, but for special layouts we might want more control so we provide also separate commands.

The next command was called `\@combinefloats` in the past.

```
629 \def \@outputbox@attachfloats {%
630     \@outputbox@attachtopfloats
631     \@outputbox@attachbottomfloats
632 }
633 \def \@outputbox@attachtopfloats {%
634     \ifx \@toplist\@empty \else \@cflt \fi
635 }
```

Before we append the bottom floats we run the `build/column/baselineattach` so that the `\textfloatsep` is measured from the baseline if that is desired.

```
636 \def \@outputbox@attachbottomfloats {%
637     \ifx \@botlist\@empty \else
638         \@outputbox@append{\UseSocket{build/column/baselineattach}}%
639         \@cflb
640     \fi
641 }
```

(End of definition for \@outputbox@attachfloats, \@outputbox@attachtopfloats, and \@outputbox@attachbottomfloats.)

The next three conditionals might be useful when setting up running headers and footers so perhaps they should be change to CamelCase names. For now they are internal.

`\@if@flushbottom@TF` Test for `\flushbottom` (currently not used).

```
642 \def\@if@flushbottom@TF{%
643   \ifx\@textbottom\relax
644     \expandafter\@firstoftwo
645   \else
646     \expandafter\@secondoftwo
647   \fi
648 }
```

(End of definition for \@if@flushbottom@TF.)

`\@if@footnotes@TF` Test if footnotes are present on the current page or column and not yet added to the `\@outputbox`.

```
649 \def\@if@footnotes@TF{%
650   \ifvoid\footins
651     \expandafter\@secondoftwo
652   \else
653     \expandafter\@firstoftwo
654   \fi
655 }
```

(End of definition for \@if@footnotes@TF.)

`\@if@bottomfloats@TF` Test if bottom floats are around and not yet added to the `\@outputbox`.

```
656 \def\@if@bottomfloats@TF{%
657   \ifx \@botlist\@empty
658     \expandafter\@secondoftwo
659   \else
660     \expandafter\@firstoftwo
661   \fi
662 }
```

(End of definition for \@if@bottomfloats@TF.)

`build/column/outputbox (socket)` We have one socket that is supposed to augment the `\@outputbox` by attaching floats and footnotes with appropriate spacing.

```
663 \NewSocket{build/column/outputbox}{0}
```

The following plugs are available for this socket:

`space-footnotes-floats (plug)` After the galley text there is a vertical `\vfil` followed by any footnotes followed by the bottom floats, if any.

```
664 \NewSocketPlug {build/column/outputbox}{space-footnotes-floats} {%
665   \@if@footnotes@TF
666     {\@outputbox@appendfil}%
667   {\@if@bottomfloats@TF
668     {\@outputbox@appendfil}%
669     {\@outputbox@reinsertbskip}%
670   }%
671   \@outputbox@appendfootnotes
672   \@outputbox@attachfloats
673 }
```

`footnotes-space-floats` (*plug*) As before but the `\vfil` is between footnotes and floats.

```

674 \NewSocketPlug {build/column/outputbox}{footnotes-space-floats} {%
675   \@outputbox@appendfootnotes
676   \@if@bottomfloats@TF
677     {\@outputbox@appendfil}%
678     {\@outputbox@reinsertbskip}%
679   \@outputbox@attachfloats
680 }

```

`floats-space-footnotes` (*plug*) Floats immediately after the galley text and footnotes at the bottom.

```

681 \NewSocketPlug {build/column/outputbox}{floats-space-footnotes} {%
682   \@outputbox@attachfloats
683   \@if@footnotes@TF
684     {\@outputbox@appendfil}%
685     {\@outputbox@reinsertbskip}%
686   \@outputbox@appendfootnotes
687 }

```

`space-floats-footnotes` (*plug*) Both floats and footnotes are pushed to the bottom with footnotes last.⁶⁴

```

688 \NewSocketPlug {build/column/outputbox}{space-floats-footnotes} {%
689   \@if@bottomfloats@TF
690     {\@outputbox@appendfil}%
691     {\@if@footnotes@TF
692       {\@outputbox@appendfil}%
693       {\@outputbox@reinsertbskip}}%
694   \@outputbox@attachfloats
695   \@outputbox@appendfootnotes
696 }

```

`floats-footnotes` (*plug*) All excess space has to be distributed across the existing glue on the page, e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

```

697 \NewSocketPlug {build/column/outputbox}{floats-footnotes} {%
698   \@outputbox@attachfloats
699   \@outputbox@appendfootnotes

```

We do reinsert the bottom skip from `\newpage` if it was taken out earlier. This is, strictly speaking, not necessary in most cases, but it is a `\vfil` while `\raggedbottom` is only generating `\vspace{0pt plus .0001fil}`, so if you have several `\vfil` on the page before the `\newpage` you would alter the space distribution if one is taken out.

```

700   \@outputbox@reinsertbskip
701 }

```

`footnotes-floats` (*plug*)

```

702 \NewSocketPlug {build/column/outputbox}{footnotes-floats} {%
703   \@outputbox@appendfootnotes
704   \@outputbox@attachfloats
705   \@outputbox@reinsertbskip
706 }

```

⁶⁴There are two more permutations, but neither of them has ever been requested so they aren't set up by default — doing that in a class would be trivial though.

The `footnote-floats` plug implements the layout used by L^AT_EX but with the bottom skip bug corrected. This will be the default when `\DocumentMetadata` is used; it can be overwritten either through `footmisc` or by assigning any of the other plugs (or by coding yet another plug for the socket).

`footnotes-floats-legacy` (*plug*) This implements the 2.09 layout (including its bottom skip bug).

```
707 \NewSocketPlug {build/column/outputbox}{footnotes-floats-legacy} {%
```

In the legacy case we don't really want to take out the bottom skip, but rather than altering `\@outputbox@removebskip` in `\makecol` to do nothing, which we would then have to undo in every other layout, we immediately reinsert the dropped skip again.

In the legacy case there is, however, one other thing that needs doing: if there was a `\vfil` at the bottom of the page (e.g., from `\newpage`) and there are no footnotes, then we need to back up by the depth of the last line in the page box. Otherwise you might end up with incorrect alignment if there is a higher-order fill on the page (e.g., `\vfill`) that pushes the text material to the bottom, since in such cases the depth will not be taken into account by T_EX.

This is done automatically by `\@outputbox@reinsertbskip` as long as there are no footnotes waiting to be placed. If there are footnotes, the operation is not done because by default the `\skip\footins` separation is measured from the bottom of the line not from the baseline. But if the user has asked to change this behavior we should honor it and that is done by putting the experimental socket `build/column/baselineattach` in which performs the backing up if desired. This should happen only if footnotes are actually present, so we test for this. Otherwise, there may be bottom floats and the `\textfloatsep` should be measured from the bottom of the last line and if not then any necessary back up is done by `\@outputbox@reinsertbskip`.

Note that after this code has backed up, the depth of `\@outputbox` will be zero as there is a skip at its bottom. So even if the same code is executed a second time (in `\@outputbox@reinsertbskip` or in `\@outputbox@appendfootnotes`) it will not alter the box further.

```
708 \if@footnotes@TF
709   {\@outputbox@append{%
710     \UseSocket{build/column/baselineattach}}}%
711   }{%
```

If there are no footnotes but there are bottom floats we execute the socket `build/column/baselineattach` so that `\textfloatsep` is measured from the baseline if that is being asked for.

```
712 \if@bottomfloats@TF
713   {\@outputbox@append{%
714     \UseSocket{build/column/baselineattach}}}%
715   }{%
716   }%
717 \@outputbox@reinsertbskip
718 \@outputbox@appendfootnotes
719 \@outputbox@attachfloats
720 }
```

The `footnote-floats-legacy` plug is the default used by L^AT_EX when `\DocumentMetadata` is not used; it can be overwritten either through `footmisc` or by assigning any of the other plugs (or by coding yet another plug for the socket).

```
721 \AssignSocketPlug {build/column/outputbox}{footnotes-floats-legacy}
```

build/column/footnotes (*socket*) The socket allowing the manipulation of `\footins` box (result needs to be moved back in there). Used when footnotes are reformatted into a single paragraph by the `para` option of `footmisc`. By default it does nothing.

```
722 \NewSocket{build/column/footnotes}{0}
```

build/column/baselineattach (*socket*) Socket that allows backing up by the depth of the outputbox (but not more than `\maxdepth` before attaching any footnotes or floats.

```
723 \NewSocket{build/column/baselineattach}{0}
```

on (*plug*)

```
724 \NewSocketPlug {build/column/baselineattach}{on} {%
```

We do nothing if the depth is not positive, and we never back up by more than `\maxdepth`.

```
725 \@backup@outputbox@depth
```

```
726 }
```

off (*plug*) The `off` plug is actually identical to the `noop` plug which is automatically made available; but offering that it under the name `off` makes the interface a bit nicer.

```
727 \NewSocketPlug {build/column/baselineattach}{off}{}%
```

This feature is not turned on by default to preserve the behavior of old documents. With new documents using `\DocumentMetadata` it may become the default.

```
728 \AssignSocketPlug {build/column/baselineattach}{off}
```

build/page/before (*hook*) Hooks at the start and end of `\@outputpage` for use by packages.

build/page/after (*hook*) 729 \NewMirroredHookPair{build/page/before}{build/page/after}

build/page/reset (*hook*) Hook in `\@outputpage` to reset special galley conventions within the output routine. L^AT_EX does a lot of resetting in front of this hook, so by default it is empty. They are not made part of the hook (which would have been possible) to ensure that the kernel resets always come first and can be overwritten without the need to apply hook rules unnecessarily.

```
730 \NewHook {build/page/reset}
```

build/column/before (*hook*) Hooks at the start and end of `\@makecol` for use by packages.

build/column/after (*hook*) 731 \NewMirroredHookPair{build/column/before}{build/column/after}

```
732 </2ekernel | latexrelease>
```

```
733 <latexrelease>\EndIncludeInRelease
```

```
734 <latexrelease>\IncludeInRelease{0000/00/00}%
```

```
735 <latexrelease> {\@makecol}{\@makecol adding hooks/sockets}%
```

```
736 <latexrelease>
```

```
737 <latexrelease>\gdef \@makecol {%
```

```
738 <latexrelease> \ifvoid\footins
```

```
739 <latexrelease> \setbox\@outputbox \box\@cclv
```

```
740 <latexrelease> \else
```

```
741 <latexrelease> \setbox\@outputbox \vbox {%
```

```
742 <latexrelease> \boxmaxdepth \@maxdepth
```

```
743 <latexrelease> \unvbox \@cclv
```

```
744 <latexrelease> \vskip \skip\footins
```

```
745 <latexrelease> \color@begingroup
```

```
746 <latexrelease> \normalcolor
```

```

747 <latexrelease> \footnoterule
748 <latexrelease> \unvbox \footins
749 <latexrelease> \color@endgroup
750 <latexrelease> }%
751 <latexrelease> \fi
752 <latexrelease> \let\@elt\relax
753 <latexrelease> \xdef\@freelist{\@freelist\@midlist}%
754 <latexrelease> \global \let \@midlist \@empty
755 <latexrelease> \@combinefloats
756 <latexrelease> \ifvbox\@kludgeins
757 <latexrelease> \@makespecialcolbox
758 <latexrelease> \else
759 <latexrelease> \setbox\@outputbox \vbox to\@colht {%
760 <latexrelease> \@texttop
761 <latexrelease> \dimen@ \dp\@outputbox
762 <latexrelease> \unvbox \@outputbox
763 <latexrelease> \vskip -\dimen@
764 <latexrelease> \@textbottom
765 <latexrelease> }%
766 <latexrelease> \fi
767 <latexrelease> \global \maxdepth \@maxdepth
768 <latexrelease> }
769 <latexrelease>
770 <latexrelease> \gdef \@makespecialcolbox {%
771 <latexrelease> \setbox\@outputbox \vbox {%
772 <latexrelease> \@texttop
773 <latexrelease> \dimen@ \dp\@outputbox
774 <latexrelease> \unvbox\@outputbox
775 <latexrelease> \vskip-\dimen@
776 <latexrelease> }%
777 <latexrelease> \@tempdima \@colht
778 <latexrelease> \ifdim \wd\@kludgeins>\z@
779 <latexrelease> \advance \@tempdima -\ht\@outputbox
780 <latexrelease> \advance \@tempdima \pageshrink
781 <latexrelease> \setbox\@outputbox \vbox to \@colht {%
782 <latexrelease> \unvbox\@outputbox
783 <latexrelease> \vskip \@tempdima
784 <latexrelease> \@textbottom
785 <latexrelease> }%
786 <latexrelease> \else
787 <latexrelease> \advance \@tempdima -\ht\@kludgeins
788 <latexrelease> \setbox \@outputbox \vbox to \@colht {%
789 <latexrelease> \vbox to \@tempdima {%
790 <latexrelease> \unvbox\@outputbox
791 <latexrelease> \@textbottom}%
792 <latexrelease> \vss}%
793 <latexrelease> \fi
794 <latexrelease> {\setbox \@tempboxa \box \@kludgeins}%
795 <latexrelease> }
796 <latexrelease>
797 <latexrelease> \let \@make@normalcolbox \@undefined
798 <latexrelease> \let \@make@specialcolbox \@undefined
799 <latexrelease> \let \@outputbox@removebskip \@undefined
800 <latexrelease> \let \@outputbox@reinsertbskip \@undefined

```

```

801 <latexrelease>
802 <latexrelease> \let \@outputbox@append \@undefined
803 <latexrelease> \let \@outputbox@appendfootnotes \@undefined
804 <latexrelease> \let \@outputbox@attachfloats \@undefined
805 <latexrelease> \let \@outputbox@attachtopfloats \@undefined
806 <latexrelease> \let \@outputbox@attachbottomfloats \@undefined
807 <latexrelease>

```

If we roll back we also need to restore this definition.

```

808 <latexrelease> \def \@combinefloats {%
809 <latexrelease> \ifx \@toplist\@empty \else \@cflt \fi
810 <latexrelease> \ifx \@botlist\@empty \else \@cflb \fi
811 <latexrelease> }
812 <latexrelease>
813 <latexrelease> \let \@ifflushbottom@TF \@undefined
814 <latexrelease> \let \@iffootnotes@TF \@undefined
815 <latexrelease> \let \@ifbottomfloats@TF \@undefined
816 <latexrelease>
817 <latexrelease>\EndIncludeInRelease
818 <*2ekernel>

```

\@reinserts This is the code which reinserts the inserts. It puts them all in one place; this can make some of them come out on the wrong page. It has been put into a separate macro to expedite experimentation.

```

819 \gdef \@reinserts{%
820 \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
821 \ifvbox\@kludgeins\insert\@kludgeins
822 {\unvbox\@kludgeins}\fi
823 }

```

(End of definition for \@reinserts.)

\@texttop These do nothing as a default.
\@textbottom

```

824 \let \@texttop \relax
825 \let \@textbottom \relax

```

(End of definition for \@texttop and \@textbottom.)

\@resetactivechars RmS 93/09/06: added hook to protect against certain active characters in the output routine. Default checks are for active space and end-of-line.
\@activechar@info

```

826 \def\@activechar@info #1{%
827 \@latex@info@no@line {Active #1 character found while
828 output routine is active
829 \MessageBreak
830 This may be a bug in a package file
831 you are using}%
832 }

```

Do not put any spaces in this next bit!

```

833 \begingroup
834 \obeylines\obeyspaces%
835 \catcode'\'\active%
836 \gdef\@resetactivechars{%
837 \def~M{\@activechar@info{EOL}\space}%
838 \def {\@activechar@info{space}\space}%

```



```

839 \let'\active@math@prime}%
840 \endgroup

```

(End of definition for \@resetactivechars and \@activechar@info.)

\@outputpage The `\color@hbox` hooks here are used to avoid putting just a color special into an otherwise empty box (in a header or footer). These boxes are often set to be completely empty and so adding a special produces a very underfull box message.

There has been extensive tidying up of the old code here; including the removal of a level of grouping.

The setting of `\protect` immediately before the `\shipout` is needed so that protected commands within `\writes` are handled correctly.

Within `shipout`'s vbox it is reset to its default value, `\relax`.

Resetting it to its default value after the `shipout` has been completed (and the contents of the `writes` have been expanded) must be done by use of `\aftergroup`. This is because it must have the value `\relax` before macros coming from other uses of `\aftergroup` within this box are expanded.

Putting this into the `\aftergroup` token list does not affect the definition used in expanding the `\writes` because the `aftergroup` token list is only constructed when popping the save-stack, it is not expanded until after the `shipout` is completed.

Question: should things from an `\aftergroup` within the shipped out box be executed in the environment set up for the `writes`, or after it finishes?

A lot of this code has been in-lined to prevent mis-use of internal commands as hooks.

```

841 </2ekernel>
842 <latexrelease>\IncludeInRelease{2025/06/01}%
843 <latexrelease> {\@outputpage}{Use new mark mechanism}%
844 <*2ekernel|latexrelease>

```

Temp definition to vanish again when something is offered by the backend:

```

845 \protected \def \pdfannot@link@on@@ { \csname pdfannot_link_on:\endcsname }
846 \protected \def \pdfannot@link@off@@ { \csname pdfannot_link_off:\endcsname }
847 \def \@outputpage {%

```

We start with a hook available to packages that want to prepend code to `\@outputpage`.

```

848 \UseHook {build/page/before}%

```

The `\endgroup` is put in by `\aftergroup`.

```

849 \begingroup

```

Now all the set-up stuff has been in-lined for Frank.

First the stuff for the `writes`.

From here on was originally in the command `\@writesetup`.

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

Reset `\language` to the value current at `\begin{document}`. In particular this ensures that a pagebreak in `verbatim` does not prevent hyphenation in the page head.

```

850 \language\document@default@language

```

Now a few more resets for things that may have changed during galley processing. We start with resetting some active character definitions.

```

851 \@resetactivechars

```

Then we deal with catcodes that might got altered by `verbatim` or similar environments and reset them to standard values.

```

852 \catcode'\z@
853 \catcode'\{@ne
854 \catcode'\}\tw@
855 \catcode'\$\thr@@
856 \catcode'\&4\relax
857 \catcode'\^M5\relax
858 \catcode'\#6\relax
859 \catcode'\^7\relax
860 \catcode'\_8\relax
861 \catcode'\ 10\relax
862 \catcode'\^^I10\relax
863 \@makeother\<%
864 \@makeother\>%
865 \@makeother\*%
866 \@makeother\.%
867 \@makeother\-%
868 \@makeother\/%
869 \@makeother\[%
870 \@makeother\]%
871 \@makeother\'%
872 \@makeother\'%
873 \@makeother\"%

874 \@makeother\,%
875 \catcode'\~13\relax
876 \catcode'\%14\relax

```

If a page break happens between the start of a list and its first item then `@newlist` will be true and this will mess up any list that is used in the header or footer of the page. So we have to reset that flag.

```

877 \global\let\@if@newlist\if@newlist
878 \global\@newlistfalse

```

This next hook replaces the following:

```

\let\-\@dischph
\let'\@acci\let'\@accii\let=\@acciii
\let\\\@normalcr
\let\par\@par %% 15 Sep 87 (this was once inside the box)

```

and it does more than they did; in particular it sets:

```

\parindent\z@
\parskip\z@skip
\everypar{}%
\leftskip\z@skip
\rightskip\z@skip
\parfillskip\@flushglue
\lineskip\normallineskip
\baselineskip\normalbaselineskip
\sloppy

879 \@parboxrestore

```

... to here was in the command `\@writsetup`.

Hook to allow adding resets local to the output routine processing, e.g., in `\write` or running headers/footers, for packages that set up special conventions in the main galley that should not leak into the page production just because a page break happens in the middle of such an environment.

```
880 \UseHook {build/page/reset}%
```

The following definition of `\protect` is the one active during the `\write` statement that migrate out of the `\shipout` box, so even though it is immediately changed below (inside `\shipout` it still has to be here!

```
881 \let \protect \noexpand
```

```
882 \shipout \vbox{%
```

Inside the `\shipout` box we have a different setting for `\protect`.

```
883 \set@typeset@protect
```

```
884 \aftergroup \endgroup
```

Once the `\shipout` box ends `\protect` is again `\noexpand` which is correct for any `\write` statements, but once they are processed we want to get back to the following setting:

```
885 \aftergroup \set@typeset@protect
```

Now the setup inside the shipped out box; this should contain all the stuff that could only affect typesetting; other stuff may need to be reset for the writes also.

From here ... was in the command `\@shipoutsetup`.

```
886 \if@specialpage
887 \global \@specialpagefalse
888 \@nameuse {ps@\@specialstyle}%
889 \fi
890 \if@twoside
891 \ifodd\count\z@
892 \let \@thehead \@oddhead
893 \let \@thefoot \@oddfoot
894 \let \@themargin \oddsidemargin
895 \else
896 \let \@thehead \@evenhead
897 \let \@thefoot \@evenfoot
898 \let \@themargin \evensidemargin
899 \fi
900 \fi
```

The rest was always inside the box.

RmS 91/08/15: added this line:

```
901 \reset@font
```

RmS 93/08/06 Added `\lineskiplimit=0pt` to guard against it being nonzero: e.g. by `\offinterlineskip` being in effect.

There are probably lots of other things that may need resetting.

```
902 \normalsize
```

Reset the space factors.

```
903 \normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```

904 \let \label \@gobble@with@sphack@om
905 \let \index \@gobble@with@sphack@som
906 \let \glossary \@gobble@with@sphack@om

907 \baselineskip \z@skip
908 \lineskip \z@skip
909 \lineskiplimit \z@

```

... to here was in the command \@shipoutsetup.

```

910 \@beginndvi
911 \vskip \topmargin
912 \moveright\@themargin \vbox {%
913   \setbox\@tempboxa \vbox to\headheight {%
914     \vfil

```

Tagging socket that receives the header in its second argument to surround the header with appropriate tagging structures (first argument is unused). If tagging is disabled it returns the content of the second argument.

```

915   \pdfannot@link@off@@
916   \UseTaggingSocket{build/page/header}{}%
917   {
918     \color@hbox
919     \normalcolor
920     \hb@xt@ \textwidth {\@thehead }%
921     \color@endbox
922   }
923   \pdfannot@link@on@@
924 }
925 \dp \@tempboxa \z@
926 \box \@tempboxa
927 \vskip \headsep
928 \box \@outputbox
929 \baselineskip \footskip

```

Tagging socket that receives the footer in its second argument to surround the footer with appropriate tagging structures (first argument is unused). If tagging is disabled it returns the content of the second argument.

```

930   \pdfannot@link@off@@
931   \UseTaggingSocket{build/page/footer}{}%
932   {
933     \color@hbox
934     \normalcolor
935     \hb@xt@ \textwidth {\@thefoot }%
936     \color@endbox
937   }
938   \pdfannot@link@on@@
939 }
940 }

```

\endgroup now inserted by \aftergroup

Restore \if@newlist

```

941 \global \let \if@newlist \@@if@newlist
942 \global \@colht \textheight
943 \stepcounter{page}%

```

Another hook for packages that want to append material to \@outputpage.

```

944 \UseHook {build/page/after}%
945 }
946 \<latexrelease>\EndIncludeInRelease

947 \<latexrelease>\IncludeInRelease{2017/04/15}%
948 \<latexrelease> {\@outputpage}{Reset language for hyphenation}%
949 \<latexrelease>\def\@outputpage{%
950 \<latexrelease>\begingroup
951 \<latexrelease> \let \protect \noexpand
952 \<latexrelease> \language\document@default@language
953 \<latexrelease> \@resetactivechars
954 \<latexrelease> \global\let\@if@newlist\if@newlist
955 \<latexrelease> \global\@newlistfalse
956 \<latexrelease> \@parboxrestore
957 \<latexrelease> \shipout \vbox{%
958 \<latexrelease> \set@typeset@protect
959 \<latexrelease> \aftergroup \endgroup
960 \<latexrelease> \aftergroup \set@typeset@protect
961 \<latexrelease> \if@specialpage
962 \<latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
963 \<latexrelease> \fi
964 \<latexrelease> \if@twoside
965 \<latexrelease> \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
966 \<latexrelease> \let\@themargin\oddsidemargin
967 \<latexrelease> \else \let\@thehead\@evenhead
968 \<latexrelease> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
969 \<latexrelease> \fi
970 \<latexrelease> \fi
971 \<latexrelease> \reset@font
972 \<latexrelease> \normalsize
973 \<latexrelease> \normalsfcodes
974 \<latexrelease> \let\label\@gobble
975 \<latexrelease> \let\index\@gobble
976 \<latexrelease> \let\glossary\@gobble
977 \<latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
978 \<latexrelease> \@begindvi
979 \<latexrelease> \vskip \topmargin
980 \<latexrelease> \moveright\@themargin \vbox {%
981 \<latexrelease> \setbox\@tempboxa \vbox to\headheight{%
982 \<latexrelease> \vfil
983 \<latexrelease> \color@hbox
984 \<latexrelease> \normalcolor
985 \<latexrelease> \hb@xt@\textwidth{\@thehead}%
986 \<latexrelease> \color@endbox
987 \<latexrelease> }%
988 \<latexrelease> \dp\@tempboxa \z@
989 \<latexrelease> \box\@tempboxa
990 \<latexrelease> \vskip \headsep
991 \<latexrelease> \box\@outputbox
992 \<latexrelease> \baselineskip \footskip
993 \<latexrelease> \color@hbox
994 \<latexrelease> \normalcolor
995 \<latexrelease> \hb@xt@\textwidth{\@thefoot}%

```

```

996 <latexrelease> \color@endbox
997 <latexrelease> }%
998 <latexrelease> }%
999 <latexrelease> \global\let\if@newlist\@if@newlist
1000 <latexrelease> \global \colht \textheight
1001 <latexrelease> \stepcounter{page}%

```

It is now clear that this does something useful (in the old mark mechanism), thanks to Pieter van Oostrum pointing this out. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```

1002 <latexrelease> \let\firstmark\botmark
1003 <latexrelease> }
1004 </2ekernel | latexrelease>
1005 <latexrelease> \EndIncludeInRelease

1006 <latexrelease> \IncludeInRelease{0000/00/00}%
1007 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
1008 <latexrelease> \def\@outputpage{%
1009 <latexrelease> \begingroup
1010 <latexrelease> \let \protect \noexpand
1011 <latexrelease> \resetactivechars
1012 <latexrelease> \global\let\@if@newlist\if@newlist
1013 <latexrelease> \global\@newlistfalse
1014 <latexrelease> \@parboxrestore
1015 <latexrelease> \shipout \vbox{%
1016 <latexrelease> \set@typeset@protect
1017 <latexrelease> \aftergroup \endgroup
1018 <latexrelease> \aftergroup \set@typeset@protect
1019 <latexrelease> \if@specialpage
1020 <latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
1021 <latexrelease> \fi
1022 <latexrelease> \if@twoside
1023 <latexrelease> \ifodd\count\z@
1024 <latexrelease> \let\@thehead\@oddhead \let\@thefoot\@oddfoot
1025 <latexrelease> \let\@themargin\oddsidemargin
1026 <latexrelease> \else \let\@thehead\@evenhead
1027 <latexrelease> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
1028 <latexrelease> \fi
1029 <latexrelease> \fi
1030 <latexrelease> \reset@font
1031 <latexrelease> \normalsize
1032 <latexrelease> \normalsfcodes
1033 <latexrelease> \let\label\@gobble
1034 <latexrelease> \let\index\@gobble
1035 <latexrelease> \let\glossary\@gobble
1036 <latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
1037 <latexrelease> \@begindvi
1038 <latexrelease> \vskip \topmargin
1039 <latexrelease> \moveright\@themargin \vbox {%
1040 <latexrelease> \setbox\@tempboxa \vbox to\headheight{%
1041 <latexrelease> \vfil
1042 <latexrelease> \color@hbox
1043 <latexrelease> \normalcolor
1044 <latexrelease> \hb@xt@\textwidth{\@thehead}%

```

```

1045 <latexrelease>      \color@endbox
1046 <latexrelease>      }%
1047 <latexrelease>      \dp\@tempboxa \z@
1048 <latexrelease>      \box\@tempboxa
1049 <latexrelease>      \vskip \headsep
1050 <latexrelease>      \box\@outputbox
1051 <latexrelease>      \baselineskip \footskip
1052 <latexrelease>      \color@hbox
1053 <latexrelease>      \normalcolor
1054 <latexrelease>      \hb@xt@\textwidth{\@thefoot}%
1055 <latexrelease>      \color@endbox
1056 <latexrelease>      }%
1057 <latexrelease>      }%
1058 <latexrelease>      \global\let\if@newlist\@if@newlist
1059 <latexrelease>      \global \@colht \textheight
1060 <latexrelease>      \stepcounter{page}%
1061 <latexrelease>      \let\firstmark\botmark
1062 <latexrelease>      }
1063 <latexrelease>\EndIncludeInRelease
1064 <*2ekernel>

```

(End of definition for \@outputpage, \@shipoutsetup, and \@writesetup.)

\@beginndvi This unboxes stuff that must appear before anything else in the .dvi file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```

1065 \def \@beginndvi{%
1066   \unvbox \@beginndvibox
1067   \global\let \@beginndvi \@empty
1068 }

```

(End of definition for \@beginndvi.)

2.2.1 Dealing with floats

\@combinefloats Old name for what is now called \@outputbox@attachfloats; kept to support legacy packages.

```

1069 \let \@combinefloats \@outputbox@attachfloats

```

(End of definition for \@combinefloats.)

\@cflt The \boxmaxdepth setting here was not made local to a box so was dangerous. It is needed only within the box made by \@cflt (and not normally even there), so it has been moved there; this also agrees with the original pseudocode.

\@cflb

```

1070 \def \@cflt{%
1071   \let \@elt \@comflelt
1072   \setbox\@tempboxa \vbox{}%
1073   \@toplist
1074   \setbox\@outputbox \vbox{%
1075     \boxmaxdepth \maxdepth
1076     \unvbox\@tempboxa
1077     \vskip -\floatsep
1078     \topfigrule
1079     \vskip \textfloatsep

```

```

1080             \unvbox\@outputbox
1081         }%
1082     \let\@elt\relax
1083     \xdef\@freelist{\@freelist\@toplist}%
1084     \global\let\@toplist\@empty
1085 }
1086 \def \@cflb {%
1087     \let\@elt\@comflelt
1088     \setbox\@tempboxa \vbox{}%
1089     \@botlist
1090     \setbox\@outputbox \vbox{%
1091         \unvbox\@outputbox
1092         \vskip \textfloatsep
1093         \botfigrule
1094         \unvbox\@tempboxa
1095         \vskip -\floatsep
1096     }%
1097     \let\@elt\relax
1098     \xdef\@freelist{\@freelist\@botlist}%
1099     \global \let \@botlist\@empty
1100 }

```

(End of definition for \@cflt and \@cflb.)

```

\@comflelt
\@comdblfelet
\@combinedblfloats
1101 \def\@comflelt#1{\setbox\@tempboxa
1102     \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
1103 \def\@comdblfelet#1{\setbox\@tempboxa
1104     \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
1105 \def \@combinedblfloats{%
1106     \ifx \@dbltoplist \@empty
1107     \else
1108         \setbox\@tempboxa \vbox{}%
1109         \let \@elt \@comdblfelet
1110         \@dbltoplist
1111         \let \@elt \relax
1112         \xdef \@freelist {\@freelist\@dbltoplist}%
1113         \global\let \@dbltoplist \@empty
1114         \setbox\@outputbox \vbox to\textheight

```

The setting of \boxmaxdepth here has no effect since the \@outputbox should already have depth zero. Even so, it would have no effect on the layout of the page.

```

1115     {%
1116     \unvbox\@tempboxa\vskip-\dblfloatsep

```

Here we need different typesetting if the top float comes from \@topnewpage.

```

1117     \ifnum \@dbltopnum>\m@ne
1118         \dblfigrule
1119     \fi
1120     \vskip \dbltextfloatsep

```

If pdf links are present in the galley and those links get broken across pages they have to end up being on the same level of boxing (even if not actually in the same structure) due to some engine restrictions in pdfTeX and LuaTeX. We therefore unbox \@outputbox

here (which only contains a single `\hbox`) so that this case has the same boxing level as a normal twocolumn page without top floats.

```

1121     \unvbox\@outputbox
1122   }%
1123   \fi
1124 }
1125 \if2ekernel)

```

(End of definition for `\@comflelt`, `\@comdbflelt`, and `\@combinedblfloats`.)

`\@startcolumn` We could combine (most of) these two into `\@startcol <list>`. Note that `\@xstartcol` was only used once (i.e. in `\@startcolumn`); it has therefore been removed. This is not quite as efficient but it now has the same structure as `\@startdblcolumn`.

The empty-list test has been moved to `\@tryfcolumn`.

```

1126 \if2ekernel | fltrace)
1127 \def \@startcolumn {%
1128   \global \@colroom \@colht
1129   \@tryfcolumn \@deferlist
1130   \if@fcolmade
1131 \if*trace)
1132   \fl@trace{PAGE: float \if@twocolumn column \else page \fi
1133             completed}%
1134 \if/trace)
1135   \else
1136   \begingroup
1137     \let \reserved@b \@deferlist
1138     \global \let \@deferlist \@empty
1139     \let \@elt \@scolelt
1140     \reserved@b
1141   \endgroup
1142   \fi
1143 }

```

This one does not need to set `\@colht`.

```

1144 \if2ekernel | fltrace)
1145 \if*latexrelease | fltrace)\IncludeInRelease{2015/01/01}%
1146 \if*latexrelease | fltrace) {\@startdblcolumn}{float order in 2-column}%
1147 \if2ekernel | latexrelease | fltrace)
1148 \def \@startdblcolumn {%
1149   \@tryfcolumn \@deferlist
1150   \if@fcolmade
1151 \if*fltrace) \fl@trace{PAGE: double float page completed}%
1152 \if/else)
1153   \begingroup
1154     \let \reserved@b \@deferlist
1155     \global \let \@deferlist \@empty
1156     \let \@elt \@sdblcolelt
1157     \reserved@b
1158   \endgroup
1159   \fi
1160 }%
1161 \if2ekernel | latexrelease | fltrace)
1162 \if*latexrelease | fltrace)\EndIncludeInRelease

```

```

1163 <latexrelease|fltrace>\IncludeInRelease{0000/00/00}%
1164 <latexrelease|fltrace> {\@startdblcolumn}{float order in 2-column}%
1165 <latexrelease|fltrace>\def \@startdblcolumn {%

Not needed since this always comes after \@outputpage:

1166 <latexrelease|fltrace>% \global \@colht \textheight
1167 <latexrelease|fltrace> \@tryfcolumn \@dbldeferlist
1168 <latexrelease|fltrace> \if@fcolmade
1169 <*trace>
1170 <latexrelease|fltrace> \fl@trace{PAGE: double float page completed}%
1171 </trace>
1172 <latexrelease|fltrace> \else

1173 <latexrelease|fltrace> \begingroup
1174 <latexrelease|fltrace> \let \reserved@b \@dbldeferlist
1175 <latexrelease|fltrace> \global \let \@dbldeferlist \@empty
1176 <latexrelease|fltrace> \let \@elt \@sdblcolelt
1177 <latexrelease|fltrace> \reserved@b
1178 <latexrelease|fltrace> \endgroup
1179 <latexrelease|fltrace> \fi
1180 <latexrelease|fltrace>}%
1181 <latexrelease|fltrace>\EndIncludeInRelease
1182 <*2ekernel|fltrace>

```

(End of definition for \@startcolumn and \@startdblcolumn.)

\@tryfcolumn Now tests if its list is empty before any further exertion.

```

1183 \def \@tryfcolumn #1{%
1184   \global \@fcolmadefalse
1185   \ifx #1\@empty
1186   \else
1187   <*trace>
1188     \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
1189               ---\string #1}%
1190     \fl@trace{----- \string #1: #1}%
1191   </trace>

1192   \xdef\@trylist{#1}%
1193   \global \let \@failedlist \@empty
1194   \begingroup
1195     \let \@elt \@xtryfc \@trylist
1196   \endgroup
1197   \if@fcolmade
1198     \@vtryfc #1%
1199   \fi
1200   \fi
1201 }
1202 </2ekernel|fltrace>

```

(End of definition for \@tryfcolumn.)

```

1203 <*2ekernel>

```

\@scolelt

```

1204 \def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}

```

```

(End of definition for \@scolelt.)

\@sdbblelt
1205 \def\@sdbblelt#1{\def\@currbox{#1}\@addtodblelt}
(End of definition for \@sdbblelt.)

\@vtryfc
1206 \def\@vtryfc #1{%
1207   \global\setbox\@outputbox\vbox{}%
1208   \let\@elt\@wtryfc
1209   \@flsucceed
1210   \global\setbox\@outputbox \vbox to\@colht{%
1211     \vskip \@fptop
1212     \vskip -\@fpsep
1213     \unvbox \@outputbox
1214     \vskip \@fpbot}%
1215   \let\@elt\relax
1216   \xdef #1{\@failedlist\@flfail}%
1217   \xdef\@freelist{\@freelist\@flsucceed}}
(End of definition for \@vtryfc.)

\@wtryfc
1218 \def\@wtryfc #1{%
1219   \global\setbox\@outputbox\vbox{%
1220     \unvbox\@outputbox
1221     \vskip\@fpsep
1222     \box #1}}
(End of definition for \@wtryfc.)

\@xtryfc
1223 </2ekernel>
1224 <latexrelease>\IncludeInRelease{2015/01/01}{\@xtryfc}%
1225 <latexrelease> {float order in 2-column}%
1226 <*2ekernel | latexrelease>
1227 \def\@xtryfc #1{%
1228   \@next\reserved@a\@trylist{}{}%
1229   \@currtype \count #1%
1230   \divide\@currtype\@xxxii
1231   \multiply\@currtype\@xxxii
1232   \@bitor \@currtype \@failedlist
1233   \@testfp #1%
1234   \@testwrongwidth #1%
1235   \ifdim \ht #1>\@colht
1236     \@testtrue
1237   \fi
1238   \if@test
1239     \@cons\@failedlist #1%
1240   \else
1241     \@ytryfc #1%
1242     \fi}%
1243 </2ekernel | latexrelease>

```

```

1244 <latexrelease>\EndIncludeInRelease
1245 <latexrelease>\IncludeInRelease{0000/00/00}{\@xtryfc}%
1246 <latexrelease>                                     {float order in 2-column}%
1247 <latexrelease>\def\@xtryfc #1{%
1248 <latexrelease>  \@next\reserved@a\@trylist{}{}%
1249 <latexrelease>  \@currtype \count #1%
1250 <latexrelease>  \divide\@currtype\@xxxii
1251 <latexrelease>  \multiply\@currtype\@xxxii
1252 <latexrelease>  \@bitor \@currtype \@failedlist
1253 <latexrelease>  \@testfp #1%
1254 <latexrelease>  \ifdim \ht #1>\@colht
1255 <latexrelease>    \testtrue
1256 <latexrelease>  \fi
1257 <latexrelease>  \if@test
1258 <latexrelease>    \@cons\@failedlist #1%
1259 <latexrelease>  \else
1260 <latexrelease>    \@ytryfc #1%
1261 <latexrelease>  \fi}%
1262 <latexrelease>\EndIncludeInRelease
1263 <*2ekernel>

```

(End of definition for \@xtryfc.)

\@ytryfc

```

1264 \def\@ytryfc #1{%
1265   \begingroup
1266   \gdef\@flsucceed{\@elt #1}%
1267   \global\let\@flfail\@empty
1268   \@tempdima\ht #1%
1269   \let\@elt\@ztryfc
1270   \@trylist
1271   \ifdim \@tempdima >\@fpmin
1272     \global\@fcolmadetrue
1273   \else
1274     \@cons\@failedlist #1%
1275   \fi
1276 \endgroup
1277 \if@fcolmade
1278   \let\@elt\@gobble
1279 \fi}

```

(End of definition for \@ytryfc.)

\@ztryfc

```

1280 </2ekernel>
1281 <latexrelease>\IncludeInRelease{2015/01/01}{\@ztryfc}%
1282 <latexrelease>                                     {float order in 2-column}%
1283 <*2ekernel | latexrelease>
1284 \def\@ztryfc #1{%
1285   \@tempcnta\count #1%
1286   \divide\@tempcnta\@xxxii
1287   \multiply\@tempcnta\@xxxii
1288   \@bitor \@tempcnta {\@failedlist \@flfail}%
1289   \@testfp #1%

```

```

not in fixfloats?
1290 \testwrongwidth #1%
1291 \@tempdimb\@tempdima
1292 \advance\@tempdimb\ht #1%
1293 \advance\@tempdimb\@fpsep
1294 \ifdim \@tempdimb >\@colht
1295 \testtrue
1296 \fi
1297 \if@test
1298 \@cons\@flfail #1%
1299 \else
1300 \@cons\@flsucceed #1%
1301 \@tempdima\@tempdimb
1302 \fi}%
1303 </2ekernel | latexrelease>
1304 <latexrelease>\EndIncludeInRelease
1305 <latexrelease>\IncludeInRelease{0000/00/00}{\ztryfc}%
1306 <latexrelease> {float order in 2-column}%
1307 <latexrelease>\def\@ztryfc #1{%
1308 <latexrelease> \@tempcnta \count#1%
1309 <latexrelease> \divide\@tempcnta\@xxxii
1310 <latexrelease> \multiply\@tempcnta\@xxxii
1311 <latexrelease> \@bitor \@tempcnta {\@failedlist \@flfail}%
1312 <latexrelease> \testfp #1%
1313 <latexrelease> \@tempdimb\@tempdima
1314 <latexrelease> \advance\@tempdimb \ht#1%
1315 <latexrelease> \advance\@tempdimb\@fpsep
1316 <latexrelease> \ifdim \@tempdimb >\@colht
1317 <latexrelease> \testtrue
1318 <latexrelease> \fi
1319 <latexrelease> \if@test
1320 <latexrelease> \@cons\@flfail #1%
1321 <latexrelease> \else
1322 <latexrelease> \@cons\@flsucceed #1%
1323 <latexrelease> \@tempdima\@tempdimb
1324 <latexrelease> \fi}%
1325 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@ztryfc.)

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

\@addtobot Lots of changes.

```

1326 <*2ekernel | fltrace>
1327 \def \@addtobot {%
1328 <*trace>
1329 \fl@trace{***Start addtobot}%
1330 </trace>
1331 \@getfpsbit 4\relax
1332 <*trace>
1333 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
1334 \the \@fpstype}%
1335 </trace>

```

```

1336 \ifodd \@tempcnta
1337 \@flsetnum \@botnum
1338 \ifnum \@botnum>\z@
1339 \@tempswafalse
1340 \@flcheckspace \@botroom \@botlist
1341 \if@tempswa

1342 % \global \maxdepth \z@
1343 \@flupdates \@botnum \@botroom \@botlist
1344 <*trace>
1345 \fl@trace{colroom (after-bot) = \the \@colroom}%
1346 \fl@trace{colnum (after-bot) = \the \@colnum}%
1347 \fl@trace{botnum (after-bot) = \the \@botnum}%
1348 \fl@trace{***Success: bot}%
1349 </trace>
1350 \inserttrue
1351 \fi
1352 <*trace>
1353 \else
1354 \fl@trace{Fail: botnum = \the \@botnum:
1355 fpstype \the \@fpstype=ORD?}%
1356 \ifnum \@fpstype<\sixt@n
1357 \fl@trace{ERROR: !b float not successful (addtobot)}%
1358 \fi
1359 </trace>
1360 \fi
1361 \fi
1362 }

```

(End of definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```

1363 \def \@addtotoporbot {%
1364 <*trace>
1365 \fl@trace{***Start addtotoporbot}%
1366 </trace>
1367 \@getfpsbit \tw@
1368 <*trace>
1369 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1370 \the \@fpstype}%
1371 </trace>
1372 \ifodd \@tempcnta
1373 \@flsetnum \@topnum
1374 \ifnum \@topnum>\z@
1375 \@tempswafalse
1376 \@flcheckspace \@toproom \@toplist
1377 \if@tempswa
1378 \@bitor\@currtype{\@midlist\@botlist}%
1379 <*trace>
1380 \fl@trace{(mid+bot)list: \@midlist, \@botlist:
1381 (addtotoporbot-before)}%
1382 </trace>
1383 \if@test
1384 <*trace>

```

```

1385         \fl@trace{type already on list: mid or bot---sent to addtobot}}%
1386     \end{trace}
1387     \else
1388         \@flupdates \@topnum \@toproom \@toplist
1389     \end{trace}
1390     \fl@trace{colroom (after-top) = \the \@colroom}%
1391     \fl@trace{colnum (after-top) = \the \@colnum}%
1392     \fl@trace{topnum (after-top) = \the \@topnum}%
1393     \fl@trace{***Success: top}%
1394 \end{trace}
1395     \@inserttrue
1396 \fi
1397 \fi
1398 \end{trace}
1399     \else
1400         \fl@trace{Fail: topnum = \the \@topnum: fpstype
1401                                     \the \@fpstype=ORD?}%
1402         \ifnum \@fpstype<\sixt@n
1403             \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
1404         \fi
1405     \end{trace}
1406 \fi
1407 \fi
1408 \if@insert
1409     \else
1410 \end{trace}
1411         \fl@trace{sent to addtobot (addtotoporbot)}%
1412 \end{trace}
1413         \@addtobot
1414     \fi
1415 }
1416 \end{2ekernel} \end{fltrace}

```

(End of definition for \@addtotoporbot.)

\@addtocurcol Lots of changes.

```

1417 \<latexrelease | fltrace | flafter>\IncludeInRelease{2025/06/01}%
1418 \<latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1419 \<*2ekernel | latexrelease | fltrace | flafter>
1420 \def \@addtocurcol {%
1421 \end{trace}
1422     \fl@trace{***Start addtocurcol}%
1423 \end{trace}
1424     \@insertfalse
1425     \@setfloattypecounts
1426     \ifnum \@fpstype=8
1427 \end{trace}
1428         \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1429 \end{trace}
1430     \else
1431         \ifnum \@fpstype=24
1432 \end{trace}
1433         \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
1434 \end{trace}

```

```

1435     \else
1436     \@flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that `\@reqcolroom` will include the whole of the page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```

1437 <*trace>
1438     \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
1439 </trace>
1440     \advance \@textmin \@textfloatsheight
1441     \@reqcolroom \@pageht

```

This line must be removed since `\@specialoutput` changed.

```

1442 %     \advance \@reqcolroom \@pagedp
1443 <*trace>
1444     \fl@trace{textmin + textfloatsheight: \the \@textmin}%
1445     \fl@trace{page-so-far: \the \@reqcolroom}%
1446 </trace>
1447     \ifdim \@textmin>\@reqcolroom
1448     \@reqcolroom \@textmin
1449 <*trace>
1450     \fl@trace{ORD? textmin being used}%
1451 </trace>
1452     \fi
1453     \advance \@reqcolroom \ht\@currbox

```

We save the current value of `\@reqcolroom` so that we can return to this value later, in case a test fails that may have added something to it.

```

1454     \saved@reqcolroom \@reqcolroom
1455 <*trace>
1456     \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
1457     \fl@trace{colroom = \the \@colroom (addtocurcol)}%
1458     \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
1459 </trace>
1460     \ifdim \@colroom>\@reqcolroom
1461     \@flsetnum \@colnum
1462     \ifnum \@colnum>\z@
1463     \@bitor\@currtype\@deferlist

```

We need to defer the float also if its width doesn't fit.

```

1464     \@testwrongwidth\@currbox
1465 <*trace>
1466     \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
1467 </trace>
1468     \if@test
1469 <*trace>
1470     \fl@trace{type already on list: defer (addtocurcol)}%
1471 </trace>
1472     \else
1473     \@bitor\@currtype\@botlist
1474 <*trace>
1475     \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
1476 </trace>
1477     \if@test

```



```

1478 <*trace>
1479         \fl@trace{type already on list: bot---sent to addtobot}%
1480 </trace>
1481         \@addtobot
1482     \else
1483 <*trace>
1484         \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
1485             here: \the \@fpstype}%
1486 </trace>
1487         \ifodd \count\@currbox
1488             \advance \@reqcolroom \intextsep
1489             \ifdim \@colroom>\@reqcolroom
1490                 \global \advance \@colnum \m@ne
1491                 \global \advance \@textfloatsheight \ht\@currbox

```

This may sometimes give an overestimate.

```

1492         \global \advance \@textfloatsheight 2\intextsep
1493         \@cons \@midlist \@currbox
1494 <*trace>
1495         \fl@trace{***Success: here}%
1496         \fl@trace{textfloatsheight (after-here) =
1497             \the \@textfloatsheight}%
1498         \fl@trace{colnum (after-here) = \the \@colnum}%
1499 </trace>

```

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1500         \if@nobreak
1501             \nobreak
1502             \@nobreakfalse
1503             \everypar{}%
1504         \else
1505             \addpenalty \interlinepenalty
1506         \fi
1507         \vskip \intextsep
1508         \box\@currbox
1509         \penalty\interlinepenalty
1510         \vskip\intextsep
1511         \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi

```

Typesetting ends here.

```

1512         \outputpenalty \z@
1513         \@inserttrue
1514 <*trace>

```

```

1515             \else
1516             \fl@trace{Fail---no room at 2nd test of colroom
1517                     (addtocurcol \string\intextsep)}%
1518     \end{trace}
1519     \fi
1520     \fi
1521     \if@insert
1522     \else

```

Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside the kernel and the fltrace package and \@addtobot shows up in the flafter package. Guess that could have been done a bit more obvious :-)

```

1523 <*2ekernel | fltrace | latexrelease>
1524 <*trace>
1525             \fl@trace{not here: sent to addtotoporbot}%
1526 \end{trace}
1527             \@addtotoporbot
1528 </2ekernel | fltrace | latexrelease>
1529 <!*2ekernel&!fltrace&!latexrelease>
1530 <*trace>
1531             \fl@trace{not here: sent to addtobot}%
1532 \end{trace}
1533             \@addtobot
1534 </!*2ekernel&!fltrace&!latexrelease>
1535             \fi
1536             \fi
1537             \fi
1538 <*trace>
1539     \else
1540     \fl@trace{Fail: colnum = \the \@colnum:
1541             fpstype \the \@fpstype=ORD?}%
1542     \ifnum \@fpstype<\sist@n
1543     \fl@trace{ERROR: BANG float not successful (addtocurcol)}%
1544     \fi
1545 \end{trace}
1546     \fi
1547 <*trace>
1548     \else
1549     \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1550             (addtocurcol)}%
1551 \end{trace}
1552     \fi
1553     \fi
1554     \fi
1555     \if@insert
1556     \else
1557     \@resetfhps
1558 <*trace>
1559     \fl@trace{put on deferlist (addtocurcol)}%
1560 \end{trace}
1561     \@cons\@deferlist\@currbox
1562 <*trace>
1563     \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
1564 \end{trace}

```

```

1565 \fi
1566 }%
1567 </2ekernel | latexrelease | fltrace | flafter>
1568 <latexrelease | fltrace | flafter>\EndIncludeInRelease

The rollback code is rather complicated as it has to account for the fact that we roll back
not just the kernel but also the packages flafter and fltrace.

1569 <latexrelease | fltrace | flafter>\IncludeInRelease{2015/01/01}%
1570 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1571 <latexrelease | fltrace | flafter>\def \@addtocurcol {%
1572 *trace>
1573 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1574 </trace>
1575 <latexrelease | fltrace | flafter> \@insertfalse
1576 <latexrelease | fltrace | flafter> \@setfloattypecounts
1577 <latexrelease | fltrace | flafter> \ifnum \@fpstype=8
1578 *trace>
1579 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1580 </trace>
1581 <latexrelease | fltrace | flafter> \else
1582 <latexrelease | fltrace | flafter> \ifnum \@fpstype=24
1583 *trace>
1584 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
1585 </trace>
1586 <latexrelease | fltrace | flafter> \else
1587 <latexrelease | fltrace | flafter> \@flsettextmin
1588 *trace>
1589 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
1590 </trace>
1591 <latexrelease | fltrace | flafter> \advance \@textmin \@textfloatsheight
1592 <latexrelease | fltrace | flafter> \@reqcolroom \@pageht
1593 *trace>
1594 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight: \the \@textmin}%
1595 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \@reqcolroom}%
1596 </trace>
1597 <latexrelease | fltrace | flafter> \ifdim \@textmin>\@reqcolroom
1598 <latexrelease | fltrace | flafter> \@reqcolroom \@textmin
1599 *trace>
1600 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1601 </trace>
1602 <latexrelease | fltrace | flafter> \fi
1603 <latexrelease | fltrace | flafter> \advance \@reqcolroom \ht\@currbox
1604 *trace>
1605 <latexrelease | fltrace | flafter> \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
1606 <latexrelease | fltrace | flafter> \fl@trace{colroom = \the \@colroom (addtocurcol)}%
1607 <latexrelease | fltrace | flafter> \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
1608 </trace>
1609 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1610 <latexrelease | fltrace | flafter> \@flsetnum \@colnum
1611 <latexrelease | fltrace | flafter> \ifnum \@colnum>z@
1612 <latexrelease | fltrace | flafter> \@bitor\@currtype\@deferlist
1613 <latexrelease | fltrace | flafter> \@testwrongwidth\@currbox
1614 *trace>
1615 <latexrelease | fltrace | flafter> \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%

```

```

1616 </trace>
1617 <latexrelease | fltrace | flafter>
1618 <*trace>
1619 <latexrelease | fltrace | flafter>
1620 </trace>
1621 <latexrelease | fltrace | flafter>
1622 <latexrelease | fltrace | flafter>
1623 <*trace>
1624 <latexrelease | fltrace | flafter>
1625 </trace>
1626 <latexrelease | fltrace | flafter>
1627 <*trace>
1628 <latexrelease | fltrace | flafter>
1629 </trace>
1630 <latexrelease | fltrace | flafter>
1631 <latexrelease | fltrace | flafter>
1632 <*trace>
1633 <latexrelease | fltrace | flafter>
1634 <latexrelease | fltrace | flafter>
1635 </trace>
1636 <latexrelease | fltrace | flafter>
1637 <latexrelease | fltrace | flafter>
1638 <latexrelease | fltrace | flafter>
1639 <latexrelease | fltrace | flafter>
1640 <latexrelease | fltrace | flafter>
1641 <latexrelease | fltrace | flafter>
1642 <latexrelease | fltrace | flafter>
1643 <*trace>
1644 <latexrelease | fltrace | flafter>
1645 <latexrelease | fltrace | flafter>
1646 <latexrelease | fltrace | flafter>
1647 <latexrelease | fltrace | flafter>
1648 </trace>
1649 <latexrelease | fltrace | flafter>
1650 <latexrelease | fltrace | flafter>
1651 <latexrelease | fltrace | flafter>
1652 <latexrelease | fltrace | flafter>
1653 <latexrelease | fltrace | flafter>
1654 <latexrelease | fltrace | flafter>
1655 <latexrelease | fltrace | flafter>
1656 <latexrelease | fltrace | flafter>
1657 <latexrelease | fltrace | flafter>
1658 <latexrelease | fltrace | flafter>
1659 <latexrelease | fltrace | flafter>
1660 <latexrelease | fltrace | flafter>
1661 <latexrelease | fltrace | flafter>
1662 <latexrelease | fltrace | flafter>
1663 <*trace>
1664 <latexrelease | fltrace | flafter>
1665 <latexrelease | fltrace | flafter>
1666 <latexrelease | fltrace | flafter>
1667 </trace>
1668 <latexrelease | fltrace | flafter>
1669 <latexrelease | fltrace | flafter>

\if@test
    \fl@trace{type already on list: defer (addtocurcol)}%
\else
    \@bitor\@currtype\@botlist
\fl@trace{botlist: \@botlist: (addtocurcol-before)}%

\if@test
    \fl@trace{type already on list: bot---sent to addtobot}

    \@addtobot
\else
    \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
        here: \the \@fpstype}%

    \ifodd \count\@currbox
        \advance \@reqcolroom \intextsep
        \ifdim \@colroom>\@reqcolroom
            \global \advance \@colnum \m@ne
            \global \advance \@textfloatsheight \ht\@currbox
            \global \advance \@textfloatsheight 2\intextsep
            \@cons \@midlist \@currbox

\fl@trace{***Success: here}%
\fl@trace{textfloatsheight (after-here) =
    \the \@textfloatsheight}%
\fl@trace{colnum (after-here) = \the \@colnum}%

    \if@nobreak
        \nobreak
        \@nobreakfalse
        \everypar{}%
    \else
        \addpenalty \interlinepenalty
    \fi
    \vskip \intextsep
    \box\@currbox
    \penalty\interlinepenalty
    \vskip\intextsep
    \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
    \outputpenalty \z@
    \@inserttrue

\else
    \fl@trace{Fail---no room at 2nd test of colroom
        (addtocurcol \string\intextsep)}%

\fi
\fi

```

```

1670 <latexrelease | fltrace | flafter>          \if@insert
1671 <latexrelease | fltrace | flafter>          \else
1672 <*trace>
1673 <fltrace | latexrelease>                    \fl@trace{not here: sent to addtotoporbot}%
1674 </trace>
1675 <fltrace | latexrelease>                    \@addtotoporbot
1676 <*trace>
1677 <flafter>          \fl@trace{not here: sent to addtobot}%
1678 </trace>
1679 <flafter>          \@addtobot
1680 <latexrelease | fltrace | flafter>          \fi
1681 <latexrelease | fltrace | flafter>          \fi
1682 <latexrelease | fltrace | flafter>          \fi
1683 <*trace>
1684 <latexrelease | fltrace | flafter>          \else
1685 <latexrelease | fltrace | flafter>          \fl@trace{Fail: colnum = \the \@colnum:
1686 <latexrelease | fltrace | flafter>          fpstype \the \@fpstype=ORD?}%
1687 <latexrelease | fltrace | flafter>          \ifnum \@fpstype<\sist@n
1688 <latexrelease | fltrace | flafter>          \fl@trace{ERROR: BANG float not successful (addtocurcol)}
1689 <latexrelease | fltrace | flafter>          \fi
1690 </trace>
1691 <latexrelease | fltrace | flafter>          \fi
1692 <*trace>
1693 <latexrelease | fltrace | flafter>          \else
1694 <latexrelease | fltrace | flafter>          \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1695 <latexrelease | fltrace | flafter>          (addtocurcol)}}%
1696 </trace>
1697 <latexrelease | fltrace | flafter>          \fi
1698 <latexrelease | fltrace | flafter>          \fi
1699 <latexrelease | fltrace | flafter>          \fi
1700 <latexrelease | fltrace | flafter>          \if@insert
1701 <latexrelease | fltrace | flafter>          \else
1702 <latexrelease | fltrace | flafter>          \@resetfps
1703 <*trace>
1704 <latexrelease | fltrace | flafter>          \fl@trace{put on deferlist (addtocurcol)}}%
1705 </trace>
1706 <latexrelease | fltrace | flafter>          \@cons\@deferlist\@currbox
1707 <*trace>
1708 <latexrelease | fltrace | flafter>          \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}}%
1709 </trace>
1710 <latexrelease | fltrace | flafter>          \fi
1711 <latexrelease | fltrace | flafter>}}%
1712 <latexrelease | fltrace | flafter>\EndIncludeInRelease
1713 <latexrelease | fltrace | flafter>\IncludeInRelease{0000/00/00}%
1714 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1715 <latexrelease | fltrace | flafter>\def \@addtocurcol {%
1716 <*trace>
1717 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1718 </trace>
1719 <latexrelease | fltrace | flafter> \@insertfalse
1720 <latexrelease | fltrace | flafter> \@setfloattyperecounts
1721 <latexrelease | fltrace | flafter> \ifnum \@fpstype=8
1722 <*trace>
1723 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol):

```

```

1724 <latexrelease | fltrace | flafter> \the \@fpstype = 8?}%
1725 </trace>
1726 <latexrelease | fltrace | flafter> \else
1727 <latexrelease | fltrace | flafter> \ifnum \@fpstype=24
1728 <*trace>
1729 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol):
1730 <latexrelease | fltrace | flafter> \the \@fpstype = 24?}%
1731 </trace>
1732 <latexrelease | fltrace | flafter> \else
1733 <latexrelease | fltrace | flafter> \@flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that \@reqcolroom will include the whole of the page-so-far, and hence includes \@textfloatsheight of floats, so before comparing it with \@textmin, we add this to \@textmin also.

```

1734 <*trace>
1735 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) =
1736 <latexrelease | fltrace | flafter> \the \@textfloatsheight}%
1737 </trace>
1738 <latexrelease | fltrace | flafter> \advance \@textmin \@textfloatsheight
1739 <latexrelease | fltrace | flafter> \@reqcolroom \@pageht

```

This line must be removed since \@specialoutput changed.

```

1740 % \advance \@reqcolroom \@pagedp
1741 <*trace>
1742 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight:
1743 <latexrelease | fltrace | flafter> \the \@textmin}%
1744 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \@reqcolroom}%
1745 <latexrelease | fltrace | flafter>
1746 </trace>
1747 <latexrelease | fltrace | flafter> \ifdim \@textmin>\@reqcolroom
1748 <latexrelease | fltrace | flafter> \@reqcolroom \@textmin
1749 <*trace>
1750 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1751 </trace>
1752 <latexrelease | fltrace | flafter> \fi
1753 <latexrelease | fltrace | flafter> \advance \@reqcolroom \ht\@currbox
1754 <*trace>
1755 <latexrelease | fltrace | flafter> \fl@trace{float size =
1756 <latexrelease | fltrace | flafter> \the \ht \@currbox (addtocurcol)}%
1757 <latexrelease | fltrace | flafter> \fl@trace{colroom =
1758 <latexrelease | fltrace | flafter> \the \@colroom (addtocurcol)}%
1759 <latexrelease | fltrace | flafter> \fl@trace{reqcolroom =
1760 <latexrelease | fltrace | flafter> \the \@reqcolroom (addtocurcol)}%
1761 </trace>
1762 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1763 <latexrelease | fltrace | flafter> \@flsetnum \@colnum
1764 <latexrelease | fltrace | flafter> \ifnum \@colnum>\z@
1765 <latexrelease | fltrace | flafter> \@bitor\@currtype\@deferlist
1766 <*trace>
1767 <latexrelease | fltrace | flafter> \fl@trace{deferlist:
1768 <latexrelease | fltrace | flafter> \@deferlist: (addtocurcol-before)}%
1769 </trace>
1770 <latexrelease | fltrace | flafter> \if@test

```

```

1771 <*trace>
1772 <latexrelease | fltrace | flafter>          \fl@trace{type already on list:
1773 <latexrelease | fltrace | flafter>              defer (addtocurcol)}%
1774 </trace>
1775 <latexrelease | fltrace | flafter>          \else
1776 <latexrelease | fltrace | flafter>              \@bitor\@currtype\@botlist
1777 <*trace>
1778 <latexrelease | fltrace | flafter>          \fl@trace{botlist: \@botlist:
1779 <latexrelease | fltrace | flafter>              (addtocurcol-before)}%
1780 </trace>
1781 <latexrelease | fltrace | flafter>          \if@test
1782 <*trace>
1783 <latexrelease | fltrace | flafter>              \fl@trace{type already on list:
1784 <latexrelease | fltrace | flafter>              bot---sent to addtobot}%
1785 </trace>
1786 <latexrelease | fltrace | flafter>          \@addtobot
1787 <latexrelease | fltrace | flafter>          \else
1788 <*trace>
1789 <latexrelease | fltrace | flafter>              \fl@trace{fpstype
1790 <latexrelease | fltrace | flafter>              \ifodd \@tempcnta OK \else not \fi
1791 <latexrelease | fltrace | flafter>              here: \the \@fpstype}%
1792 </trace>
1793 <latexrelease | fltrace | flafter>          \ifodd \count\@currbox
1794 <latexrelease | fltrace | flafter>              \advance \@reqcolroom \intextsep
1795 <latexrelease | fltrace | flafter>              \ifdim \@colroom>\@reqcolroom
1796 <latexrelease | fltrace | flafter>              \global \advance \@colnum \m@ne
1797 <latexrelease | fltrace | flafter>              \global \advance
1798 <latexrelease | fltrace | flafter>              \@textfloatsheight\ht\@currbox

```

This may sometimes give an overestimate.

```

1799 <latexrelease | fltrace | flafter>          \global \advance
1800 <latexrelease | fltrace | flafter>              \@textfloatsheight 2\intextsep
1801 <latexrelease | fltrace | flafter>              \@cons \@midlist \@currbox
1802 <*trace>
1803 <latexrelease | fltrace | flafter>          \fl@trace{***Success: here}%
1804 <latexrelease | fltrace | flafter>          \fl@trace{textfloatsheight
1805 <latexrelease | fltrace | flafter>              (after-here) =
1806 <latexrelease | fltrace | flafter>              \the \@textfloatsheight}%
1807 <latexrelease | fltrace | flafter>          \fl@trace{colnum (after-here) =
1808 <latexrelease | fltrace | flafter>              \the \@colnum}%
1809 </trace>

```

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1810 <latexrelease | fltrace | flafter> \if@nobreak
1811 <latexrelease | fltrace | flafter> \nobreak
1812 <latexrelease | fltrace | flafter> \@nobreakfalse
1813 <latexrelease | fltrace | flafter> \everypar{}%
1814 <latexrelease | fltrace | flafter> \else
1815 <latexrelease | fltrace | flafter> \addpenalty\interlinepenalty
1816 <latexrelease | fltrace | flafter> \fi
1817 <latexrelease | fltrace | flafter> \vskip \intextsep
1818 <latexrelease | fltrace | flafter> \box\@currbox
1819 <latexrelease | fltrace | flafter> \penalty\interlinepenalty
1820 <latexrelease | fltrace | flafter> \vskip\intextsep
1821 <latexrelease | fltrace | flafter> \ifnum\outputpenalty
1822 <latexrelease | fltrace | flafter> <-\@Mii \vskip
1823 <latexrelease | fltrace | flafter> -\parskip\fi

```

Typesetting ends here.

```

1824 <latexrelease | fltrace | flafter> \outputpenalty \z@
1825 <latexrelease | fltrace | flafter> \@inserttrue
1826 <*trace>
1827 <latexrelease | fltrace | flafter> \else
1828 <latexrelease | fltrace | flafter> \fl@trace{Fail---no room at 2nd test of colroom
1829 <latexrelease | fltrace | flafter> (addtocorcol \string\intextsep)}%
1830 </trace>
1831 <latexrelease | fltrace | flafter> \fi
1832 <latexrelease | fltrace | flafter> \fi
1833 <latexrelease | fltrace | flafter> \if@insert
1834 <latexrelease | fltrace | flafter> \else

```

Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside the kernel and the fltrace package and \@addtotoporbot shows up in the flafter package. Guess that could have been done a bit more obvious :-)

```

1835 <*2kernel | fltrace>
1836 <*trace>
1837 <latexrelease | fltrace | flafter> \fl@trace{not here: sent to addtotoporbot}%
1838 </trace>
1839 <latexrelease | fltrace | flafter> \@addtotoporbot
1840 </2kernel | fltrace>
1841 <!*2kernel&!autoload&!fltrace>
1842 <*trace>
1843 <latexrelease | fltrace | flafter> \fl@trace{not here: sent to addtobot}%
1844 </trace>
1845 <latexrelease | fltrace | flafter> \@addtobot
1846 </*!2kernel&!autoload&!fltrace>
1847 <latexrelease | fltrace | flafter> \fi
1848 <latexrelease | fltrace | flafter> \fi
1849 <latexrelease | fltrace | flafter> \fi
1850 <*trace>
1851 <latexrelease | fltrace | flafter> \else
1852 <latexrelease | fltrace | flafter> \fl@trace{Fail: colnum = \the \@colnum:
1853 <latexrelease | fltrace | flafter> fpstype \the \@fpstype=ORD?}%
1854 <latexrelease | fltrace | flafter> \ifnum \@fpstype<\sist@@n
1855 <latexrelease | fltrace | flafter> \fl@trace{ERROR: BANG float not successful
1856 <latexrelease | fltrace | flafter> (addtocurcol)}%
1857 <latexrelease | fltrace | flafter> \fi
1858 </trace>

```



```

1859 <latexrelease | fltrace | flafter>          \fi
1860 <*trace>
1861 <latexrelease | fltrace | flafter>          \else
1862 <latexrelease | fltrace | flafter>          \fl@trace{Fail---no room: fl box ht:
1863 <latexrelease | fltrace | flafter>          \the \ht \@currbox (addtocurcol)}%
1864 </trace>
1865 <latexrelease | fltrace | flafter>          \fi
1866 <latexrelease | fltrace | flafter>          \fi
1867 <latexrelease | fltrace | flafter>          \fi
1868 <latexrelease | fltrace | flafter>          \if@insert
1869 <latexrelease | fltrace | flafter>          \else
1870 <latexrelease | fltrace | flafter>          \@resetfps
1871 <*trace>
1872 <latexrelease | fltrace | flafter>          \fl@trace{put on deferlist (addtocurcol)}%
1873 </trace>
1874 <latexrelease | fltrace | flafter>          \@cons\@deferlist\@currbox
1875 <*trace>
1876 <latexrelease | fltrace | flafter>          \fl@trace{deferlist: \@deferlist:
1877 <latexrelease | fltrace | flafter>          (addtocurcol-after)}%
1878 </trace>
1879 <latexrelease | fltrace | flafter>          \fi
1880 <latexrelease | fltrace | flafter>          }%
1881 <latexrelease | fltrace | flafter>\EndIncludeInRelease

```

(End of definition for \@addtocurcol.)

\@addtonextcol Lots of changes.

```

1882 <latexrelease | fltrace>\IncludeInRelease{2025/06/01}
1883 <latexrelease | fltrace>  {\@addtonextcol}{float order in 2-column}%
1884 <*2ekernel | latexrelease | fltrace>
1885 \def\@addtonextcol{%
1886   \begingroup
1887   <*trace>
1888   \fl@trace{***Start addtonextcol}%
1889   </trace>
1890   \@insertfalse
1891   \@setfloattypecounts
1892   \ifnum \@fpstype=8
1893   <*trace>
1894   \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
1895   </trace>
1896   \else
1897   \ifnum \@fpstype=24
1898   <*trace>
1899   \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
1900   </trace>
1901   \else
1902   \@flsettextmin
1903   <*trace>
1904   \fl@trace{text-so-far: Opt (top of col)}%
1905   </trace>
1906   \@reqcolroom \ht\@currbox
1907   <*trace>
1908   \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%

```

```

1909 </trace>
1910     \advance \@reqcolroom \@textmin
1911     \saved@reqcolroom \@reqcolroom
1912 <*trace>
1913     \fl@trace{colroom = \the \@colroom (addtonextcol)}%
1914     \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
1915 </trace>
1916     \ifdim \@colroom>\@reqcolroom
1917         \@flsetnum \@colnum
1918         \ifnum \@colnum>\z@
1919             \@bitor \@currtype \@deferlist
1920 <*trace>
1921         \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
1922 </trace>
1923         \@testwrongwidth \@currbox
1924         \if@test
1925 <*trace>
1926             \fl@trace{type already on list: defer (addtonextcol)}%
1927 </trace>
1928         \else
1929 <*trace>
1930             \fl@trace{sent to addtotoporbot (addtonextcol)}%
1931 </trace>
1932             \@addtotoporbot
1933         \fi
1934     \fi
1935 <*trace>
1936     \else
1937         \fl@trace{Fail---no room: fl box ht: \the \@currbox
1938             (addtonextcol)}%
1939 </trace>
1940     \fi
1941     \fi
1942     \fi
1943     \if@insert
1944     \else
1945 <*trace>
1946     \fl@trace{put back on deferlist (addtonextcol)}%
1947 </trace>
1948     \@cons \@deferlist \@currbox
1949 <*trace>
1950     \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1951 </trace>
1952     \fi
1953 <*trace>
1954     \fl@trace{End of addtonextcol -- locally counts:}%
1955     \fl@trace{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1956 </trace>
1957     \endgroup
1958 <*trace>
1959     \fl@trace{End of addtonextcol -- globally counts:}%
1960     \fl@trace{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1961 </trace>

```

```

1962 }%
1963 </2ekernel | latexrelease | fltrace>
1964 <latexrelease | fltrace>\EndIncludeInRelease
1965 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}
1966 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1967 <latexrelease | fltrace>\def\@addtonextcol{%
1968 <latexrelease | fltrace> \begingroup
1969 <*trace>
1970 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1971 </trace>
1972 <latexrelease | fltrace> \@insertfalse
1973 <latexrelease | fltrace> \@setfloattyperecounts
1974 <latexrelease | fltrace> \ifnum \@fpstype=8
1975 <*trace>
1976 <latexrelease | fltrace> \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
1977 </trace>
1978 <latexrelease | fltrace> \else
1979 <latexrelease | fltrace> \ifnum \@fpstype=24
1980 <*trace>
1981 <latexrelease | fltrace> \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
1982 </trace>
1983 <latexrelease | fltrace> \else
1984 <latexrelease | fltrace> \@flsettextmin
1985 <*trace>
1986 <latexrelease | fltrace> \fl@trace{text-so-far: Opt (top of col)}%
1987 </trace>
1988 <latexrelease | fltrace> \@reqcolroom \ht\@currbox
1989 <*trace>
1990 <latexrelease | fltrace> \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
1991 </trace>
1992 <latexrelease | fltrace> \advance \@reqcolroom \@textmin
1993 <*trace>
1994 <latexrelease | fltrace> \fl@trace{colroom = \the \@colroom (addtonextcol)}%
1995 <latexrelease | fltrace> \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
1996 </trace>
1997 <latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
1998 <latexrelease | fltrace> \@flsetnum \@colnum
1999 <latexrelease | fltrace> \ifnum\@colnum>\z@
2000 <latexrelease | fltrace> \@bitor\@currtype\@deferlist
2001 <*trace>
2002 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
2003 </trace>
2004 <latexrelease | fltrace> \@testwrongwidth\@currbox
2005 <latexrelease | fltrace> \if@test
2006 <*trace>
2007 <latexrelease | fltrace> \fl@trace{type already on list: defer (addtonextcol)}%
2008 </trace>
2009 <latexrelease | fltrace> \else
2010 <*trace>
2011 <latexrelease | fltrace> \fl@trace{sent to addtotoporbot (addtonextcol)}%
2012 </trace>
2013 <latexrelease | fltrace> \@addtotoporbot
2014 <latexrelease | fltrace> \fi
2015 <latexrelease | fltrace> \fi

```

```

2016 <*trace>
2017 <latexrelease | fltrace>          \else
2018 <latexrelease | fltrace>          \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
2019 <latexrelease | fltrace>                                     (addtonextcol)}%
2020 </trace>
2021 <latexrelease | fltrace>          \fi
2022 <latexrelease | fltrace>          \fi
2023 <latexrelease | fltrace>          \fi
2024 <latexrelease | fltrace>          \if@insert
2025 <latexrelease | fltrace>          \else
2026 <*trace>
2027 <latexrelease | fltrace>          \fl@trace{put back on deferlist (addtonextcol)}%
2028 </trace>
2029 <latexrelease | fltrace>          \@cons\@deferlist\@currbox
2030 <*trace>
2031 <latexrelease | fltrace>          \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
2032 </trace>
2033 <latexrelease | fltrace>          \fi
2034 <*trace>
2035 <latexrelease | fltrace>          \fl@trace{End of addtonextcol -- locally counts:}%
2036 <latexrelease | fltrace>          \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
2037 </trace>
2038 <latexrelease | fltrace>          \endgroup
2039 <*trace>
2040 <latexrelease | fltrace>          \fl@trace{End of addtonextcol -- globally counts:}%
2041 <latexrelease | fltrace>          \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
2042 </trace>
2043 <latexrelease | fltrace>}%
2044 <latexrelease | fltrace>\EndIncludeInRelease

2045 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2046 <latexrelease | fltrace>  {\@addtonextcol}{float order in 2-column}%
2047 <latexrelease | fltrace>\def\@addtonextcol{%
2048 <latexrelease | fltrace>  \begingroup
2049 <*trace>
2050 <latexrelease | fltrace>    \fl@trace{***Start addtonextcol}%
2051 </trace>
2052 <latexrelease | fltrace>    \@insertfalse
2053 <latexrelease | fltrace>    \@setfloattypecounts
2054 <latexrelease | fltrace>    \ifnum \@fpstype=8
2055 <*trace>
2056 <latexrelease | fltrace>        \fl@trace{fpstype not curcol:
2057 <latexrelease | fltrace>                                \the \@fpstype = 8?}%
2058 </trace>
2059 <latexrelease | fltrace>    \else
2060 <latexrelease | fltrace>        \ifnum \@fpstype=24
2061 <*trace>
2062 <latexrelease | fltrace>        \fl@trace{fpstype not curcol:
2063 <latexrelease | fltrace>                                \the \@fpstype = 24?}%
2064 </trace>
2065 <latexrelease | fltrace>    \else
2066 <latexrelease | fltrace>        \@flsettextmin
2067 <*trace>
2068 <latexrelease | fltrace>        \fl@trace{text-so-far: 0pt (top of col)}%
2069 </trace>

```

```

2070 <latexrelease | fltrace> \@reqcolroom \ht\@currbox
2071 <*trace>
2072 <latexrelease | fltrace> \fl@trace{float size:
2073 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%
2074 <latexrelease | fltrace>
2075 </trace>
2076 <latexrelease | fltrace> \advance \@reqcolroom \@textmin
2077 <*trace>
2078 <latexrelease | fltrace> \fl@trace{colroom =
2079 <latexrelease | fltrace> \the \@colroom (addtonextcol)}%
2080 <latexrelease | fltrace> \fl@trace{reqcolroom =
2081 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%
2082 </trace>
2083 <latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
2084 <latexrelease | fltrace> \@flsetnum \@colnum
2085 <latexrelease | fltrace> \ifnum\@colnum>\z@
2086 <latexrelease | fltrace> \@bitor\@currtype\@deferlist
2087 <*trace>
2088 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
2089 <latexrelease | fltrace> (addtonextcol-before)}%
2090 </trace>
2091 <latexrelease | fltrace> \if@test
2092 <*trace>
2093 <latexrelease | fltrace> \fl@trace{type already on list:
2094 <latexrelease | fltrace> defer (addtonextcol)}%
2095 </trace>
2096 <latexrelease | fltrace> \else
2097 <*trace>
2098 <latexrelease | fltrace> \fl@trace{sent to addtotoporbot
2099 <latexrelease | fltrace> (addtonextcol)}%
2100 </trace>
2101 <latexrelease | fltrace> \@addtotoporbot
2102 <latexrelease | fltrace> \fi
2103 <latexrelease | fltrace> \fi
2104 <*trace>
2105 <latexrelease | fltrace> \else
2106 <latexrelease | fltrace> \fl@trace{Fail---no room: fl box ht:
2107 <latexrelease | fltrace> \the \ht \@currbox (addtonextcol)}%
2108 </trace>
2109 <latexrelease | fltrace> \fi
2110 <latexrelease | fltrace> \fi
2111 <latexrelease | fltrace> \fi
2112 <latexrelease | fltrace> \if@insert
2113 <latexrelease | fltrace> \else
2114 <*trace>
2115 <latexrelease | fltrace> \fl@trace{put back on deferlist
2116 <latexrelease | fltrace> (addtonextcol)}%
2117 </trace>
2118 <latexrelease | fltrace> \@cons\@deferlist\@currbox
2119 <*trace>
2120 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
2121 <latexrelease | fltrace> (addtonextcol-after)}%
2122 </trace>
2123 <latexrelease | fltrace> \fi

```

```

2124 <*trace>
2125 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
2126 <latexrelease | fltrace> locally counts:}%
2127 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.
2128 <latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%
2129 </trace>
2130 <latexrelease | fltrace> \endgroup
2131 <*trace>
2132 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
2133 <latexrelease | fltrace> globally counts:}%
2134 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.
2135 <latexrelease | fltrace> top: \the \@topnum. bot: \the \@botnum.}%
2136 </trace>
2137 <latexrelease | fltrace>}%
2138 <latexrelease | fltrace>\EndIncludeInRelease

```

(End of definition for \@addtonextcol.)

\@addtodblcol Lots of changes.

```

2139 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
2140 <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
2141 <*2kernel | latexrelease | fltrace>
2142 \def\@addtodblcol{%
2143 \begingroup
2144 <*trace>
2145 \fl@trace{***Start addtodblcol}%
2146 </trace>
2147 \insertfalse
2148 \setfloattypecounts
2149 \getfpsbit \tw@
2150 <*trace>
2151 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
2152 \the \@fpstype}%
2153 </trace>
2154 \ifodd\@tempcnta
2155 \flsetnum \@dbltopnum
2156 \ifnum \@dbltopnum>\z@
2157 \tempswafalse
2158 \ifdim \@dbltoproom>\ht\@currbox
2159 \tempswatrue
2160 <*trace>
2161 \fl@trace{Space OK: \@dbltoproom =
2162 \the \@dbltoproom > \the \ht \@currbox
2163 (dbltoproom)}%
2164 </trace>
2165 \else
2166 <*trace>
2167 \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
2168 </trace>
2169 \ifnum \@fpstype<\sist@n
2170 <*trace>
2171 \fl@trace{BANG float ignoring \@dbltoproom}%
2172 \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
2173 Ht float: \the \ht \@currbox-BANG}%

```

```

2174 </trace>

Need to check that there is room on the page, using the local value of \@textmin to make
the necessary adjustment to \@dbltoproom.

2175         \advance \@dbltoproom \@textmin
2176 <*trace>
2177         \fl@trace{Local value of texmin: \the\@textmin}%
2178         \fl@trace{\@spaces space on page = \the \@dbltoproom.
2179                 Ht float: \the \ht \@currbox-BANG}%
2180 </trace>
2181         \ifdim \@dbltoproom>\ht\@currbox
2182         \@tempwattrue
2183 <*trace>
2184         \fl@trace{Space OK BANG: space on page =
2185                 \the \@dbltoproom > \the \ht \@currbox}%
2186         \else
2187         \fl@trace{fpstype: \the \@fpstype}%
2188         \fl@trace{Fail---no room dbltoproom-BANG?:}%
2189         \fl@trace{\@spaces space on page = \the \@dbltoproom.
2190                 Ht float: \the \ht \@currbox}%
2191 </trace>
2192         \fi
2193         \advance \@dbltoproom -\@textmin
2194 <*trace>
2195         \else
2196         \fl@trace{fpstype: \the \@fpstype}%
2197         \fl@trace{Fail---no room dbltoproom-ORD?:}%
2198         \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
2199                 Ht float: \the \ht \@currbox}%
2200 </trace>
2201         \fi
2202         \fi
2203         \if@tempswa
2204         \@bitor \@currtype \@deferlist
2205 <*trace>
2206         \fl@trace{(dbl)deferlist: \@deferlist: (before)}%
2207 </trace>
2208         not in fixfloats?
2209         \@testwrongwidth\@currbox
2210 <*trace>
2211         \fl@trace{type already on list: (dbl)defer}%
2212 </trace>
2213         \else
2214         \@tempdima -\ht\@currbox
2215         \advance\@tempdima
2216         -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
2217                 \dblfloatsep \fi
2218         \global \advance \@dbltoproom \@tempdima
2219         \global \advance \@colht \@tempdima
2220         \global \advance \@dbltopnum \m@ne
2221         \@cons \@dbltoplist \@currbox
2222 <*trace>

```

```

2223             \fl@trace{dbltopnum (after) = \the \@dbltopnum}%
2224             \fl@trace{***Success: dbltop}%
2225     </trace>
2226             \@inserttrue
2227     \fi
2228 \fi
2229 <*trace>
2230     \else
2231         \fl@trace{Fail: dbltopnum = \the \@dbltopnum: fpstype
2232                                     \the \@fpstype=ORD?}%
2233         \ifnum \@fpstype<\sixt@@n
2234             \fl@trace{ERROR: !t float not successful (addtodblcol)}%
2235         \fi
2236 </trace>
2237     \fi
2238 \fi
2239     \if@insert
2240     \else
2241 <*trace>
2242         \fl@trace{put on deferlist}%
2243 </trace>
2244         \@cons\@deferlist\@currbox
2245 <*trace>
2246         \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
2247 </trace>
2248     \fi
2249 <*trace>
2250     \fl@trace{End of addtodblcol -- locally count:}%
2251     \fl@trace{dbltop: \the \@dbltopnum.}%
2252 </trace>
2253 \endgroup
2254 <*trace>
2255     \fl@trace{End of addtodblcol -- globally count:}%
2256     \fl@trace{dbltop: \the \@dbltopnum.}%
2257 </trace>
2258 }%
2259 </2ekernel | latexrelease | fltrace>
2260 <latexrelease | fltrace>\EndIncludeInRelease
2261 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2262 <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
2263 <latexrelease | fltrace>\def\@addtodblcol{%
2264 <latexrelease | fltrace> \begingroup
2265 <*trace>
2266 <latexrelease | fltrace> \fl@trace{***Start addtodblcol}%
2267 </trace>
2268 <latexrelease | fltrace> \@insertfalse
2269 <latexrelease | fltrace> \@setfloattyperecounts
2270 <latexrelease | fltrace> \@getfpsbit \tw@
2271 <*trace>
2272 <latexrelease | fltrace> \fl@trace{fpstype \ifodd \@tempcnta OK
2273 <latexrelease | fltrace> \else not \fi dbltop: \the \@fpstype}%
2274 </trace>
2275 <latexrelease | fltrace> \ifodd\@tempcnta
2276 <latexrelease | fltrace> \@flsetnum \@dbltopnum

```



```

2277 <latexrelease | fltrace> \ifnum \@dbltopnum>\z@
2278 <latexrelease | fltrace> \@tempswafalse
2279 <latexrelease | fltrace> \ifdim \@dbltoproom>\ht\@currbox
2280 <latexrelease | fltrace> \@tempswatrue
2281 <*trace>
2282 <latexrelease | fltrace> \fl@trace{Space OK: \@dbltoproom =
2283 <latexrelease | fltrace> \the \@dbltoproom > \the \ht \@currbox
2284 <latexrelease | fltrace> (dbltoproom)}%
2285 </trace>
2286 <latexrelease | fltrace> \else
2287 <*trace>
2288 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
2289 </trace>
2290 <latexrelease | fltrace> \ifnum \@fpstype<\sixt@n
2291 <*trace>
2292 <latexrelease | fltrace> \fl@trace{BANG float ignoring \@dbltoproom}%
2293 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
2294 <latexrelease | fltrace> \the \@dbltoproom.
2295 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
2296 </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

2297 <latexrelease | fltrace> \advance \@dbltoproom \@textmin
2298 <*trace>
2299 <latexrelease | fltrace> \fl@trace{Local value of texmin: \the \@textmin}%
2300 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
2301 <latexrelease | fltrace> \the \@dbltoproom.
2302 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
2303 </trace>
2304 <latexrelease | fltrace> \ifdim \@dbltoproom>\ht\@currbox
2305 <latexrelease | fltrace> \@tempswatrue
2306 <*trace>
2307 <latexrelease | fltrace> \fl@trace{Space OK BANG: space on page =
2308 <latexrelease | fltrace> \the\@dbltoproom > \the\ht\@currbox}%
2309 <latexrelease | fltrace> \else
2310 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
2311 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-BANG?:}%
2312 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
2313 <latexrelease | fltrace> \the \@dbltoproom.
2314 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
2315 </trace>
2316 <latexrelease | fltrace> \fi
2317 <latexrelease | fltrace> \advance \@dbltoproom -\@textmin
2318 <*trace>
2319 <latexrelease | fltrace> \else
2320 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
2321 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-ORD?:}%
2322 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
2323 <latexrelease | fltrace> \the \@dbltoproom.
2324 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
2325 </trace>
2326 <latexrelease | fltrace> \fi
2327 <latexrelease | fltrace> \fi

```

```

2328 <latexrelease | fltrace> \if@tempswa
2329 <latexrelease | fltrace> \@bitor \@currtype \@dbldeferlist
2330 <*trace>
2331 <latexrelease | fltrace> \fl@trace{dbldeferlist:
2332 <latexrelease | fltrace> \@dbldeferlist: (before)}}%
2333 </trace>
2334 <latexrelease | fltrace> \if@test
2335 <*trace>
2336 <latexrelease | fltrace> \fl@trace{type already on list: dbldefer}%
2337 </trace>
2338 <latexrelease | fltrace> \else
2339 <latexrelease | fltrace> \@tempdima -\ht\@currbox
2340 <latexrelease | fltrace> \advance\@tempdima
2341 <latexrelease | fltrace> -\ifx \@dbltoplist\@empty
2342 <latexrelease | fltrace> \dbltextfloatsep
2343 <latexrelease | fltrace> \else \dblfloatsep \fi
2344 <latexrelease | fltrace> \global \advance \@dbltoproom \@tempdima
2345 <latexrelease | fltrace> \global \advance \@colht \@tempdima
2346 <latexrelease | fltrace> \global \advance \@dbltopnum \m@ne
2347 <latexrelease | fltrace> \@cons \@dbltoplist \@currbox
2348 <*trace>
2349 <latexrelease | fltrace> \fl@trace{dbltopnum (after) =
2350 <latexrelease | fltrace> \the \@dbltopnum}%
2351 <latexrelease | fltrace> \fl@trace{***Success: dbltop}%
2352 </trace>
2353 <latexrelease | fltrace> \@inserttrue
2354 <latexrelease | fltrace> \fi
2355 <latexrelease | fltrace> \fi
2356 <*trace>
2357 <latexrelease | fltrace> \else
2358 <latexrelease | fltrace> \fl@trace{Fail: dbltopnum = \the \@dbltopnum:
2359 <latexrelease | fltrace> fpstype \the \@fpstype=ORD?}%
2360 <latexrelease | fltrace> \ifnum \@fpstype<\sist@n
2361 <latexrelease | fltrace> \fl@trace{ERROR: !t float not successful
2362 <latexrelease | fltrace> (addtodblcol)}}%
2363 <latexrelease | fltrace> \fi
2364 </trace>
2365 <latexrelease | fltrace> \fi
2366 <latexrelease | fltrace> \fi
2367 <latexrelease | fltrace> \if@insert
2368 <latexrelease | fltrace> \else
2369 <*trace>
2370 <latexrelease | fltrace> \fl@trace{put on dbldeferlist}%
2371 </trace>
2372 <latexrelease | fltrace> \@cons\@dbldeferlist\@currbox
2373 <*trace>
2374 <latexrelease | fltrace> \fl@trace{dbldeferlist: \@dbldeferlist: (after)}}%
2375 </trace>
2376 <latexrelease | fltrace> \fi
2377 <*trace>
2378 <latexrelease | fltrace> \fl@trace{End of addtodblcol -- locally count:}%
2379 <latexrelease | fltrace> \fl@trace{ dbltop: \the \@dbltopnum.}%
2380 </trace>
2381 <latexrelease | fltrace> \endgroup

```

```

2382 <*trace>
2383 <latexrelease|fltrace> \fl@trace{End of addtodblcol -- globally count:}%
2384 <latexrelease|fltrace> \fl@trace{dbltop: \the \@dbltopnum.}%
2385 </trace>
2386 <latexrelease|fltrace>}%
2387 <latexrelease|fltrace>\EndIncludeInRelease

```

(End of definition for \@addtodblcol.)

\@addmarginpar

```

2388 <*2kernel>
2389 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
2390 \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@one
2391 \if@twocolumn
2392 \if@firstcolumn \@tempcnta\m@ne \fi
2393 \else
2394 \if@mparswitch
2395 \ifodd\c@page \else\@tempcnta\m@ne \fi
2396 \fi
2397 \if@reversemargin \@tempcnta -\@tempcnta \fi
2398 \fi
2399 \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
2400 \@tempdima\@mparbottom
2401 \advance\@tempdima -\@pageht
2402 \advance\@tempdima\ht\@marbox
2403 \ifdim\@tempdima >\z@
2404 \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
2405 \else
2406 \@tempdima\z@
2407 \fi
2408 \global\@mparbottom\@pageht
2409 \global\advance\@mparbottom\@tempdima
2410 \global\advance\@mparbottom\dp\@marbox
2411 \global\advance\@mparbottom\marginparpush
2412 \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

2413 \global\setbox \@marbox
2414 \vbox {\vskip \@tempdima
2415 \box \@marbox}%
2416 \global \ht\@marbox \z@
2417 \global \dp\@marbox \z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

2418 \kern -\@pagedp
2419 \nointerlineskip
2420 \hb@xt@\columnwidth
2421 {\ifnum \@tempcnta >\z@
2422 \hskip\columnwidth \hskip\marginparsep
2423 \else
2424 \hskip -\marginparsep \hskip -\marginparwidth
2425 \fi
2426 \box\@marbox \hss}%

```

For this reason the following code can vanish:

```

\nobreak          %% No longer needed.  CAR92/12
\vskip -\@tempdima  %% No longer needed.  CAR92/12

```

```

2427 \nointerlineskip
2428 \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}

```

(End of definition for \addmarginpar.)

2.2.2 Kludgeins

This part of the file is part of the implementation of the following two new commands for L^AT_EX 2_ε.

`\enlargethispage{<dim>}`

Adds <dim> to the height of the current column only. On the printed page the bottom of this column is extended downwards by exactly <dim> without having any effect on the placement of the footer; this may result in an overprinting.

`\enlargethispage*{<dim>}`

Similar to `\enlargethispage` but it tries to squeeze the column to be printed in as small a space as possible, ie it uses any shrinkability in the column. If the column was not explicitly broken (e.g. with `\pagebreak`) this may result in an overfull box message but except for this it will come out as expected (if you know what to expect).

The star form of this command is dedicated to Leslie Lamport, the other we need for ourselves (FMi, CAR).

These commands may well have unwanted effects if used soon before a `\clearpage`: please give keep them clear of such places.

`\@kludgeins` The insert which makes T_EX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```

2429 \newinsert \@kludgeins
2430 \global\dimen\@kludgeins \maxdimen
2431 \global\count\@kludgeins 1000

```

(End of definition for \@kludgeins.)

`\enlargethispage` The user command.

```

\enlargethispage* 2432 \gdef \enlargethispage {%
2433   \@ifstar
2434   {%
2435   <*trace>
2436   \fl@trace{Enlarging page height * }%
2437   </trace>
2438   \@enlargepage{\hbox{\kern\p@}}}%
2439   {%
2440   <*trace>
2441   \fl@trace{Enlarging page height exactly---}%
2442   </trace>
2443   \@enlargepage\@empty}%
2444   }

```

(End of definition for \enlargethispage and \enlargethispage*.)

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```

2445 \gdef\@enlargepage#1#2{%
2446   \*trace)
2447   \fl@trace{\@spaces\@spaces by #2}%
2448   \*trace)
2449   \@tempskipa#2\relax
2450   \ifdim \@tempskipa>.5\maxdimen
2451     \@latex@error{Suggested\space extra\space height\space
2452                 (\the\@tempskipa)\space dangerously\space
2453                 large}\@eha
2454   \else
2455     \ifdim \vsize<.5\maxdimen
2456   \*trace)
2457     \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
2458   \*trace)
2459     \bsphack
2460     \insert\@kludgeins{#1\vskip-\@tempskipa}%
2461     \@esphack

```

This next bit is for tracing only:

```

2462 \*trace)
2463   \ifvmode \par
2464   \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
2465   \fi
2466 \*trace)
2467   \else
2468     \@latex@error{Page\space height\space already\space
2469                 too\space large}\@eha
2470   \fi
2471 \fi
2472 }

```

(End of definition for \@enlargepage.)

`\ShowFloat` This command provides some information about the contents of a float register. Float registers have internal names of the form `\bx@<Uppercase-letter(s)-or numbers>` and you specify just this letter or letters as the argument, e.g., `\ShowFloat{A}`. (There is not much error recovery if you specify something that isn't a float.)

```

2473 \*kernel)
2474 \*2kernel | latexrelease)
2475 \*kernel)\IncludeInRelease{2021/11/15}%
2476 \*kernel)\ShowFloat\Show float register contents}%
2477 \def\ShowFloat#1{\begingroup
2478   \let \fl@trace \fl@tracemessage
2479   \fl@trace{***Float #1 details:}%
2480   \ifcsname bx@#1\endcsname
2481     \expandafter\fl@ShowFloat\csname bx@#1\endcsname
2482   \else
2483     \fl@trace{Not a float!}%
2484   \fi
2485 \endgroup
2486 }
2487 \def\fl@ShowFloat#1{%
2488   \fl@traceval{\count#1}% % this here should be interpreted on day

```

```

2489 \fl@traceval{\ht#1}%
2490 \fl@traceval{\dp#1}%
2491 \fl@traceval{\wd#1}%
2492 {\tracingonline1\showboxbreadth10\showboxdepth3\showbox#1}%
2493 }

```

Here are two definitions from `fltrace` that make the above code work:

```

2494 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
2495 \def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
2496 </2ekernel | latexrelease>
2497 <latexrelease>\EndIncludeInRelease
2498 <latexrelease>\IncludeInRelease{0000/00/00}%
2499 <latexrelease> \ShowFloat{Show float register contents}%
2500 <latexrelease>
2501 <latexrelease>\let\ShowFloat\@undefined
2502 <latexrelease>\let\fl@ShowFloat\@undefined
2503 <latexrelease>\let\fl@traceval\@undefined
2504 <latexrelease>\let\fl@tracemessage\@undefined
2505 <latexrelease>\EndIncludeInRelease

```

(End of definition for \ShowFloat.)

2.2.3 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in `LATEX2e`.

`\suppressfloats`

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

[t] suppresses only floats at the top of the page [b] suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, `!`, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the following are ignored:

- all restrictions on the number of floats which can appear;
- all explicit restrictions on the amount of space which should (not) be occupied by floats and/or text.

The mechanism will still attempt to ensure that pages are not overfull.

These specifiers override, for the single float, the suppression commands described above.

In its current form, it also supplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

```

\fl@trace Set-up tracing for floats independent of other tracing as it produces mega-output. Default
\tracefloatsoff is no tracing.
\tracefloats
\fl@traceval
\tracefloatvals
\fl@tracemessage
2506 <*\fltrace>
2507 \def \fl@tracemessage #1{{\let\@elt\@empty\typeout{LaTeX2e: #1}}}
2508 \def \tracefloats{\let \fl@trace \fl@tracemessage}
2509 \def \tracefloatsoff {\let \fl@trace \@gobble}
2510 \tracefloatsoff
2511 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
2512 \IncludeInRelease{2015/01/01}{\tracefloatvals}%
2513 {trace float vals}%
2514 \def \tracefloatvals{%

As \@dblfloatplacement sets \f@depth it needs to be run inside a group, otherwise the
float placement will test for the wrong value.65
2515 \begingroup

When the user requests \tracefloatvals then they should show regardless of the tracing
state, so locally we make sure that it is activated.
2516 \tracefloats
2517 \@dblfloatplacement
2518 \@floatplacement
2519 \fl@trace{***Float placement parameters:}%
2520 \fl@traceval\@colnum
2521 \fl@traceval\@colroom
2522 \fl@traceval\@topnum
2523 \fl@traceval\@toproom
2524 \fl@traceval\@botnum
2525 \fl@traceval\@botroom
2526 \fl@traceval\@fpmin
2527 \fl@trace{\string\textfraction = \textfraction}%
2528 \fl@traceval\@dbltopnum
2529 \fl@traceval\@dbltoproom
2530 \fl@trace{\string\textfraction = \textfraction}%
2531 \fl@trace{toplist: \@toplist}%
2532 \fl@trace{botlist: \@botlist}%
2533 \fl@trace{midlist: \@midlist}%
2534 \fl@trace{deferlist: \@deferlist}%
2535 \fl@trace{dbltoplist: \@dbltoplist}%
2536 %Fmi \fl@trace{dbldeferlist: \@dbldeferlist}%
2537 \endgroup
2538 }
2539 \EndIncludeInRelease
2540 \IncludeInRelease{0000/00/00}{\tracefloatvals}%
2541 {trace float vals}%
2542 \def \tracefloatvals{%
2543 \begingroup
2544 \tracefloats
2545 \@dblfloatplacement
2546 \@floatplacement
2547 \fl@trace{***Float placement parameters:}%
2548 \fl@traceval\@colnum
2549 \fl@traceval\@colroom

```

⁶⁵This is a somewhat questionable design.

```

2550 \fl@traceval\@topnum
2551 \fl@traceval\@toproom
2552 \fl@traceval\@botnum
2553 \fl@traceval\@botroom
2554 \fl@traceval\@fpmin
2555 \fl@trace{\string\textfraction = \textfraction}%
2556 \fl@traceval\@dbltopnum
2557 \fl@traceval\@dbltoproom
2558 \fl@trace{\string\textfraction = \textfraction}%
2559 \fl@trace{toplist: \@toplist}%
2560 \fl@trace{botlist: \@botlist}%
2561 \fl@trace{midlist: \@midlist}%
2562 \fl@trace{deferlist: \@deferlist}%
2563 \fl@trace{dbltoplist: \@dbltoplist}%
2564 % next line only in old releases
2565 \fl@trace{dbldeferlist: \@dbldeferlist}%
2566 \endgroup
2567 }
2568 \EndIncludeInRelease

```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```

2569 \@ifpackageloaded{flafter}
2570 {
2571   \PackageWarningNoLine
2572     {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
                Attempting to recover by reloading 'flafter'}%
2573   \expandafter\let\csname ver@flafter.sty\endcsname\relax
2574   \def\reserved@a#1{%
2575     \expandafter\let\csname string#1+flafter+IIR\endcsname\relax}%
2576   \reserved@a\@addtocurcol
2577   \reserved@a\@addtonextcol
2578   \RequirePackage{flafter}}{}
2579 </fltrace>

```

As the code for `flafter` will contain tracing calls so that it works in conjunction with `fltrace` we need to provide a dummy definition for `\fl@trace` in that package.

```

2580 <flafter>
2581 \providecommand\fl@trace[1]{}
2582 </flafter>

```

(End of definition for \fl@trace and others.)

\suppressfloats **\@flstop** Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular float by an `!` specifier.

```

2583 <*2kernel>
2584 \def \suppressfloats {%
2585   \@ifnextchar [%
2586     \@flstop
2587     {\global \@colnum \z@}%
2588   }

```

Maybe this should be a loop over `#1`?

```

2589 \def \@flstop [#1]{%
2590   \if t#1%

```



```

2591     \global \@topnum \z@
2592   \fi
2593   \if b#1%
2594     \global \@botnum \z@
2595   \fi
2596 }

```

(End of definition for `\suppressfloats` and `\flstop`.)

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` First a new count register to go with `\@currtype`.
`\@reqcolroom` Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.
`\@textfloatsheight`

Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of `\@addtocurcol` which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```

2597 \newcount \@fpstype
2598 \newdimen \@reqcolroom
2599 \newdimen \@textfloatsheight

```

(End of definition for `\@fpstype`, `\@reqcolroom`, and `\@textfloatsheight`.)

`\saved@reqcolroom` Saved value of `\@reqcolroom`; this is needed when making several tests.

```

2600 \if2\kernel
2601 \if*2\kernel | latexrelease
2602 \if<latexrelease> \IncludeInRelease{2025/06/01}%
2603 \if<latexrelease> {\saved@reqcolroom}{float placement calculation}%
2604 \newdimen \saved@reqcolroom
2605 \if2\kernel | latexrelease
2606 \if<latexrelease> \EndIncludeInRelease
2607 \if<latexrelease> \IncludeInRelease{0000/00/00}%
2608 \if<latexrelease> {\saved@reqcolroom}{float placement calculation}%
2609 \if<latexrelease> \let \saved@reqcolroom \@undefined
2610 \if<latexrelease> \EndIncludeInRelease

```

(End of definition for `\saved@reqcolroom`.)

`\@fpsadddefault` Adds the default placement to what is already there.

Should not need to change this, but could do it as follows:

```

def \@fpsadddefault {%
  \temptokena \expandafter\expandafter\expandafter
    {\csname fps@\@capytype \endcsname}%
  \edef \reserved@a {\the\@temptokena}%
  \@onelevel@sanitize \reserved@a
  \edef \@fps {\@fps\reserved@a}%
}

```

```

2611 <*2ekernel| fltrace)
2612 \def \@fpsadddefault {%
2613 <*trace)
2614   \fl@trace{fps changed from: \@fps}%
2615 </trace)
2616   \edef \@fps {\@fps\csname fps@\@capytype \endcsname}%
2617   \@latex@warning {%
2618     No positions in optional float specifier.\MessageBreak
2619     Default added (so using ‘\@fps’)}%
2620 }

```

(End of definition for \@fpsadddefault.)

\@setfloattypecounts Sets counters \@fpstype and \@currtype.

BANG == bit4 of \count\@currbox = 0.

```

2621 \def \@setfloattypecounts {%
2622   \@currtype \count\@currbox
2623   \@fpstype \count\@currbox
2624   \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
2625   \advance \@fpstype -\@currtype
2626 <*trace)
2627   \fl@trace{(mod 32) fpstype: \the \@fpstype}%
2628   \fl@trace{(mult of 32) currtype: \the \@currtype}%
2629 % Tracing only: but some should be changed into real errors/warnings?
2630   \ifnum \@fpstype<\sist@n
2631     \ifnum \@fpstype=\z@
2632       \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
2633     \fi
2634     \ifnum \@fpstype=\@ne
2635       \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
2636     \fi
2637     \fl@trace{BANG float}%
2638   \else
2639     \ifnum \@fpstype=\sist@n
2640       \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
2641     \fi
2642     \ifnum \@fpstype=17
2643       \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
2644     \fi
2645     \fl@trace{ORD float}%
2646   \fi
2647 </trace)
2648 }
2649 </2ekernel| fltrace)

```

(End of definition for \@setfloattypecounts.)

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```

2650 <*2ekernel)
2651 \def \@getfpsbit {%
2652   \@boxfpsbit \@currbox
2653 }

```

(End of definition for \@getfpsbit.)

`\@boxfpsbit` Used above.

```
2654 \def \@boxfpsbit #1#2{%
2655     \@tempcnta \count#1%
2656     \divide \@tempcnta #2\relax
2657 }
```

(End of definition for \@boxfpsbit.)

`\@testfp` New definition of the float page test.

```
2658 \def \@testfp #1{%
2659     \@boxfpsbit #18\relax % Really '#1 8' for human readers!
2660     \ifodd \@tempcnta
2661     \else
2662         \@testtrue
2663     \fi
2664 }
```

(End of definition for \@testfp.)

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
2665 \def \@setfpsbit #1{%
2666     \@tempcntb \@tempcnta
2667     \divide \@tempcntb #1\relax
2668     \ifodd \@tempcntb
2669     \else
2670         \advance \@tempcnta #1\relax
2671     \fi
2672 }
2673 </2ekernel>
```

(End of definition for \@setfpsbit.)

`\@resethfps` Globally adds `t` as a possible location for an `h` or `!h` only placement: this must be done using the count.

Although it will leave `\@fpstype` set to 17 even if it was originally 1, this does not matter since it is the last thing in `\@addtocurcol`.

```
2674 <*2ekernel | fltrace>
2675 \def \@resethfps {%
2676     \let\reserved@a\@empty
2677     \ifnum \@fpstype=\@ne
2678         \def \reserved@a {!}%
2679         \@fpstype 17
2680     \fi
2681     \ifnum \@fpstype=17
2682         \global \advance \count\@currbox \tw@
2683         \@latex@warning@no@line {%
2684             '\reserved@a h' float specifier changed to '\reserved@a ht'}%
2685 <*trace>
2686         \fl@trace{%
2687             't' added to '\reserved@a h'- new Count: \the \count\@currbox}%
2688 </trace>
2689     \fi
2690 }
```

(End of definition for \@resetfps.)
Special stuff for BANG floats.

\@flsetnum Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assignments are safe so long as the counter involved is only consulted once (i.e. only for the ‘bang float’) with the changed value. This is the case within \@addtocurcol because it is used only once within a call of the output routine (which forms a group).

For \@addtonextcol this is achieved by putting a group around its code; this is needed because it is called (by \@startcolumn) for each float which was on the deferlist. Almost identical considerations pertain to \@addtodblcol. There may be more efficient ways to handle this, but the group seems to be the simplest.

```

2691 \def \@flsetnum #1{%
2692   \ifnum #1=0
2693     \ifnum \@fpstype<\sixt@@n
2694       \ifnum #1=\z@
2695         \ifnum \@fpstype<\sixt@@n
2696           \ifnum #1=\z@
2697             \ifnum \@fpstype<\sixt@@n
2698               \fl@trace{BANG float resetting \string#1 to 1}%
2699             \fi
2700           \fi
2701         \fi
2702       \fi
2703     \fi
2704     \fl@trace{#1 (before) = \the #1}%
2705   \fi
2706 }

```

(End of definition for \@flsetnum.)

\@flsettextmin This ignores \textfraction space restriction in case BANG.

```

2707 \def \@flsettextmin {%
2708   \ifnum #1=0
2709     \ifnum \@fpstype<\sixt@@n
2710       \ifnum #1=\z@
2711         \ifnum \@fpstype<\sixt@@n
2712           \fl@trace{BANG ignoring textmin}%
2713         \fi
2714       \fi
2715     \fi
2716     \ifnum \@fpstype<\sixt@@n
2717       \ifnum #1=\z@
2718         \ifnum \@fpstype<\sixt@@n
2719           \fl@trace{ORD textmin = \the \@textmin}%
2720         \fi
2721       \fi
2722     \fi

```

(End of definition for \@flsettextmin.)

`\@flcheckspace` This ignores space restriction in case BANG; this is still slightly conservative since it does not allow for the fact that, if there is no text in the column then `\textfloatsep` is not needed. Sets `@tempswa` true if there is room for `\@currbox`.

```

2723 \</2ekernel | fltrace>
2724 \<*2ekernel | fltrace | latexrelease>
2725 \<latexrelease | fltrace>\IncludeInRelease{2025/06/01}%
2726 \<latexrelease | fltrace>                {\@flcheckspace}{float placement calculation}%
2727 \def \@flcheckspace #1#2{%

```

When this test is executed, the value of `\@reqcolroom` may no longer be correct due to a previous test that altered it. We therefore reset it to a value saved earlier.

```

2728     \@reqcolroom \saved@reqcolroom
2729     \advance \@reqcolroom
2730     \ifx #2\@empty \textfloatsep \else \floatsep \fi
2731 \<*trace>
2732     \fl@trace{colroom = \the \@colroom
2733                                     (flcheckspace \string#1 \string#2)}%
2734     \fl@trace{reqcolroom = \the \@reqcolroom
2735                                     (flcheckspace \string#1 \string#2)}%
2736 \</trace>
2737     \ifdim \@colroom>\@reqcolroom
2738     \ifdim #1>\ht\@currbox
2739     \@tempswatrue
2740 \<*trace>
2741     \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2742                                     (flcheckspace \string#1 \string#2)}%
2743 \</trace>
2744     \else
2745 \<*trace>
2746     \fl@trace{fpstype: \the \@fpstype
2747                                     (flcheckspace \string#1 \string#2)}%
2748 \</trace>
2749     \ifnum \@fpstype<\sift@n
2750 \<*trace>
2751     \fl@trace{BANG float ignoring #1
2752                                     (flcheckspace \string#1 \string#2):}%
2753     \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2754                                     \space BANG}%
2755 \</trace>
2756     \@tempswatrue
2757 \<*trace>
2758     \else
2759     \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2760                                     (fpstype \the \@fpstype=ORD?):}%
2761     \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2762                                     \space ORD?}%
2763 \</trace>
2764     \fi
2765     \fi
2766 \<*trace>
2767     \else
2768     \fl@trace{Fail---no room at 2nd test of colroom
2769                                     (flcheckspace \string#1 \string#2)}%
2770 \</trace>

```

```

2771 \fi
2772 }
2773 \</2ekernel>| fltrace | latexrelease>
2774 <latexrelease| fltrace>\EndIncludeInRelease
2775 <latexrelease| fltrace>\IncludeInRelease{0000/00/00}%
2776 <latexrelease| fltrace> \{\@flcheckspace\}{float placement calculation}%
2777 <latexrelease| fltrace>
2778 <latexrelease| fltrace>\def \@flcheckspace #1#2{%
2779 <latexrelease| fltrace> \advance \@reqcolroom
2780 <latexrelease| fltrace> \ifx #2\@empty \textfloatsep \else \floatsep \fi
2781 <*trace>
2782 <latexrelease| fltrace> \fl@trace{colroom = \the \@colroom
2783 <latexrelease| fltrace> (flcheckspace \string#1 \string#2)}%
2784 <latexrelease| fltrace> \fl@trace{reqcolroom = \the \@reqcolroom
2785 <latexrelease| fltrace> (flcheckspace \string#1 \string#2)}%
2786 </trace>
2787 <latexrelease| fltrace> \ifdim \@colroom>\@reqcolroom
2788 <latexrelease| fltrace> \ifdim #1>\ht\@currbox
2789 <latexrelease| fltrace> \@tempwattrue
2790 <*trace>
2791 <latexrelease| fltrace> \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2792 <latexrelease| fltrace> (flcheckspace \string#1 \string#2)}%
2793 </trace>
2794 <latexrelease| fltrace> \else
2795 <*trace>
2796 <latexrelease| fltrace> \fl@trace{fpstype: \the \@fpstype
2797 <latexrelease| fltrace> (flcheckspace \string#1 \string#2)}%
2798 </trace>
2799 <latexrelease| fltrace> \ifnum \@fpstype<\sist@n
2800 <*trace>
2801 <latexrelease| fltrace> \fl@trace{BANG float ignoring #1
2802 <latexrelease| fltrace> (flcheckspace \string#1 \string#2):}%
2803 <latexrelease| fltrace> \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2804 <latexrelease| fltrace> \space BANG}%
2805 </trace>
2806 <latexrelease| fltrace> \@tempwattrue
2807 <*trace>
2808 <latexrelease| fltrace> \else
2809 <latexrelease| fltrace> \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2810 <latexrelease| fltrace> (fpstype \the \@fpstype=ORD?):}%
2811 <latexrelease| fltrace> \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2812 <latexrelease| fltrace> \space ORD?}%
2813 </trace>
2814 <latexrelease| fltrace> \fi
2815 <latexrelease| fltrace> \fi
2816 <*trace>
2817 <latexrelease| fltrace> \else
2818 <latexrelease| fltrace> \fl@trace{Fail---no room at 2nd test of colroom
2819 <latexrelease| fltrace> (flcheckspace \string#1 \string#2)}%
2820 </trace>
2821 <latexrelease| fltrace> \fi
2822 <latexrelease| fltrace>}
2823 <latexrelease| fltrace>\EndIncludeInRelease

```

(End of definition for \@flcheckspace.)

\@flupdates This updates everything when a float is placed.

```
2824 <*2ekernel>
2825 \def \@flupdates #1#2#3{%
2826     \global \advance #1\m@ne
2827     \global \advance \@colnum \m@ne
2828     \@tempdima -\ht\@currbox
2829     \advance \@tempdima
2830     -\ifx #3\@empty \textfloatsep \else \floatsep \fi
2831     \global \advance #2\@tempdima
2832     \global \advance \@colroom \@tempdima
2833     \@cons #3\@currbox
2834 }
2835 </2ekernel>
```

(End of definition for \@flupdates.)

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn` floatplacement was wrong: dbl not needed, ord needed.
3. `\@floatplacement` was not called after `\@startdblcol` or `\@topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\@topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the ‘float’ from `\@topnewpage`.
5. If the specifier was just h and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a ‘th’: this is only an error-recovery action, putting just h or !h should be deprecated.
6. `\@dblmaxsep` was ‘the maximum of `\dblfloatsep` and `\dbltextfloatsep`’. But it was never used! Now gone completely, like `\@maxsep`.
7. After an h float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?
8. Two consecutive h floats are separated by twice `\intertextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an h float immediately follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.
9. Now `\@addtocurcol` checks first for just p fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.
10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?

11. Why is `\@colroom` set in `\@docclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the ‘wrong page’, but this may be a coding problem.
14. The `!` option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\@textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?
15. There are four possibilities for supporting this:
`\twocolumn[\maketitle more text]`
 One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twocolumn[more text]` will work. The former is more robust from the user’s viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global `twocolumn` flag locally within `\@topnewpage`. Yet another is to move the column count register from the `multicol` package into the kernel. This has been done.
16. Where should the reinserts be put to maximise the probability that footnotes come out on the correct page? Or should we go for as much compatibility as possible (but see next item)?
17. Should we continue to support (as much as possible) `\samepage`? Some of its intended functionality is now advertised as being provided by `\enlargethispage`. Use of either is likely to result in wrongly placed footnotes, marginals, etc. Which should have priority: obeying the pagination instructions, or correct placement of notes/marginalia?
18. Is the adjustment of space to cause shrinking in the kludge-* case correct? Should it be limited to 0pt?
19. Is the setting of `\boxmaxdepth` in `makecol` and friends needed? It only has any effect if `\@textbottom` ends with a box or rule, in which case the `vskip` to allow for its depth should also be added. If it is kept, it should probably be the last thing in the box. It has now been removed.
 It would perhaps be better to document that `\@textbottom` and `\@texttop` must have natural height 0pt.
20. I cannot see why the `vskip` adjustment for the depth is needed if `boxmaxdepth` is used to ensure that there is never a too deep box.
21. The value of `\boxmaxdepth` should be explicitly set whenever necessary: it is too risky to assume that it has any particular value. Care is needed in deciding what to set it to.

It is interesting to note that the value of `\boxmaxdepth` is unique in being read before the local settings for the box group are reset; all other parameter settings which affect the box construction use their values outside the box group.

22. Should `\@maxdepth` store the setting of `\maxdepth` from `lplain`? Or should we provide a proper interface to class files for setting these?

An analysis of various other macros.

`\@opcol` should do `\@floatplacement`, but where? Right at the end, since it always occurs at the start of a column.

```
\def\@opcol{%
  % Why is this done first?
  \global \@mparbottom \z@
  \if@twocolumn
    \@outputdblcol
  \else
    \@outputpage
    % This is not needed since it is done at the end of
    %   |\@outputpage|:
    \global \@colht \textheight
  \fi}
```

Only tracing has been added to these.

```
2836 <latexrelease|fltrace>\IncludeInRelease{2017/01/01}%
2837 <latexrelease|fltrace> {\@makefcolumn}{negative height floats}%
2838 <*2ekernel|fltrace|latexrelease>
2839 \def\@makefcolumn #1{%
2840   \begingroup

2841   \@fpmin -\maxdimen

2842   \let \@testfp \@gobble
2843   \@tryfcolumn #1%
2844   \endgroup
2845   <*trace>
2846   \if@fcolmade
2847     \fl@trace{PAGE: in \string\clearpage
2848               \if@twocolumn ---twocolumn\fi---}%
2849     \fl@trace{----- float column/page completed from \string#1}%
2850     \fi
2851   </trace>
2852   }

2853 <latexrelease|fltrace>\EndIncludeInRelease
2854 <latexrelease|fltrace>\IncludeInRelease{0000/00/00}%
2855 <latexrelease|fltrace> {\@makefcolumn}{negative height floats}%
2856 <latexrelease|fltrace>\def\@makefcolumn #1{%
2857   <latexrelease|fltrace> \begingroup
2858   <latexrelease|fltrace>   \@fpmin \z@
2859   <latexrelease|fltrace>   \let \@testfp \@gobble
2860   <latexrelease|fltrace>   \@tryfcolumn #1%
2861   <latexrelease|fltrace> \endgroup
2862   <*trace>
```

```

2863 <latexrelease | fltrace> \if@fcolmade
2864 <latexrelease | fltrace> \fl@trace{PAGE: in \string\clearpage
2865 <latexrelease | fltrace> \if@twocolumn ---twocolumn\fi---}%
2866 <latexrelease | fltrace> \fl@trace{----- float column/page completed
2867 <latexrelease | fltrace> from \string#1}%
2868 <latexrelease | fltrace> \fi
2869 </trace>
2870 <latexrelease | fltrace>}
2871 <latexrelease | fltrace>\EndIncludeInRelease
2872 </2ekernel | fltrace | latexrelease>

```

This will line up the last baselines in the two columns provided they are constructed in the normal way: i.e. ending in a skip of minus the original depth, with `\@textbottom` adding nothing.

Thus again it is essential for `\@textbottom` to have depth 0pt.

```

2873 <latexrelease | fltrace>\IncludeInRelease{2025/06/01}%
2874 <latexrelease | fltrace> {\@outputdblcol}{Use new mark mechanism}%
2875 <*2ekernel | fltrace | latexrelease>

2876 \def\@outputdblcol{%
2877   \if@firstcolumn
2878     \global\@firstcolumnfalse

```

Save the left column

```

2879   \global\setbox\@leftcolumn\copy\@outputbox
2880 <fltrace> \fl@trace{PAGE: first column boxed}%

2881   \else
2882     \global\@firstcolumntrue
2883     \setbox\@outputbox\vbox{%
2884       \hb@xt@\textwidth{%
2885         \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
2886         \hfil

```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```

2887       {\normalcolor\vrule \@width\columnseprule}%
2888       \hfil
2889       \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2890 <fltrace> \fl@trace{PAGE: second column also boxed}%
2891   \@combinedblfloats
2892   \@outputpage
2893 <fltrace> \fl@trace{PAGE: two column page completed}%
2894   \begingroup
2895     \dblfloatplacement
2896     \@startdblcolumn
2897     \@whiles\if@fcolmade \fi{\@outputpage
2898 <fltrace> \fl@trace{PAGE: double float page completed}%
2899     \@startdblcolumn}%
2900   \endgroup
2901   \fi}%

2902 <latexrelease | fltrace>\EndIncludeInRelease
2903 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
2904 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2905 <latexrelease | fltrace>\def\@outputdblcol{%

```

```

2906 <latexrelease | fltrace> \if@firstcolumn
2907 <latexrelease | fltrace> \global\@firstcolumnfalse
2908 <latexrelease | fltrace> \global\setbox\@leftcolumn\copy\@outputbox
2909 <latexrelease | fltrace> \splitmaxdepth\maxdimen
2910 <latexrelease | fltrace> \vbadness\maxdimen
2911 <latexrelease | fltrace> \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
2912 <latexrelease | fltrace> \setbox\@outputbox\vsplit\@outputbox to\maxdimen
2913 <latexrelease | fltrace> \toks@\expandafter{\topmark}%
2914 <latexrelease | fltrace> \xdef\@firstcoltopmark{\the\toks@}%
2915 <latexrelease | fltrace> \toks@\expandafter{\splitfirstmark}%
2916 <latexrelease | fltrace> \xdef\@firstcolfirstmark{\the\toks@}%
2917 <latexrelease | fltrace> \ifx\@firstcolfirstmark\@empty
2918 <latexrelease | fltrace> \global\let\@setmarks\relax
2919 <latexrelease | fltrace> \else
2920 <latexrelease | fltrace> \gdef\@setmarks{%
2921 <latexrelease | fltrace> \let\firstmark\@firstcolfirstmark
2922 <latexrelease | fltrace> \let\topmark\@firstcoltopmark}%
2923 <latexrelease | fltrace> \fi
2924 <latexrelease | fltrace> \else
2925 <latexrelease | fltrace> \global\@firstcolumntrue
2926 <latexrelease | fltrace> \setbox\@outputbox\vbox{%
2927 <latexrelease | fltrace> \hb@xt@\textwidth{%
2928 <latexrelease | fltrace> \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
2929 <latexrelease | fltrace> \hfil
2930 <latexrelease | fltrace> {\normalcolor\vrule \@width\columnseprule}%
2931 <latexrelease | fltrace> \hfil
2932 <latexrelease | fltrace> \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2933 <latexrelease | fltrace> \@combinedblfloats
2934 <latexrelease | fltrace> \@setmarks
2935 <latexrelease | fltrace> \@outputpage
2936 <latexrelease | fltrace> \begingroup
2937 <latexrelease | fltrace> \@dblfloatplacement
2938 <latexrelease | fltrace> \@startdblcolumn
2939 <latexrelease | fltrace> \@whiles\if@fcolmade \fi{\@outputpage
2940 <latexrelease | fltrace> \@startdblcolumn}%
2941 <latexrelease | fltrace> \endgroup
2942 <latexrelease | fltrace> \fi}
2943 <latexrelease | fltrace> \EndIncludeInRelease

2944 <latexrelease | fltrace> \IncludeInRelease{0000/00/00}%
2945 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2946 <latexrelease | fltrace> \def\@outputdblcol{%
2947 <latexrelease | fltrace> \if@firstcolumn
2948 <latexrelease | fltrace> \global \@firstcolumnfalse
2949 <latexrelease | fltrace> \global \setbox\@leftcolumn \box\@outputbox
2950 <*trace>
2951 <latexrelease | fltrace> \fl@trace{PAGE: first column boxed}%
2952 </trace>
2953 <latexrelease | fltrace> \else
2954 <latexrelease | fltrace> \global \@firstcolumntrue
2955 <latexrelease | fltrace> \setbox\@outputbox \vbox {%
2956 <latexrelease | fltrace> \hb@xt@\textwidth {%
2957 <latexrelease | fltrace> \hb@xt@\columnwidth {%
2958 <latexrelease | fltrace> \box\@leftcolumn \hss}%
2959 <latexrelease | fltrace> \hfil

```

```

2960 <latexrelease | fltrace>                {\normalcolor\vrule
2961 <latexrelease | fltrace>                \width\columnseprule}%
2962 <latexrelease | fltrace>                \hfil
2963 <latexrelease | fltrace>                \hb@xt@\columnwidth {%
2964 <latexrelease | fltrace>                \box\@outputbox \hss}%
2965 <latexrelease | fltrace>                }%
2966 <latexrelease | fltrace>                }%
2967 <*trace>
2968 <latexrelease | fltrace>                \fl@trace{PAGE: second column also boxed}%
2969 </trace>
2970 <latexrelease | fltrace>                \@combinedblfloats
2971 <latexrelease | fltrace>                \@outputpage
2972 <*trace>
2973 <latexrelease | fltrace>                \fl@trace{PAGE: two column page completed}%
2974 </trace>
2975 <latexrelease | fltrace>                \begingroup
2976 <latexrelease | fltrace>                \dblfloatplacement
2977 <latexrelease | fltrace>                \startdblcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```

2978 <latexrelease | fltrace>                \@whiles\if@fcolmade \fi
2979 <latexrelease | fltrace>                {\@outputpage
2980 <*trace>
2981 <latexrelease | fltrace>                \fl@trace{PAGE: double float page completed}%
2982 </trace>
2983 <latexrelease | fltrace>                \@startdblcolumn}%
2984 <latexrelease | fltrace>                \endgroup
2985 <latexrelease | fltrace> \fi
2986 <latexrelease | fltrace>}%
2987 <latexrelease | fltrace>\EndIncludeInRelease
2988 </2ekernel | fltrace | latexrelease>

```

2.2.4 Float placement parameters

The main purpose of this section is to ensure that all the float-placement parameters which need to be set in a class file or package have been declared. It also describes their use and sets values for them which are reasonable for typical documents using US letter or A4 sized paper.

Limits for the placement of floating objects

\c@topnumber This counter holds the maximum number of floats that can appear at the top of a text page or column.

```

2989 <*2ekernel>
2990 \newcount\c@topnumber
2991 \setcounter{topnumber}{2}

```

(End of definition for `\c@topnumber`.)

\topfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the top.

```

2992 \newcommand\topfraction{.7}

```

(End of definition for \topfraction.)

\c@bottomnumber This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

2993 `\newcount\c@bottomnumber`

2994 `\setcounter{bottomnumber}{1}`

(End of definition for \c@bottomnumber.)

\bottomfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the bottom.

2995 `\newcommand\bottomfraction{.3}`

(End of definition for \bottomfraction.)

\c@totalnumber This counter holds the maximum number of floats that can appear on any text page or column.

2996 `\newcount\c@totalnumber`

2997 `\setcounter{totalnumber}{3}`

(End of definition for \c@totalnumber.)

\textfraction This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

2998 `\newcommand\textfraction{.2}`

(End of definition for \textfraction.)

\floatpagefraction This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a ‘float page’ is produced.

2999 `\newcommand\floatpagefraction{.5}`

(End of definition for \floatpagefraction.)

\c@dbltopnumber This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

3000 `\newcount\c@dbltopnumber`

3001 `\setcounter{dbltopnumber}{2}`

(End of definition for \c@dbltopnumber.)

\dbltopfraction This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column floats at the top.

3002 `\newcommand\dbltopfraction{.7}`

(End of definition for \dbltopfraction.)

\dblfloatpagefraction This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a ‘double-column float page’ is produced.

3003 `\newcommand\dblfloatpagefraction{.5}`

(End of definition for \dblfloatpagefraction.)

Floats on a text page

`\floatsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intextsep` is the space between in-text floats and the text.

```
3004 \newskip\floatsep
3005 \newskip\textfloatsep
3006 \newskip\intextsep
3007 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
3008 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
3009 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
```

(End of definition for `\floatsep`, `\textfloatsep`, and `\intextsep`.)

`\dblfloatsep` When double-column floats (floating objects that span the whole `\textwidth`) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by `\dblfloatsep` and `\dbltextfloatsep`. They are rubber lengths.

`\dblfloatsep` is the space between adjacent double-column floats placed at the top of the text page.

`\dbltextfloatsep` is the space between the main text and double-column floats at the top of the page.

```
3010 \newskip\dblfloatsep
3011 \newskip\dbltextfloatsep
3012 \setlength\dblfloatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
3013 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

(End of definition for `\dblfloatsep` and `\dbltextfloatsep`.)

Floats on their own page or column

`\@fptop` When floating objects are placed on a separate page or column, called a ‘float page’, the layout of the page is controlled by these parameters, which are rubber lengths.

`\@fpsep` At the top of the page `\@fptop` is inserted; typically this supplies some stretchable whitespace. At the bottom of the page `\@fpbot` is inserted. Between adjacent floats `\@fpsep` is inserted.

These parameters are used for all floating objects on a ‘float page’ in one-column mode, and for single-column floats in two-column mode.

Note that at least one of the two parameters `\@fptop` and `\@fpbot` should contain a `plus ...fil` so as to fill the remaining empty space.

```
3014 \newskip\@fptop
3015 \newskip\@fpsep
3016 \newskip\@fpbot
3017 \setlength\@fptop{0\p@ \@plus 1fil}
3018 \setlength\@fpsep{8\p@ \@plus 2fil}
3019 \setlength\@fpbot{0\p@ \@plus 1fil}
```

(End of definition for `\@fptop`, `\@fpsep`, and `\@fpbot`.)

```

\@dblftop Double-column ‘float pages’ in two-column mode use similar parameters.
\@dblpsep 3020 \newskip\@dblftop
\@dblpbot 3021 \newskip\@dblpsep
3022 \newskip\@dblpbot
3023 \setlength\@dblftop{0\p@ \@plus 1fil}
3024 \setlength\@dblpsep{8\p@ \@plus 2fil}
3025 \setlength\@dblpbot{0\p@ \@plus 1fil}

(End of definition for \@dblftop, \@dblpsep, and \@dblpbot.)

\topfigrule The macros can be used to put in rules between floats and text; whatever they insert
\botfigrule should be vertical mode material which takes up zero space.
\dblfigrule 3026 \let\topfigrule=\relax
3027 \let\botfigrule=\relax
3028 \let\dblfigrule=\relax
3029 \endkernel

(End of definition for \topfigrule, \botfigrule, and \dblfigrule.)

```

File 55

ltagging.dtx

1 <*2ekernel | latexrelease>

2 \ExplSyntaxOn

1 General support for tagged output

\SuspendTagging

\ResumeTagging

\tag_suspend:n

\tag_resume:n

There are places in code where it is important to stop any tagging activities, e.g., when we are doing trial typesetting that it is done several times. In such a case one must tag only the final version that is actually used, otherwise tagging structures are generated which then do not end up in the PDF and confuse the mechanism. For this we have two commands that can be used in packages: `\SuspendTagging` and `\ResumeTagging` (with corresponding L3 programming layer commands). They are available as part of the L^AT_EX kernel, so that they can be safely used in packages whether or not tagging is requested. They all take a string argument that is used for debugging to easily identify why tagging was suspended or restarted, for example, in `tabularx` you find `\SuspendTagging{tabularx}`. By default these four commands do nothing.

The argument is used literally (in `\typeout` messages) without any expansion when debugging is turned on and otherwise it is not used at all. This means it is safe to write something like `\SuspendTagging{\foo}` or even `\SuspendTagging\foo` which means L^AT_EX has to parse only a single token instead of putting a string of characters into the argument. This means a tiny speed improvement but with many such debugging strings...

\NewTaggingSocket

\NewTaggingSocketPlug

\AssignTaggingSocketPlug

Tagging sockets are implemented as normal sockets but with a name that starts with `tagsupport/` and some special conventions how their arguments (if any) are to be interpreted. This means in principle one can use the standard socket commands, which is what we started out with.

However, providing dedicated declaration commands is more convenient and helps in keeping the interfaces clearer and simpler, e.g., `\NewTaggingSocket` not only declares the socket but also automatically sets up necessary plugs and assigns a suitable default plug when tagging is not enabled.

`\NewTaggingSocketPlug` and `\AssignTaggingSocketPlug` on the other hand are mainly syntactic sugar and do nothing more than adding the `tagsupport/` to the socket name behind the scenes.

\UseTaggingSocket

\tag_socket_use:n

\tag_socket_use:nn

\tag_socket_use:nnn

Given that we sometimes have to suspend tagging, it would be fairly inefficient to put different plugs into these sockets whenever that happens. We therefore offer `\UseTaggingSocket` which is like `\UseSocket` except that it expects a socket starting with `tagsupport/` but the socket name is specified without this prefix, i.e.,

`\UseTaggingSocket{foo} → \UseSocket{tagsupport/foo}`

Beside being slightly shorter, the big advantage is that this way we can change `\UseTaggingSocket` to do nothing by switching a boolean instead of changing the plugs of the tagging support sockets back and forth.

Usually, these sockets have (beside the default plug defined for every socket) one additional plug defined and directly assigned. This plug is used when tagging is active. There may be more plugs, e.g., tagging with special debugging or special behavior depending on the class or PDF version etc., but right now it is usually just on or off.

When tagging is suspended they all have the same predefined behavior: The sockets with zero arguments do nothing. The sockets with one argument gobble their argument. The sockets with two arguments will drop their first argument and pass the second unchanged.

It is possible to use the tagging support sockets with `\UseSocket` directly, but in this case the socket remains active if `\SuspendTagging` is in force. There may be reasons for doing that but in general we expect to always use `\UseTaggingSocket`.

`\UseExpandableTaggingSocket` For special cases like in some `\halign` contexts we need a fully expandable version of the command. For these cases, `\UseExpandableTaggingSocket` can be used. To allow being expandable, it does not output any debugging information if `\DebugSocketsOn` is in effect and therefore should be avoided whenever possible.

The L3 programming layer versions `\tag_socket_use_expandable:n`, `\tag_socket_use:n`, `\tag_socket_use:nn`, and `\tag_socket_use:nnn` are slightly more efficient than `\UseTaggingSocket` because they do not have to determine how many arguments the socket takes when disabling it.

`\MathCollectTrue` The tagging of math has to collect/grab the math first. This is not wanted for all uses of `$`. These command allow to control the behavior of the math shift token. Without `\MathCollectFalse` the math tagging code they do nothing. Their behavior with the math tagging code is documented in `latex-lab-math.pdf`

`\MathMLintent` These two commands take two arguments. The first argument is the value of the `\MathMLarg` `intent` or `arg` attribute to be added to the MathML generated by the term in the second argument. By default the commands are no-op and discard the first argument and expand to the second. If `luamml` is loaded via the math tagging code, then these commands are redefined. They could also potentially be used by other `TEX` to MathML converters to control the generated MathML.

`\NewStructureName` Structure elements in a document can use as tag a name from the standard PDF namespaces like `Sect`, `H1` or `Figure` but they can also use new names (which then must be rolemapped to a standard name). The second option is useful for three reasons:

- It looks nicer, if, e.g., a bible uses tag names like `Testament` or `Chapter` or `Book` instead of `Sect`.
- It is possible to formulate additional constraint on such structures in a Schema and thus ensure that there is no `Testament` inside a `Book`, something that can not be done if `Sect` is used everywhere.
- We can provide a uniform LaTeX set of names for tags.

To make it possible to adapt the tag names of a structure in document, the tag name should be stored in a command. These three commands offer an interface to declare, use and reassign such symbolic structure names. `\NewStructureName` takes one argument and declares the internal command, initially its value is `NonStruct`. The expandable command `\UseStructureName` takes one argument and allows to use the stored value. `\AssignStructureRole` takes two arguments. The first is a symbolic structure names, the second a role which should be a simply string allowed as a tag name. When the tagging code is loaded `\AssignStructureRole` is redefined to setup the rolemapping, see the description in `latex-lab-namespace.dtx`. There is also a description of the naming scheme and a list of the already predeclared names.

2 Implementation

`\tag_suspend:n` In the kernel, these commands get dummy definitions so that they can be used without harm in packages. The real definition is used when tagging gets enabled.

`\tag_resume:n`

`\SuspendTagging` 3 `\cs_new_eq:NN \tag_suspend:n \use_none:n`

`\ResumeTagging` 4 `\cs_new_eq:NN \tag_resume:n \use_none:n`

5 `\cs_new_protected:Npn \SuspendTagging #1 { \tag_suspend:n {#1} }`

6 `\cs_new_protected:Npn \ResumeTagging #1 { \tag_resume:n {#1} }`

(End of definition for \tag_suspend:n and others.)

`\NewTaggingSocket` Initialize a new tagging socket and assign it a suitable plug.

```
7 \cs_new_protected:Npn \NewTaggingSocket #1 #2 {
8   \socket_new:nn { tagsupport / #1 } { #2 }
9   \int_case:nnF
10     { #2 }
11     {
12       0 \prg_do_nothing:
13       1 { \socket_assign_plug:nn { tagsupport / #1 } { noop } }
```

Tagging sockets with two arguments use a special **transparent** plug that just passes the second argument. Its already assigned so we only have to alter it.

```
14       2 { \socket_new_plug:nnn { tagsupport / #1 } { transparent }
15           { ##2 }
16           \socket_assign_plug:nn { tagsupport / #1 } { transparent } }
17     }
18     \ERRORnewtaggingsocket % that should get a proper error message
19 }
```

(End of definition for \NewTaggingSocket.)

```
\NewTaggingSocketPlug
\AssignTaggingSocketPlug 20 \cs_new_protected:Npn \NewTaggingSocketPlug #1 {
21   \NewSocketPlug { tagsupport/ #1 }
22 }
23 \cs_new_protected:Npn \AssignTaggingSocketPlug #1 {
24   \AssignSocketPlug { tagsupport/ #1 }
25 }
```

(End of definition for \NewTaggingSocketPlug and \AssignTaggingSocketPlug.)

`\tag_socket_use:n` Again this is not the final definition for the kernel; it is just a version to get going while some parts of the kernel support are still missing.

`\tag_socket_use:nn`

`\tag_socket_use:nnn`

```
26 \prg_new_conditional:Npnn \tag_if_active: { p , T , TF , F }
27 { \prg_return_false: }
```

`\tag_socket_use_expandable:n`

`\UseTaggingSocket`

`\UseExpandableTaggingSocket`

Dummy definitions in the kernel. These definitions will get updated in **tagpdf**. The default in the kernel is simply to get rid of the first argument, while the second argument is preserved if present:

```
28 \cs_new:Npn \tag_socket_use_expandable:n #1 { }
29 \cs_new_protected:Npn \tag_socket_use:n #1 { }
30 \cs_new_protected:Npn \tag_socket_use:nn #1#2 { }
31 \cs_new_protected:Npn \tag_socket_use:nnn #1#2#3 { #3 }
32 \cs_new_protected:Npn \UseTaggingSocket #1 { }
```

```

33 \int_case:nnF
34 { \int_use:c { c__socket_taggsupport/#1_args_int } }
35 {
36 0 \prg_do_nothing:
37 1 \use_none:n
38 2 \use_ii:nn

```

We do not expect tagging sockets with more than one or two arguments, so for now we only provide those.

```

39 }
40 \ERRORusetaggingsocket % that should get a proper error message
41 }

```

The same as an expandable command:

```

42 \cs_new:Npn \UseExpandableTaggingSocket #1 {
43 \int_case:nnF
44 { \int_use:c { c__socket_taggsupport/#1_args_int } }
45 {
46 0 \prg_do_nothing:
47 1 \use_none:n
48 2 \use_ii:nn
49 }
50 \ERRORusetaggingsocket % that should get a proper error message
51 }

```

(End of definition for \tag_socket_use:n and others.)

2.1 Math collection

The documentation is in latex-lab-math.

```

\MathCollectTrue
\MathCollectFalse
52 \cs_new_protected:Npn \MathCollectTrue{}
53 \cs_new_protected:Npn \MathCollectFalse{}

```

(End of definition for \MathCollectTrue and \MathCollectFalse.)

2.2 Tagging sockets

This collects tagging sockets that should be generally available so that they can also be used even if the tagging code is not loaded.

2.3 Generic sockets

These sockets are used in various places and should not be reassigned globally.

mc (*tag socket*) At first a generic socket to surround content with the mc-commands. The first argument can be used to pass options like **artifact**

```

54 \NewTaggingSocket{mc}{2}
55 \NewTaggingSocketPlug{mc}{kernel}
56 {
57 \tag_mc_begin:n {#1}
58 #2
59 \tag_mc_end:
60 }
61 \AssignTaggingSocketPlug{mc}{kernel}

```

struct-mc (*tag socket*) A socket to surround content with a structure and an mc-command. With the first argument the tag and other options can be set.

```

62 \NewTaggingSocket{struct-mc}{2}
63 \NewTaggingSocketPlug{struct-mc}{kernel}
64 {
65   \tag_struct_begin:n{#1}
66   \tag_mc_begin:n {}
67   #2
68   \tag_mc_end:
69   \tag_struct_end:
70 }
71 \AssignTaggingSocketPlug{struct-mc}{kernel}

```

struct/begin (*tag socket*) Sockets to start and end a structure. With the first argument the tag and other options
struct/end (*tag socket*) can be set.

```

72 \NewTaggingSocket{struct/begin}{1}
73 \NewTaggingSocket{struct/end}{0}
74 \NewTaggingSocketPlug{struct/begin}{kernel}
75 {
76   \tag_struct_begin:n{#1}
77 }
78 \NewTaggingSocketPlug{struct/end}{kernel}
79 {
80   \tag_struct_end:
81 }
82 \AssignTaggingSocketPlug{struct/begin}{kernel}
83 \AssignTaggingSocketPlug{struct/end}{kernel}

```

inline/begin (*tag socket*) These sockets can be used to insert small structures (often a Span with some attributes)
inline/end (*tag socket*) inside a paragraph: They close and reopen MC's. With the argument structure name and attributes can be passed on.

```

84 \NewTaggingSocket{inline/begin}{1}
85 \NewTaggingSocketPlug{inline/begin}{kernel}
86 {
87   \tag_mc_end_push:
88   \tag_struct_begin:n{#1}
89   \tag_mc_begin:n{#1}
90 }
91 \AssignTaggingSocketPlug{inline/begin}{kernel}
92 \NewTaggingSocket{inline/end}{0}
93 \NewTaggingSocketPlug{inline/end}{kernel}
94 {
95   \tag_mc_end:
96   \tag_struct_end:
97   \tag_mc_begin_pop:n{#1}
98 }
99 \AssignTaggingSocketPlug{inline/end}{kernel}

```

2.3.1 Tagging support for paragraph setup

Paragraphs are tagged through the code in the para/hooks. This code is sometimes adjusted, e.g. to produce a “flattened” paragraph or to use a different tag. Sockets related to such code parts are collected here.

`\l__tag_block_flattened_level_int` The block code needs to know if they are nested blockenvs inside a flattened environment. For this it uses a counter. Inside some contexts, e.g. at the begin of a minipage or a footnote this counter must be reset. We therefore define the counter here so that we can use it in the following socket.

```
100 \int_new:N \l__tag_block_flattened_level_int
```

(End of definition for `\l__tag_block_flattened_level_int`.)

`para/on` (*tag socket*) These sockets allow paragraph tagging to be enabled/disabled.

```
para/off (tag socket) 101 \NewTaggingSocket{para/on}{0}
102 \NewTaggingSocketPlug{para/on}{kernel}{\bool_set_true:N \l__tag_para_bool}
103 \NewTaggingSocket{para/off}{0}
104 \NewTaggingSocketPlug{para/off}{kernel}{\bool_set_false:N \l__tag_para_bool}
105 \AssignTaggingSocketPlug{para/on}{kernel}
106 \AssignTaggingSocketPlug{para/off}{kernel}
```

`para/restore` (*tag socket*) This socket restores the para related settings to their default. It should be used in places where “normal” paragraph tagging must be ensured, for example at the begin of a footnote.

```
107 \NewTaggingSocket{para/restore}{0}
```

`default` (*tag plug*)

```
108 \NewTaggingSocketPlug{para/restore}{default}
109 {
110   \AssignStructureRole{para/semantic}{\c_tag_para_semantic_tl}
111   \AssignStructureRole{para/textblock}{\c_tag_para_textblock_tl}
112   \bool_set_false:N \l__tag_para_flattened_bool
113   \int_zero:N \l__tag_block_flattened_level_int
114   \bool_set_true:N \l__tag_para_bool
115 }
116 \AssignTaggingSocketPlug{para/restore}{default}
```

`para/begin` (*tag socket*) These sockets were previously defined in tagpdf. There are the main sockets to handle

`para/end` (*tag socket*) the paragraph tagging when it is active (i.e., when `para/on` or `para/restore` was executed earlier).

```
117 \NewTaggingSocket{para/begin}{0}
118 \NewTaggingSocket{para/end}{0}
```

Until tagpdf is updated we need to provide the plain and block plugs

```
119 \NewTaggingSocketPlug{para/begin}{plain}{}
120 \NewTaggingSocketPlug{para/end}{plain}{}
121 \NewTaggingSocketPlug{para/begin}{block}{}
122 \NewTaggingSocketPlug{para/end}{block}{}
123
```

`para/semantic/begin` (*tag socket*) These sockets handle the begin and end structure. These are stored in sockets of their own to be able to disable them globally.

```
124 \NewTaggingSocket{para/semantic/begin}{1}
125 \NewTaggingSocket{para/semantic/end}{1}
```

(para/semantic/begin) (*tag plug*) This tagging socket adds a structure. The argument allows to add additional commands like the command that puts the structure number of this command on a stack.

```

126 \tl_new:N \l__tag_para_attr_class_tl
127 \NewTaggingSocketPlug{para/semantic/begin}{kernel}
128 {
129   \__tag_gincr_para_main_begin_int:
130   \tag_struct_begin:n
131   {
132     tag=\UseStructureName{para/semantic},
133     attribute-class=\l__tag_para_main_attr_class_tl,
134   }
135   #1
136 }

```

1 (para/semantic/end) (*tag plug*) This socket should be used if a -structure is closed. The argument allows e.g. to add debug info.

```

137 \NewTaggingSocketPlug{para/semantic/end}{kernel}
138 {
139   #1
140   \__tag_gincr_para_main_end_int:
141   \tag_struct_end:
142 }

```

And now we assign these plugs:

```

143 \AssignTaggingSocketPlug{para/semantic/begin}{kernel}
144 \AssignTaggingSocketPlug{para/semantic/end}{kernel}

```

para/textblock/begin (*tag socket*) These sockets handle the begin and end structure. These are stored in sockets of their own to be able to reuse them in places where only a textblock is needed.

```

145 \NewTaggingSocket{para/textblock/begin}{1}
146 \NewTaggingSocket{para/textblock/end}{0}

```

para/textblock/begin (*tag plug*) This tagging socket opens a structure and mc. It is used below in the para/begin tagging socket. The argument is used to adapt the debugging label.

```

147 \NewTaggingSocketPlug{para/textblock/begin}{kernel}
148 {
149   \__tag_gincr_para_begin_int:
150   \__tag_check_typeout_v:n {==>~increment~ P \on@line }
151   \tag_struct_begin:n
152   {
153     tag=\UseStructureName{para/textblock}
154     ,attribute-class=\l__tag_para_attr_class_tl
155   }
156   \__tag_check_para_begin_show:nn {green}{#1}
157   \tag_mc_begin:n {}
158 }

```

(para/textblock/end) (*tag plug*) This socket closes a -mc and structure. It is used below in the para/end tagging socket.

TeX automatically removes a final space at the end of the paragraph text but this internal fails if we add the tagging code after that final space, so we have to remove any space manually before adding that code.

```

159 \NewTaggingSocketPlug{para/textblock/end}{kernel}

```

```

160 {
161   \unskip
162   \__tag_gincr_para_end_int:
163   \__tag_check_typeout_v:n {==>~increment~ /P \on@line }
164   \tag_mc_end:
165   \__tag_check_para_end_show:nn {red}{ }
166   \tag_struct_end:
167 }

```

And now we assign these plugs:

```

168 \AssignTaggingSocketPlug{para/textblock/begin}{kernel}
169 \AssignTaggingSocketPlug{para/textblock/end}{kernel}

```

kernel (para/begin) (tag plug) This plug is similar to the block socket currently defined in tagpdf and should overwrite it.

```

170 \NewTaggingSocketPlug{para/begin}{kernel}
171 {
172   \bool_if:NT \l__tag_para_bool
173   {
174     \legacy_if:nF { @inlabel }
175     {
176       \__tag_check_typeout_v:n
177       {==>~ @endpe = \legacy_if:nTF { @endpe }{true}{false} \on@line }
178       \legacy_if:nF { @endpe }
179       {
180         \bool_if:NF \l__tag_para_flattened_bool
181         {
182           \UseTaggingSocket{para/semantic/begin}
183           { }
184         }
185       }
186       \UseTaggingSocket{para/textblock/begin}{NP-}
187     }
188   }
189 }

```

kernel (para/end) (tag plug) This socket is used at the end of paragraphs.

```

190 \NewTaggingSocketPlug{para/end}{kernel}
191 {
192   \bool_if:NT \l__tag_para_bool
193   {
194     \UseTaggingSocket{para/textblock/end}
195     \bool_if:NF \l__tag_para_flattened_bool
196     {
197       \UseTaggingSocket{para/semantic/end}{ }
198     }
199   }
200 }

```

As tagpdf is currently assigning the sockets after the block code we assign if even later (for now):

```

201 \AddToHook{package/latex-lab-testphase-sec/after}
202 {

```

```

203 \AssignTaggingSocketPlug{para/begin}{kernel}
204 \AssignTaggingSocketPlug{para/end}{kernel}
205 }

```

2.3.2 Tagging socket for targets

refstepcounter (*tag socket*) When tagging is active we want to track the current structure number when targets are set. This will be mostly used in `\refstepcounter` but also if targets are set manually.

```

206 \NewTaggingSocket{recordtarget}{0}

```

kernel (*recordtarget*) (*tag plug*)

```

207 %
208 \NewTaggingSocketPlug{recordtarget}{kernel}
209 {
210   \tag_struct_recordtarget:
211 }
212 \AssignTaggingSocketPlug{recordtarget}{kernel}
213 \ExplSyntaxOff

```

2.3.3 Tagging Sockets for boxes

```

214 \NewTaggingSocket{minipage/before}{0}
215 \NewTaggingSocket{minipage/after}{0}
216 \NewTaggingSocket{parbox/before}{0}
217 \NewTaggingSocket{parbox/after}{0}

```

2.3.4 Tagging Sockets for lists and blocks

block/list/label (*tag socket*) A tagging socket around the label in a list.

```

218 \NewTaggingSocket{block/list/label}{2}

```

block/recipe (*tag socket*) A tagging socket to set the tagging recipe.

```

219 \NewTaggingSocket{block/recipe}{1}

```

2.3.5 Tagging sockets for headings

Tagging headings is quite tricky. The sockets (and plugs) used by the kernel definitions often expect specially formatted arguments. For this reason some of them may not be suitable in other classes or packages. In that case additional sockets (or plugs) might be required.

sec/begin (*tag socket*) These two sockets should surround the whole section by a `Sect` structure. They should be suited also for classes and packages. The begin socket takes two brace groups as argument: the level and key for the structure (typically `tag=somename`), the end socket takes as argument a level.

```

220 \NewTaggingSocket{sec/begin}{1}
221 \NewTaggingSocket{sec/end}{1}

```

sec/title/begin (*tag socket*) These two sockets are used around display heading where the number is on a line of its own. The argument of the begin socket consists of two brace groups containing the level and the title.

```

222 \NewTaggingSocket{sec/title/begin}{1}
223 \NewTaggingSocket{sec/title/end}{0}

```


`sec/title/hang` (*tag socket*) This socket is used in headings with hanging number.

224 `\NewTaggingSocket{sec/title/hang}{2}`

`sec/title/init` (*tag socket*) These sockets are used by run-in heading to initialize tagging and to split heading and
`sec/title/split` (*tag socket*) following text and tag the title.

`/title/runin/number` (*tag socket*) 225 `\NewTaggingSocket{sec/title/init}{1}`
226 `\NewTaggingSocket{sec/title/split}{0}`
227 `\NewTaggingSocket{sec/title/runin/number}{2}`

`sec/title/number` (*tag socket*) This is a socket to tag the number. It takes two arguments: the level and the content.

228 `\NewTaggingSocket{sec/title/number}{2}`

2.3.6 Tagging sockets for toc

`contentsline/before` (*tag socket*) Tagging sockets at the begin and end of contentsline. They receive *all* contentsline
`/contentsline/after` (*tag socket*) arguments as one argument in four brace groups. The socket code should then use the
parts it needs.

229 `\NewTaggingSocket{toc/contentsline/before}{1}`
230 `\NewTaggingSocket{toc/contentsline/after}{1}`

`toc/starttoc/before` (*tag socket*) Tagging sockets for the begin and end of start of `\@starttoc`. They take one argument,
`toc/starttoc/after` (*tag socket*) the extension.

231 `\NewTaggingSocket{toc/starttoc/before}{1}`
232 `\NewTaggingSocket{toc/starttoc/after}{1}`

`toc/leaders/before` (*tag socket*) Tagging sockets to make the dot leaders an artifact. They do not take an argument.

`toc/leaders/after` (*tag socket*) 233 `\NewTaggingSocket{toc/leaders/before}{0}`
234 `\NewTaggingSocket{toc/leaders/after}{0}`

2.3.7 Tagging support for marginpar

`marginpar/begin` (*tag socket*)

`marginpar/end` (*tag socket*) 235 `\NewTaggingSocket{marginpar/begin}{0}`
236 `\NewTaggingSocket{marginpar/end}{0}`

2.3.8 Tagging support for table/tabular packages

The code uses a number of sockets to inject the tagging commands. These can be easily set to a noop-plug in case the automated tagging is not wanted.

`tbl/cell/begin` (*tag socket*) At first sockets for the begin and end of cells and table rows:

`tbl/cell/end` (*tag socket*) 237 `\NewTaggingSocket{tbl/cell/begin}{0}`
`tbl/pcell/end` (*tag socket*) 238 `\NewTaggingSocket{tbl/cell/end}{0}`
`tbl/pcell/end` (*tag socket*) 239 `\NewTaggingSocket{tbl/row/begin}{0}`
`tbl/row/begin` (*tag socket*) 240 `\NewTaggingSocket{tbl/row/end}{0}`
`tbl/row/end` (*tag socket*)

Multi-line cells have their own sockets (as they start out in vertical mode and need different treatment).

241 `\NewTaggingSocket{tbl/pcell/begin}{0}`
242 `\NewTaggingSocket{tbl/pcell/end}{0}`

`tbl/init` (*tag socket*) This socket should be at the begin of the table, inside a group. It is used for settings such as disabling para-tagging inside the table. This socket can perhaps be merged later into the begin-sockets.

243 `\NewTaggingSocket{tbl/init}{0}`

`tbl/init/celldata` (*tag socket*) This socket is used in `\tbl_init_cell_data_for_table`, the command that stores and initialize cell data to handle nested tables. It can be used to restore similar tagging related values

244 `\NewTaggingSocket{tbl/init/celldata}{0}`

`tbl/finalize` (*tag socket*) To fine tune the structure (change cells to header cells, remove unwanted structures, move a foot to the end, etc.). We also need a socket that is executed at the end of the table but *before* all the variables are restored to the outer or default values. The code in the socket can make assignments, but probably shouldn't do typesetting and not write whatsits.

245 `\NewTaggingSocket{tbl/finalize}{0}`

`tbl/restore/celldata` (*tag socket*) This socket is used in `\tbl_restore_outer_cell_data:`, the command that restores cell data when quitting a nested table. It can be used to restore similar tagging related values

246 `\NewTaggingSocket{tbl/restore/celldata}{0}`

`tbl/colspan` (*tag socket*) This socket is used to manage spanning cells, e.g., a `\multicolumn`. It expects one argument (the number of cells spanned) and if tagging is enabled set appropriate tag attributes in the background. We probably need a similar socket for row spans eventually.

247 `\NewTaggingSocket{tbl/colspan}{1}`

`tbl/hmode/begin` (*tag socket*) These sockets are used in the begin and end code of environments, to allow a fast enabling and disabling of the tagging. We distinguish between tables that can be used inside paragraphs and standalone tables such as `longtable` that are always in vertical mode.

`tbl/hmode/end` (*tag socket*)

`tbl/vmode/begin` (*tag socket*)

`tbl/vmode/end` (*tag socket*)

248 `\NewTaggingSocket{tbl/hmode/begin}{0}`

249 `\NewTaggingSocket{tbl/hmode/end}{0}`

250 `\NewTaggingSocket{tbl/vmode/begin}{0}`

251 `\NewTaggingSocket{tbl/vmode/end}{0}`

`tbl/longtable/init` (*tag socket*) `longtable` needs its own sockets to fine tune the structure. Simply switching the plug in the previous socket interferes with enabling/disabling the tagging.

252 `\NewTaggingSocket{tbl/longtable/init}{0}`

253 `\NewTaggingSocket{tbl/longtable/finalize}{0}`

`tbl/longtable/head` (*tag socket*) Header and footer boxes need special handling because they are repeatedly used.

`tbl/longtable/foot` (*tag socket*)

254 `\NewTaggingSocket{tbl/longtable/head}{0}`

255 `\NewTaggingSocket{tbl/longtable/foot}{0}`

`tbl/leaders/begin` (*tag socket*) Sockets around leaders such as rules or dotted lines, that should be tagged as artifacts, used, for example, in `\cline`.

`tbl/leaders/end` (*tag socket*)

256 `\NewTaggingSocket{tbl/leaders/begin}{0}`

257 `\NewTaggingSocket{tbl/leaders/end}{0}`

2.3.9 Tagging Support for floats

<code>float/hmode/begin</code> (<i>tag socket</i>)	These sockets are used if the float is called in hmode.
<code>float/hmode/end</code> (<i>tag socket</i>)	<div>258 <code>\NewTaggingSocket{float/hmode/begin}{0}</code></div> <div>259 <code>\NewTaggingSocket{float/hmode/end}{0}</code></div>
<code>float/begin</code> (<i>tag socket</i>)	These sockets start and stop the float structure.
<code>float/end</code> (<i>tag socket</i>)	<div>260 <code>\NewTaggingSocket{float/begin}{0}</code></div> <div>261 <code>\NewTaggingSocket{float/end}{0}</code></div>
<code>caption/begin</code> (<i>tag socket</i>)	These sockets are used in <code>\@makecaption</code> . They open and close the <code>Caption</code> structure.
<code>caption/end</code> (<i>tag socket</i>)	<div>262 <code>\NewTaggingSocket{caption/begin}{1}</code></div> <div>263 <code>\NewTaggingSocket{caption/end}{0}</code></div>
<code>caption/label/begin</code> (<i>tag socket</i>)	These sockets are used in <code>\@makecaption</code> around the label. Their default plugs ensure
<code>caption/label/end</code> (<i>tag socket</i>)	that the label is outside the paragraph and that the rest of the caption uses flattened para mode. If the caption is not in a hbox, the <code>para/begin</code> socket should follow to properly start the paragraph.
	<div>264 <code>\NewTaggingSocket{caption/label/begin}{0}</code></div> <div>265 <code>\NewTaggingSocket{caption/label/end}{0}</code></div>

2.4 Tagging support for output routines

<code>build/page/header</code> (<i>tag socket</i>)	These sockets receive the formatted running header/footer in its second argument (the first is not used) following the convention of tagging sockets, i.e., only the second argument is processed if tagging is not active.
<code>build/page/footer</code> (<i>tag socket</i>)	<div>266 <code>\NewTaggingSocket{build/page/header}{2}</code></div> <div>267 <code>\NewTaggingSocket{build/page/footer}{2}</code></div>
<code>ld/column/outputbox</code> (<i>tag socket</i>)	This socket is used to add any missing tagging structures to the <code>\@outputbox</code> box, if necessary.
	268 <code>\NewTaggingSocket{build/column/outputbox}{0}</code>
<code>build/column/footins</code> (<i>tag socket</i>)	This socket is used to add any missing tagging structures to the <code>\footins</code> box, if necessary.
	269 <code>\NewTaggingSocket{build/column/footins}{0}</code>
<code>page@sofar</code> (<i>tag socket</i>)	This socket is declared and used in the multicol output routines. Only listed for reference.
	270 <code>%\NewTaggingSocket{page@sofar}{0}</code>

2.5 Tagging support for math

2.5.1 General sockets

The following sockets are the main math sockets.

```
271 \NewTaggingSocket{math/inline/begin}{0}  
272 \NewTaggingSocket{math/inline/end}{0}  
273 \NewTaggingSocket{math/inline/formula/begin}{2}  
274 \NewTaggingSocket{math/inline/formula/end}{0}  
275 \NewTaggingSocket{math/display/begin}{0}  
276 \NewTaggingSocket{math/display/end}{0}  
277 \NewTaggingSocket{math/display/formula/begin}{2}  
278 \NewTaggingSocket{math/display/formula/end}{0}  
279 \NewTaggingSocket{math/display/tag/begin}{0}  
280 \NewTaggingSocket{math/display/tag/end}{0}
```

2.5.2 Sockets specific for luamml

Save sockets These sockets are wrappers around the `\luamml_save:...` commands. They take an argument which should contain the argument of the save command.

`math/luamml/save/nn` (*tag socket*) The argument should contain the two arguments of the command.

```
281 \NewTaggingSocket{math/luamml/save/nn}{1}
```

`math/luamml/save/nNn` (*tag socket*) The argument should contain the three arguments of the command.

```
282 \NewTaggingSocket{math/luamml/save/nNn}{1}
```

Socket to annotate

`luamml/annotate/false` (*tag socket*) These socket can be used for content that should be annotated with `core=false`

```
283 \NewTaggingSocket{math/luamml/annotate/false}{2}
```

Array sockets These sockets will be used in `array` to add luamml support to the array environment.

`h/luamml/array/save` (*tag socket*) This socket will be used in `\endarray`. The plug is set by luamml.

```
284 \NewTaggingSocket{math/luamml/array/save}{0}
```

`luamml/array/finalize` (*tag socket*) This socket will be used in `\endarray`. The plug is set by luamml.

```
285 \NewTaggingSocket{math/luamml/array/finalize}{0}
```

`luamml/array/initcol` (*tag socket*) This socket will be used in `\@classz`. The plug is set by luamml.

```
286 \NewTaggingSocket{math/luamml/array/initcol}{0}
```

`l/array/finalizecol` (*tag socket*) This socket will be used in `\@classz`. The plug is set by luamml. The argument sets the type of the column.

```
287 \NewTaggingSocket{math/luamml/array/finalizecol}{1}
```

Alignment environments Multiline environments like `align`, `multline` or `gather` are tagged as `mtable`.

`/mtable/finalizecol` (*tag socket*) This socket is used at the end of alignment cells and adds them to the row. The argument passes a type like `last` or `box`.

288 `\NewTaggingSocket{math/luamml/mtable/finalizecol}{1}`

`luamml/mtable/finalize` (*tag socket*) This socket is used at the end of alignment environment to finalize the `mtable` code. It should be used normally with `\UseExpandableTaggingSocket`.

289 `\NewTaggingSocket{math/luamml/mtable/finalize}{1}`

`luamml/mtable/aligncol` (*tag socket*) This socket is used in `multline` to add attributes describing the alignment to the left and right. It takes an argument, the alignment.

290 `\NewTaggingSocket{math/luamml/mtable/aligncol}{1}`

`table/innertable/save` (*tag socket*) This socket is used in `\endaligned` to save the table. It takes no argument.

291 `\NewTaggingSocket{math/luamml/mtable/innertable/save}{0}`

`table/smallmatrix/save` (*tag socket*) This socket is used in `\endsmallmatrix` to save the table. It takes no argument. TODO: Check if this socket and the `innertable` socket can/should be merged into a more generic version.

292 `\NewTaggingSocket{math/luamml/mtable/smallmatrix/save}{0}`

`table/innertable/finalize` (*tag socket*) This socket is used e.g. in `\endsmallmatrix` and `\gathered` to finalize the table. It takes no argument.

293 `\NewTaggingSocket{math/luamml/mtable/innertable/finalize}{0}`

`luamml/mtable/tag/save` (*tag socket*) This socket is used to save a tag for later use. has been save before.

294 `\NewTaggingSocket{math/luamml/mtable/tag/save}{0}`

`luamml/mtable/tag/set` (*tag socket*) This socket should be used when a tag is placed. It inserts a tag that has been save before.

295 `\NewTaggingSocket{math/luamml/mtable/tag/set}{0}`

mbox socket

`math/luamml/hbox` (*tag socket*) This socket is used around `\hbox` inside an `\mbox`, `\makebox` and `\text` and annotates the content if the `\hbox` is used inside `math`. The real plug is set by `luamml` but a default plug is defined here so that the socket can also be used if `luamml` is not used.

296 `\NewTaggingSocket{math/luamml/hbox}{2}`

math phantom sockets

`math/luamml/finph@nt` (*tag socket*) This socket handles the annotation of `\finph@nt`

297 `\NewTaggingSocket{math/luamml/finph@nt}{2}`

`math/luamml/finph@nt` (*tag socket*) This socket handles the annotation of `\finsm@sh`

298 `\NewTaggingSocket{math/luamml/finsm@sh}{2}`

Artifact root sign

`\math/luamml/artifact` (*tag socket*) Unicode characters like a root sign should be marked as artifacts to avoid duplication, e.g., in derivation if mathml structure elements are used that imply the meaning.

```
299 \NewTaggingSocket{math/luamml/artifact}{0}
```

2.6 MathML intent attributes

`\MathMLintent` Stub definitions here to allow these commands to be used in packages whether or not tagging is enabled.

`\MathMLarg`

```
300 \ExplSyntaxOn
301 \cs_new_protected:Npn\MathMLintent#1#2{#2}
302 \cs_new_protected:Npn\MathMLarg#1#2{#2}
303 \ExplSyntaxOff
```

(End of definition for `\MathMLintent` and `\MathMLarg`.)

2.7 Symbolic structure names

`\NewStructureName` Stub definitions here to allow these commands to be used in packages whether or not the tagging support code is loaded.

`\UseStructureName`

`\AssignStructureRole`

```
304 <@@=tag>
305 \ExplSyntaxOn
306 \cs_new_protected:Npn\NewStructureName#1
307 {
308   \tl_new:c {l__tag_name_#1_tl}
309   \tl_set:cn {l__tag_name_#1_tl}{NonStruct}
310 }
311 \cs_new:Npn\UseStructureName#1
312 {
313   \cs:w l__tag_name_#1_tl\cs_end:
314 }
315 \cs_new_protected:Npn\AssignStructureRole#1#2
316 {
317   \tl_set:cn { l__tag_name_#1_tl }{#2}
318 }
319 \ExplSyntaxOff
```

(End of definition for `\NewStructureName`, `\UseStructureName`, and `\AssignStructureRole`.)

3 For lttab.dtx parked here for now

```
320 <@@=tbl>
321 \ExplSyntaxOn
```

3.1 Variables for row, column and span counting

This part needs a decision on names for various integer registers as well as a decision if those should be also made available for L^AT_EX 2_ε-style packages in form of 2e names and or as non-internals for the L3 programming layer.

At the moment they are all internal but this probably has to change.

`\g__tbl_col_int` `\g__tbl_row_int` holds the current row number in the table. The value 0 means we haven't yet processed the table preamble (or in case of longtable are just in front of the next chunk to be processed). It is incremented by every `\cr` including the one ending the table preamble.

TODO: due to the gymnastics needed inside the longtable code the row counter is directly exposed there rather than hidden by interfaces. This needs changing when it is decided how to manage these counters.

`\g__tbl_col_int` holds the current column number. The value 0 means we have not yet started the table or just finished a table row (with `\` typically); any other positive value means we are currently typesetting a cell in that column in some row (denoted by the `\g__tbl_row_int`).

In a `\multicolumn` it holds the column number of the first spanned column and `\g__tbl_span_tl` the info how many cells are spanned.

`\g__tbl_span_tl` is normally 1 except in a `\multicolumn` cell.

```

322 \int_new:N \g__tbl_col_int
323 \int_new:N \g__tbl_row_int
324 \tl_new:N \g__tbl_span_tl
325 \tl_new:N \g__tbl_table_cols_tl
326
327 \tl_gset:Nn \g__tbl_span_tl {1}
328 \tl_gset:Nn \g__tbl_table_cols_tl {0} % indicates outer level

```

(End of definition for `\g__tbl_col_int` and others.)

`\l__tbl_saved_col_tl` `\l__tbl_saved_row_tl` `\l__tbl_saved_span_tl` `\l__tbl_saved_table_cols_tl` Saving the outer values if we are nesting tables is necessary (as the above variables are globally altered). For this we always use token lists because they don't change and we do not need to blow additional integer registers.

```

329 \tl_new:N \l__tbl_saved_col_tl
330 \tl_new:N \l__tbl_saved_row_tl
331 \tl_new:N \l__tbl_saved_span_tl
332 \tl_new:N \l__tbl_saved_table_cols_tl
333
334 \tl_set:Nn \l__tbl_saved_col_tl{0}
335 \tl_set:Nn \l__tbl_saved_row_tl{0}
336 \tl_set:Nn \l__tbl_saved_span_tl{1}
337 \tl_set:Nn \l__tbl_saved_table_cols_tl{0} % indicates outer level

```

(End of definition for `\l__tbl_saved_col_tl` and others.)

`\g__tbl_missingcells_int` This will contain the number of missing cells in a row:

```

338 \int_new:N \g__tbl_missing_cells_int

```

(End of definition for `\g__tbl_missingcells_int`.)

3.2 Tracing/debugging

`\DebugTablesOn`
`\DebugTablesOff`

```

339 \def\DebugTablesOn{
340   \cs_set_eq:NN \__tbl_trace:n \typeout
341 }
342 \def\DebugTablesOff{
343   \cs_set_eq:NN \__tbl_trace:n \use_none:n
344 }

```

```
345 \cs_new_eq:NN \__tbl_trace:n \use_none:n
```

(End of definition for `\DebugTablesOn` and `\DebugTablesOff`.)

3.3 Interface commands

All interface commands for the cell number determination have to be public on some level because they are needed in other packages as well, e.g., `longtable`. We may or may not also want to provide 2e style names for them.

`\tbl_update_cell_data:` Updating cell data in columns after the first means we have to increment the `\g__tbl_col_int` by the span count of the previous cell (in case it was a `\multicolumn`) and then reset the `\g__tbl_span_tl` to one (as the default).

```
346 \cs_new_protected:Npn \tbl_update_cell_data: {
347     \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
348     \tl_gset:Nn \g__tbl_span_tl {1}
349 }
```

(End of definition for `\tbl_update_cell_data:`.)

`\tbl_count_table_cols:` Current implementation of `\@mkpream` uses the scratch counter `\count@` to keep track of the number of toks registers it needs (2 per column), but this can't be used as it counts also insertions made with `!{}` and `@{}`. So similar as does `longtable` for `\LT@cols` we count the numbers of ampersands instead.

```
350 \cs_new:Npn \tbl_count_table_cols: {
351     \seq_set_split:NnV\l__tbl_tmpa_seq {&} \@preamble
352     \tl_gset:Nn \g__tbl_table_cols_tl { \seq_count:N \l__tbl_tmpa_seq }
353     \__tbl_trace:n { ==>~ Table~ has~ \g__tbl_table_cols_tl \space columns }
354 }
```

(End of definition for `\tbl_count_table_cols:`.)

`\l__tbl_tmpa_seq`

```
355 \seq_new:N \l__tbl_tmpa_seq
```

(End of definition for `\l__tbl_tmpa_seq`.)

`\tbl_count_missing_cells:n` We might have the situation that some table package has not implemented the `\tbl_count_table_cols:` in which case `\g__tbl_table_cols_tl` would always be zero and we would get an error below when we try to determine the missing cells, so bypass that calculation if we aren't doing tagging (there the packages should have the proper code added). Recall that this is code, that is called by `\` and an old table package might rely on whatever the L^AT_EX kernel offers here.

```
356 \cs_new:Npn \tbl_count_missing_cells:n #1 {
357     \tag_if_active:T {
358         \int_compare:nNnT \g__tbl_col_int > 0
359         {
360             \int_gset:Nn \g__tbl_missing_cells_int
361             {
362                 \g__tbl_table_cols_tl
363                 - \g__tbl_col_int
364                 - \g__tbl_span_tl
365                 + 1
366             }

```



```

367         \int_compare:nNnT \g__tbl_missing_cells_int < 0 \ERRORmissingcells % should not happen
368         \__tbl_trace:n{==>~
369             (#1)~
370             This~ row~ needs~
371             \int_use:N \g__tbl_missing_cells_int \space
372             additional~ cell(s)
373         }
374     }
375 }
376 }

```

(End of definition for \tbl_count_missing_cells:n.)

\tbl_init_cell_data_for_table:

```

377 \cs_new_protected:Npn \tbl_init_cell_data_for_table: {
378     \tl_set:No \l__tbl_saved_col_tl {\int_use:N \g__tbl_col_int }
379     \tl_set:No \l__tbl_saved_row_tl {\int_use:N \g__tbl_row_int }
380     \tl_set_eq:NN \l__tbl_saved_table_cols_tl \g__tbl_table_cols_tl
381     \tl_set_eq:NN \l__tbl_saved_span_tl \g__tbl_span_tl
382     %
383     \__tbl_trace:n { ==>~ saved~cell~data:~
384         \l__tbl_saved_row_tl,
385         \l__tbl_saved_col_tl,
386         \l__tbl_saved_span_tl \space
387         (
388             \int_compare:nNnTF \l__tbl_saved_table_cols_tl = 0
389                 { outer~ level }
390                 { max:~ \l__tbl_saved_table_cols_tl }
391         )
392     }

```

Tagging has to initialize cell data too.

```

393     \UseTaggingSocket{tbl/init/celldata}

```

These are the initial values when starting a table:

```

394     \int_gzero:N \g__tbl_row_int
395     \int_gzero:N \g__tbl_col_int
396     \tl_gset:Nn \g__tbl_span_tl {1}
397 }

```

(End of definition for \tbl_init_cell_data_for_table:.)

\tbl_update_cell_data_for_next_row:

```

398 \cs_new_protected:Npn \tbl_update_cell_data_for_next_row: {
399     \int_gincr:N \g__tbl_row_int           % this row about to start
400     \int_gzero:N \g__tbl_col_int          % we are before first col
401 }

```

(End of definition for \tbl_update_cell_data_for_next_row:.)

\tbl_init_cell_data_for_row: If we start processing a cell in the first column we set \g__tbl_col_int to 1 as we are no longer "at" but "in" the first column. We also set \g__tbl_span_tl to its default value (not spanning cells).

```

402 \cs_new_protected:Npn \tbl_init_cell_data_for_row: {
403     \tl_gset:Nn \g__tbl_col_int {1}
404     \tl_gset:Nn \g__tbl_span_tl {1}
405 }

```

(End of definition for \tbl_init_cell_data_for_row:.)

\tbl_if_row_was_started:T We use \g__tbl_col_int equal zero to indicate that we are just after a TR (i.e.n between rows or at the very beginning of the table). Using the row count is not so good as longtable
\tbl_if_row_was_started:TF may split the table in chunks.

These conditionals have to be expandable (i.e., unprotected) as they are sometimes executed when T_EX is scanning inside a table.

```
406 \cs_new:Npn \tbl_if_row_was_started:T {
407     \int_compare:nNnT \g__tbl_col_int > 0
408 }
409 \cs_new:Npn \tbl_if_row_was_started:TF {
410     \int_compare:nNnTF \g__tbl_col_int > 0
411 }
```

(End of definition for \tbl_if_row_was_started:T and \tbl_if_row_was_started:TF.)

\tbl_gzero_row_count: This here is basically a temporary interface. What it will be in the end depends on what
\tbl_gincr_row_count: we decide concerning exposing row and column counters, if they stay internal we need
\tbl_gdecr_row_count: something like this here (perhaps using gincr etc, or perhaps some other names in the first place).

```
412 \cs_new_protected:Npn \tbl_gzero_row_count: {
413     \int_gzero:N \g__tbl_row_int
414 }
415 \cs_new_protected:Npn \tbl_gincr_row_count: {
416     \int_gincr:N \g__tbl_row_int
417 }
418 \cs_new_protected:Npn \tbl_gdecr_row_count: {
419     \int_gdecr:N \g__tbl_row_int
420 }
```

(End of definition for \tbl_gzero_row_count:, \tbl_gincr_row_count:, and \tbl_gdecr_row_count:.)

\tbl_inbetween_rows: Again name is not really brilliant so far.

```
421 \cs_new_protected:Npn \tbl_inbetween_rows: {
422     \int_gzero:N \g__tbl_col_int
423 }
```

(End of definition for \tbl_inbetween_rows:.)

\tbl_restore_outer_cell_data:

```
424 \cs_new_protected:Npn \tbl_restore_outer_cell_data: {
425     \int_gset:Nn \g__tbl_col_int { \l__tbl_saved_col_tl }
426     \int_gset:Nn \g__tbl_row_int { \l__tbl_saved_row_tl }
427     \tl_gset_eq:NN \g__tbl_span_tl \l__tbl_saved_span_tl
428     \tl_gset_eq:NN \g__tbl_table_cols_tl \l__tbl_saved_table_cols_tl
429     \UseTaggingSocket{tbl/restore/celldata}
430     \__tbl_trace:n { ==>~ restored~cell~data:~
431         \int_use:N \g__tbl_row_int,
432         \int_use:N \g__tbl_col_int,
433         \l__tbl_saved_span_tl \space
434         (
435         \int_compare:nNnTF \g__tbl_table_cols_tl = 0
436             { outer~ level }
437             { max:~ \g__tbl_table_cols_tl }
438         )
439     }
```

```

438         )
439     }
440 }

```

(End of definition for \tbl_restore_outer_cell_data:.)

`\tbl_update_multicolumn_cell_data:n` This macro updates `\g__tbl_col_int` and `\g__tbl_span_tl` inside a `\multicolumn` and possibly calls the tagging socket `tbl/row/begin`.

```

441 \cs_new_protected:Npn \tbl_update_multicolumn_cell_data:n #1 {

```

We execute socket for tagging only if this `\multicolumn` replaces the preamble of the first column. In that case we also have to set `\g__tbl_col_int` to 1 because this is no longer done in the preamble for the cell either.

```

442     \int_compare:nNnTF \g__tbl_col_int = 0
443     {
444         \UseTaggingSocket{tbl/row/begin}
445         \int_gset:Nn \g__tbl_col_int {1}
446     }

```

If we are in a later column we use `\g__tbl_span_tl` from the previous column to update.

```

447     {
448         \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
449     }

```

Then we set the span value so that it can be use in the next column.

```

450     \tl_gset:Nn \g__tbl_span_tl {#1}
451 }

```

(End of definition for \tbl_update_multicolumn_cell_data:n.)

`\tbl_crcrcr:n` This macro is used instead of the usual `\crcrcr` at the end of a table. It is deliberately defined without protection because it may get expanded by the scanning mechanism of low-level T_EX after a final `\cr` (aka `\`) in the table. In that case it shouldn't stop the expansion and the conditional inside will be false, thus it just vanishes without doing anything. If there are missing cells (in which case we also haven't see `\cr` yet) the macro `\tbl_count_missing_cells:n` is executed and then the row is finished with a final `\cr`.

```

452 \cs_new:Npn \tbl_crcrcr:n #1 {
453     \int_compare:nNnT \g__tbl_col_int > 0
454     {
455         \tbl_count_missing_cells:n {#1}
456     }

```

Even if we are at the start of a row we may have to do a `\cr`, so we do a `\crcrcr` always at the end.

```

457     \crcrcr
458 }

```

(End of definition for \tbl_crcrcr:n.)

```

459 \ExplSyntaxOff
460 <@@=>

```

This is needed for `longtable` because `\refstepcounter` is setting up a target when `hyperref` is loaded and we don't want that in `longtable`. Prevent `longtable` patching by `hyperref` until `hyperref` does so automatically:

```

461 \def\hyper@nopatch@longtable{}

```

Should there be a module?

```
462 <latexrelease>\NewModuleRelease{2024/06/01}{ltagging}  
463 <latexrelease>{Tagging support}  
  
464 <latexrelease>\IncludeInRelease{0000/00/00}{ltagging}%  
465 <latexrelease>{Undo tagging support}  
466 <latexrelease>  
467 <latexrelease>  
468 <latexrelease>  
469 <latexrelease>\EndModuleRelease  
470 </2ekernel | latexrelease>
```

File 56

lthyphen.dtx

This file contains the code for loading hyphenation patterns into L^AT_EX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your L^AT_EX system in respect of hyphenation patterns, write a file `hyphen.cfg`. If this file exists, it will be loaded instead of `hyphen.ltx`. See the comments below for additional information.

To produce the printed version of this file the following code is used. It can be extracted with the DOCSTRIP program, or one can run this file directly through L^AT_EX 2_ε.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \begin{document}
4 \DocInput{lthyphen.dtx}
5 \end{document}
6 </driver>
```

The default file `hyphen.ltx` loads hyphenation patterns for US english. If you want to load additional or other hyphenation patterns, you should create a file `hyphen.cfg`. This is best done by starting from `hyphen.ltx`.

For backward compatibility, the default file, `hyphen.ltx`, first tries to load the file `hyphen.tex`. If this file exists, an information message is issued and the appropriate defaults for T_EX's internal parameters are set: `\language` is initialized to 0, and `\lefthyphenmin` and `\righthyphenmin` to 2 and 3, respectively, to disallow x- or -xx breaks.

```
7 <*default>
8 \InputIfFileExists{hyphen.tex}%
9   {\message{Loading hyphenation patterns for US english.}}%
10   \language=0
11   \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniT_EX run is terminated by invoking `\@@end` (which is the L^AT_EX 2_ε name for T_EX's `\end` primitive).

```
12   {\errhelp{The configuration for hyphenation is incorrectly
13             installed.^^J%
14             If you don't understand this error message you need
15             to seek^^Jexpert advice.}%
16   \errmessage{OOPS! I can't find any hyphenation patterns for
17             US english.^^J \space Think of getting some or the
18             latex2e setup will never succeed}\@@end}
19 </default>
```

The following example describes the possible contents of a file `hyphen.cfg` that will load both US English and German hyphenation patterns, making the former the default. It sets `\language` to 0 for the US patterns and to 1 for the German patterns. Then `\language` is set to 0 to make this the default and the default values of `\lefthyphenmin` and `\righthyphenmin` are set.

```
language=0
input hyphen % (or \input ushyphen1 if the file has been renamed)
language=1
input ghyph31
```

```
language=0
lefthyphenmin=2
righthyphenmin=3
endinput
```

Another possibility is to use the package `babel`, by Johannes Braams. That package is distributed with a suitable `hyphen.cfg` file.

File 57

ltxfinal.dtx

1 Final settings

This section contains the final settings for L^AT_EX. It initializes some debugging and typesetting parameters, sets the default `\catcodes` and `uc/lc` codes, and inputs the hyphenation file.

1.1 Debugging

By default, L^AT_EX shows statistics:

```
1 <*2ekernel>
2 \tracingstats1
```

1.2 Typesetting parameters

```
\@lowpenalty These are penalties used internally.
\@medpenalty 3 \newcount\@lowpenalty
\@highpenalty 4 \newcount\@medpenalty
5 \newcount\@highpenalty
```

(End of definition for \@lowpenalty, \@medpenalty, and \@highpenalty.)

```
\newmarks Allocate extended marks types if etex is active. Placed here at the end of the format to
increase compatibility with count allocations in earlier releases.
```

```
6 </2ekernel>
7 <*2ekernel | latexrelease>
8 <latexrelease>\IncludeInRelease{2015/01/01}%
9 <latexrelease> \newmarks}{Extended Allocation}%
10 \ifx\marks\@undefined\else
11 \def\newmarks{%
12 \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\e@alloc@top}
13 \fi
14 </2ekernel | latexrelease>
15 <latexrelease>\EndIncludeInRelease
16 <latexrelease>\IncludeInRelease{0000/00/00}%
17 <latexrelease> \newmarks}{Extended Allocation}%
18 <latexrelease>\let\newmarks\@undefined
19 <latexrelease>\EndIncludeInRelease
20 <*2ekernel>
```

(End of definition for \newmarks.)

Allocate 3 mark classes to be used in `\markboth` and `\markright`. Should be done earlier but for that definition of `\newmarks` needs moving (which it should I guess).

```
21 </2ekernel>
22 <*2ekernel | latexrelease>
23 <latexrelease>\IncludeInRelease{2022/06/01}%
24 <latexrelease> {2e-left}{Delayed legacy marks}%
25 \NewMarkClass {2e-left}
```

```

26 \NewMarkClass {2e-right}
27 \NewMarkClass {2e-right-nonempty}

```

No rollback really, the marks will remain.

```

28 </2ekernel | latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <latexrelease>\IncludeInRelease{0000/00/00}%
31 <latexrelease> {2e-left}{Delayed legacy marks}%
32 <latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <*2ekernel>

```

`\newXeTeXintercharclass` Allocate `\XeTeXintercharclass` types if xetex is active. previously defined in `xetex.ini`.
`\xe@alloc@intercharclass`
`\e@alloc@intercharclass@top`

```

35 </2ekernel>
36 <*2ekernel | latexrelease>
37 <latexrelease>\IncludeInRelease{2015/01/01}%
38 <latexrelease> {\newXeTeXintercharclass}{Extended Allocation}%

```

Classes allocated 1 to 4094 (or 254 on older xetex) (In earlier XeLaTeX versions 1, 2 and 3 were pre-set for CJK).

```

39 \ifx\xeTeXcharclass\@undefined
40 \else
41 \ifdim\the\xeTeXversion\xeTeXrevision\p>0.99993\p@
42 \chardef\xe@alloc@intercharclass@top=4095
43 \else
44 \chardef\xe@alloc@intercharclass@top=255
45 \fi
46 \def\newXeTeXintercharclass{%
47 \e@alloc\xeTeXcharclass
48 \chardef\xe@alloc@intercharclass\m@ne\xe@alloc@intercharclass@top}
49 \fi
50 </2ekernel | latexrelease>
51 <latexrelease>\EndIncludeInRelease
52 <latexrelease>\IncludeInRelease{0000/00/00}%
53 <latexrelease> {\newXeTeXintercharclass}{Extended Allocation}%
54 <latexrelease> \ifx\xeTeXcharclass\@undefined
55 <latexrelease> \else
56 <latexrelease> \def\xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
57 <latexrelease> \xe@ch@ck#1#4#2%
58 <latexrelease> \allocationnumber#1%
59 <latexrelease> \global#3#5\allocationnumber
60 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}
61 <latexrelease> \def\xe@ch@ck#1#2#3{%
62 <latexrelease> \ifnum#1<#2\else
63 <latexrelease> \errmessage{No room for a new #3}%
64 <latexrelease> \fi}
65 <latexrelease> \def\newXeTeXintercharclass{%
66 <latexrelease> \xe@alloc@xe@alloc@intercharclass
67 <latexrelease> \XeTeXcharclass\chardef\@cclv}
68 <latexrelease> \fi
69 <latexrelease>\EndIncludeInRelease
70 <*2ekernel | latexrelease>
71 <latexrelease>\IncludeInRelease{2016/02/01}%

```



```

72 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
73 \ifx\XeTeXcharclass\@undefined
74 \else
75   \countdef\xe@alloc@intercharclass=257
76   \xe@alloc@intercharclass=\z@
77 \fi
78 </2ekernel | latexrelease>
79 <latexrelease>\EndIncludeInRelease
80 <latexrelease>\IncludeInRelease{2015/01/01}%
81 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
82 <latexrelease> \ifx\XeTeXcharclass\@undefined
83 <latexrelease> \else
84 <latexrelease>   \xe@alloc@intercharclass=\thr@@
85 <latexrelease> \fi
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
89 <latexrelease> \ifx\XeTeXcharclass\@undefined
90 <latexrelease> \else
91 <latexrelease>   \newcount\xe@alloc@intercharclass
92 <latexrelease>   \xe@alloc@intercharclass=\thr@@
93 <latexrelease> \fi
94 <latexrelease>\EndIncludeInRelease
95 <*2ekernel>

```

(End of definition for \newXeTeXintercharclass, \xe@alloc@intercharclass, and \e@alloc@intercharclass@top.)

trace_stack_levels Now define the Lua function to emulate \tracingstacklevels and install it in the input_level_string callback.

```

96 </2ekernel>
97 <*2ekernel | latexrelease>

```

In latexrelease mode we always remove the function from the callback, then add the correct version later.

```

98 <latexrelease>\ifx\directlua\@undefined
99 <latexrelease>\else
100 <latexrelease> \directlua{%
101 <latexrelease>   if luatexbase.callbacktypes['input_level_string'] and %
102 <latexrelease>     luatexbase.in_callback('input_level_string','tracingstacklevels') then
103 <latexrelease>     luatexbase.remove_from_callback('input_level_string','tracingstacklevels')
104 <latexrelease>   end}%
105 <latexrelease>\fi
106 <latexrelease>\IncludeInRelease{2021/06/01}{trace_stack_levels}%
107 <latexrelease>           {Lua trace_stack_levels function}%
108 \ifx\directlua\@undefined
109 \else
110 <*2ekernel>
111   \expanded{%
112     \everyjob{\the\everyjob
113       \noexpand%\directlua
114 </2ekernel>
115     \directlua{%
116       local function trace_stack_levels (input_ptr)
117         local tracingstacklevels = tex.count.tracingstacklevels

```

```

118     if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
119         if tracingstacklevels > 0 then
120             if input_ptr < tracingstacklevels then
121                 return "\string\n\string~" .. string.rep(".",input_ptr)
122             else
123                 return "\string~\string~"
124             end
125         else
126             return "\string\n"
127         end
128     else
129         return ""
130     end
131 end
132 <latexrelease>    if luatexbase.callbacktypes['input_level_string'] then
133     luatexbase.add_to_callback('input_level_string',
134         trace_stack_levels,'tracingstacklevels')
135 <latexrelease>    end
136 }%
137 <*2ekernel>
138 }}%
139 </2ekernel>
140 \fi
141 <latexrelease>\EndIncludeInRelease
142 <latexrelease>

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

143 <latexrelease>\IncludeInRelease{0000/00/00}{trace_stack_levels}%
144 <latexrelease>          {Lua trace_stack_levels function}%
145 <latexrelease>% Nothing here
146 <latexrelease>\EndIncludeInRelease
147 </2ekernel | latexrelease>
148 <*2ekernel>

```

(End of definition for trace_stack_levels.)

The default values of the picture and \fbox parameters:

```

149 \unitlength = 1pt
150 \fboxsep = 3pt
151 \fboxrule = .4pt

```

The saved value of T_EX's \maxdepth:

```

152 \@maxdepth      = \maxdepth

```

\vsize initialized because a \clearpage with \vsize < \topskip causes trouble. \@colroom and \@colht also initialized because \vsize may be set to them if a \clearpage is done before the \begin{document}

```

153 \vsize = 1000pt
154 \@colroom = \vsize
155 \@colht = \vsize

```

Initialize \textheight \textwidth and page style, to avoid internal errors if they are not set by the class.

```

156 \textheight=.5\maxdimen
157 \textwidth=\textheight
158 \ps@empty

```

1.3 Lccodes for hyphenation

For 7- and 8-bit engines the assumption of T1 encodings is the basis for the hyphenation patterns. That's not the case for the Unicode engines, where the assumption is engine-native working. The common loader system provides access to data from the Unicode Consortium covering not only `\lccode` but also other related data. The `\lccode` part of that at least needs to be loaded before hyphenation is tackled: XeTeX follows the standard TeX route of building patterns into the format. LuaTeX doesn't require this data be loaded *here* but it does need to be loaded somewhere. Rather than test for the Unicode engines by name, the approach here is to look for the extended math mode handling both provide: any other engine developed in this area will presumably also provide `\Umathcode`.

```
159 \ifnum 0%
160   \ifx\Umathcode\@undefined\else 1\fi
161   \ifx\XeTeXmathcode\@undefined\else 1\fi
162   >\z@
163   \message{ Unicode character data,}
164   \input{load-unicode-data}
165 \endkernel)
166 \latexrelease\IncludeInRelease{2016/02/01}%
167 \latexrelease {\XeTeXintercharclasses}{XeTeX character classes}%
168 \latexrelease \ifx\XeTeXinterchartoks\undefined
169 \latexrelease \else
170 \latexrelease \begingroup
171 \latexrelease \chardef\XeTeXcharclassID = 0 %
172 \latexrelease \chardef\XeTeXcharclassOP = 0 %
173 \latexrelease \chardef\XeTeXcharclassCL = 0 %
174 \latexrelease \chardef\XeTeXcharclassEX = 0 %
175 \latexrelease \chardef\XeTeXcharclassIS = 0 %
176 \latexrelease \chardef\XeTeXcharclassNS = 0 %
177 \latexrelease \chardef\XeTeXcharclassCM = 0 %
178 \latexrelease \input{load-unicode-xetex-classes}
179 \latexrelease \endgroup
180 \latexrelease \global\let\xtxHanGlue\undefined
181 \latexrelease \global\let\xtxHanSpace\undefined
182 \latexrelease \global\XeTeXinterchartoks 0 1 = {}
183 \latexrelease \global\XeTeXinterchartoks 0 2 = {}
184 \latexrelease \global\XeTeXinterchartoks 0 3 = {}
185 \latexrelease \global\XeTeXinterchartoks 1 0 = {}
186 \latexrelease \global\XeTeXinterchartoks 2 0 = {}
187 \latexrelease \global\XeTeXinterchartoks 3 0 = {}
188 \latexrelease \global\XeTeXinterchartoks 1 1 = {}
189 \latexrelease \global\XeTeXinterchartoks 1 2 = {}
190 \latexrelease \global\XeTeXinterchartoks 1 3 = {}
191 \latexrelease \global\XeTeXinterchartoks 2 1 = {}
192 \latexrelease \global\XeTeXinterchartoks 2 2 = {}
193 \latexrelease \global\XeTeXinterchartoks 2 3 = {}
194 \latexrelease \global\XeTeXinterchartoks 3 1 = {}
195 \latexrelease \global\XeTeXinterchartoks 3 2 = {}
196 \latexrelease \global\XeTeXinterchartoks 3 3 = {}
197 \latexrelease \fi
198 \latexrelease\EndIncludeInRelease
199 \latexrelease\IncludeInRelease{0000/00/00}%
```

```

200 <latexrelease> {\XeTeXintercharclasses}{XeTeX character classes}%
201 <latexrelease> \ifx\XeTeXinterchartoks\undefined
202 <latexrelease> \else
203 <latexrelease> \input{load-unicode-xetex-classes}
204 <latexrelease> \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
205 <latexrelease> \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
206 <latexrelease> \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
207 <latexrelease> \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
208 <latexrelease> \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
209 <latexrelease> \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
210 <latexrelease> \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
211 <latexrelease> \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
212 <latexrelease> \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
213 <latexrelease> \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
214 <latexrelease> \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
215 <latexrelease> \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}
216 <latexrelease> \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
217 <latexrelease> \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
218 <latexrelease> \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
219 <latexrelease> \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
220 <latexrelease> \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
221 <latexrelease> \fi
222 <latexrelease> \EndIncludeInRelease
223 <*2ekernel>

```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for - to itself.

```

224 \lccode'\- ='\- % default hyphen char

```

The alternative is that a “traditional” engine is in use.

```

225 \else

```

We set things up so that hyphenation files can assume that the default (T1) lccodes are in use (at present this also sets up the uccodes). We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```

226 \def\reserved@a#1#2{%
227   \@tempcnta#1\relax
228   \@tempcntb#2\relax
229   \reserved@b
230 }
231 \def\reserved@b{%
232   \ifnum\@tempcnta>\@tempcntb\else
233     \reserved@c\@tempcnta
234     \advance\@tempcnta\@ne
235     \expandafter\reserved@b
236   \fi
237 }

```

Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters.

```

238 \def\reserved@c#1{%
239   \count@=#1\advance\count@ by -"20
240   \uccode#1=\count@
241   \lccode#1=#1
242 }

```

```

243 \reserved@a{'\a}{'\z}
244 \reserved@a{"A0}{\BC}
245 \reserved@a{"E0}{\FF}

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```

246 \def\reserved@c#1{%
247   \count@=#1\advance\count@ by "20
248   \uccode#1=#1
249   \lccode#1=\count@
250   \sfcode#1=999
251 }
252 \reserved@a{'\A}{'\Z}
253 \reserved@a{"80}{\9C}
254 \reserved@a{"C0}{\DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose `uccode` or `lccode` isn't quite what you'd expect.

```

255 \uccode'\^^Y='I      % dotless i
256 \lccode'\^^Y='^^Y    % dotless i
257 \uccode'\^^Z='J      % dotless j, ae in OT1
258 \lccode'\^^Z='^^Z    % dotless j, ae in OT1
259 \lccode'\^^9d='i     % dotted I
260 \uccode'\^^9d='^^9d  % dotted I
261 \lccode'\^^9e='^^9e  % d-bar
262 \uccode'\^^9e='^^d0  % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

263 \lccode'\^^[='^^[    % oe in OT1

```

And we also set the `\lccode` of `\-` and `\textcompwordmark` so that they do not prevent hyphenation in the remainder of the word (as suggested by Lars Helström).

```

264 \lccode'\- ='\-      % default hyphen char
265 \lccode 127=127      % alternate hyphen char
266 \lccode 23 =23       % textcompwordmark in T1

```

End of the conditional to select either Unicode or T1 encoding defaults.

```

267 \fi

```

At this stage, we can install any last-minute `expl3` set-up.

```

268 \@expl@finalise@setup@@
269 \@expl@finalize@setup@@
270 \def\@expl@finalise@setup@@{}
271 \def\@expl@finalize@setup@@{}

```

This is as good a place as any to active a few XeTeX-specific settings

```

272 \ifx\XeTeXuseglyphmetrics\undefined
273 \else
274   \XeTeXuseglyphmetrics=1 %
275   \XeTeXdashbreakstate=1 %
276 \fi

```

1.4 Hyphenation

The following code will be compiled into the format file. It checks for the existence of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the `\catcodes` are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```
277 \InputIfFileExists{hyphen.cfg}
278     {\typeout{=====^~J%
279             Local configuration file hyphen.cfg used^^J%
280             =====}%
281     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
282     }
283     {\input{hyphen.ltx}}
284 \let\@addtofilelist\@gobble
```

`\l@nohyphenation`

```
285 \ifx\l@nohyphenation \@undefined
286     \newlanguage\l@nohyphenation
287 \fi
```

(End of definition for \l@nohyphenation.)

`\document@default@language` Default document language. -1 acts as language 0, but used as a flag in `\document` to see if it has been set in the preamble.

```
288 </2ekernel>
289 <*2ekernel | latexrelease>
290 <latexrelease>\IncludeInRelease{2017/04/15}%
291 <latexrelease>    {\document@default@language}{Save language for hyphenation}%
292 \let\document@default@language\m@ne
293 </2ekernel | latexrelease>
294 <latexrelease>\EndIncludeInRelease
295 <latexrelease>\IncludeInRelease{0000/00/00}%
296 <latexrelease>    {\document@default@language}{Save language for hyphenation}%
297 %
298 <latexrelease>\let\document@default@language\@undefined
299 <latexrelease>\EndIncludeInRelease
300 <*2ekernel>
```

(End of definition for \document@default@language.)

1.5 Font loading

Fonts loaded during the formatting process might already have changed the `\font@submax` from 0pt to something higher. If so, we put out a bold warning.

```
301 \ifdim \font@submax >\z@
302     \@font@warning{Size substitutions with differences\MessageBreak
303                 up to \font@submax\space have occurred.\MessageBreak
304                 \MessageBreak
305                 Please check the transcript file
306                 carefully\MessageBreak
307                 and redo the format generation if necessary!
308                 \@gobbletwo}%
309 \errhelp{Only stopped, to give you time to
310         read the above message.}
311 \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
312 \def\font@submax{0pt}
313 \fi
```

For pdfTeX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support. Since v1.40.22, this data is dumped into the format so is simply loaded directly. For older engines, the same method as LuaTeX is used (see below).

For luaTeX \pdfextension glyphtounicode would error in dvi-mode. As the outputmode can be changed in the preamble we have to test in the definition. LuaTeX does not dump this information into the format, so we need to load it in the \everyjob hook. We can't use \input there, as it would set the \jobname in the event of the run being started without loading a file. We also don't want to put all of the tokens from the files *directly* in the hook, so we save to a macro. We need to read with spaces “present” as this is part of the syntax of the primitive, so need \ExplSyntaxOff in the setup for \file_get:nnN.

For other engines we provide the two commands with noop definitions

```
314 \</2ekernel>
315 \<*2ekernel|latexrelease>
316 \<latexrelease>\IncludeInRelease{2026/06/01}%
317 \<latexrelease>{\pdfgentounicode}{Preload glyphtounicode}%
318 \ifdefined\directlua
319   \protected\def\pdfglyphtounicode{%
320     \ifnum\outputmode=1 %
321       \expandafter\@firstofone
322     \else
323       \expandafter\@gobblethree
324     \fi
325     {\pdfextension glyphtounicode}%
326   }
327   \protected\edef\pdfgentounicode{\pdfvariable gentounicode}
328 \fi
329 \ifdefined\pdfgentounicode
330 \<*2ekernel>
331   \ifnum 0=0%
332     \ifdefined\pdftexversion
333 % \pdftexversion<140 does not have \pdfgentounicode, so we only check higher values
334       \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
335     \else 1%
336     \fi
337     \relax
338 \</2ekernel>
339     \input glyphtounicode %
340     \input glyphtounicode-cmex %
341 \<*2ekernel>
342   \else
343     \ExplSyntaxOn
344       \file_get:nnN { glyphtounicode } { \ExplSyntaxOff }
345       \@@data@glyphtounicode
346       \file_get:nnN { glyphtounicode-cmex } { \ExplSyntaxOff }
347       \@@data@glyphtounicode@cmex
348     \ExplSyntaxOff
349     \everyjob\expandafter{%
```

```

350     \the\everyjob
351     \@@data@glyphtounicode
352     \@@data@glyphtounicode@cmex
353   }
354   \fi
355 </2ekernel>
356   \ifdefined\directlua
357     \pdfvariable gentounicode=1 %
358   \else
359     \pdfgentounicode=1 %
360   \fi
361 \fi

```

For other engines we provide the two commands with noop definitions

```

362 \ifx \pdfgentounicode \@undefined
363   \let\pdfglyphtounicode\@gobbletwo
364   \newcount\pdfgentounicode
365 \fi
366 </2ekernel | latexrelease>
367 <latexrelease>\EndIncludeInRelease
368 <latexrelease>\IncludeInRelease{2021/06/01}%
369 <latexrelease>          {\pdfgentounicode}{Preload glyphtounicode}%
370 <latexrelease>\ifx \directlua\undefined \else
371 <latexrelease>  \pdfvariable gentounicode=0
372 <latexrelease>\fi
373 <latexrelease>\ifdefined\pdftexversion
374 <latexrelease> \ifx \pdfgentounicode \@undefined \else
375 <latexrelease>  \ifnum 0=0%
376 <latexrelease>    \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
377 <latexrelease>    \relax
378 <latexrelease>    \input glyphtounicode
379 <latexrelease>  \else
380 <latexrelease>    \begingroup
381 <latexrelease>      \everyeof{\noexpand}\endlinechar-1
382 <latexrelease>      \edef\x{\endgroup
383 <latexrelease>        \everyjob{\the\everyjob\@@input glyphtounicode }%
384 <latexrelease>      }\x
385 <latexrelease>  \fi
386 <latexrelease>  \pdfgentounicode=1
387 <latexrelease> \fi
388 <latexrelease>\fi
389 <latexrelease>\EndIncludeInRelease

```

When rolling back we can't unload the glyphtounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used and then undefine the commands for the engines that do not natively support it.

```

390 <latexrelease>\IncludeInRelease{0000/00/00}%
391 <latexrelease>          {\pdfgentounicode}{Preload glyphtounicode}%
392 <latexrelease>  \pdfgentounicode=0
393 <latexrelease>\EndIncludeInRelease
394 <*2ekernel>

```


1.6 Input encoding

Starting with the 2018 L^AT_EX release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, encTeX or mltex.

This is done in a way largely compatible with older releases: `utf8.def` is input just as if

```
\usepackage[utf8]{inputenc}
```

had been used, however rather than input the whole package a minimal core part just enough to support loading the UTF-8 encoding files is defined here.

If a document re-specifies UTF-8 this is silently ignored.

```
395 </2ekernel>
396 <*2ekernel | latexrelease>

    Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

397 <latexrelease>\IncludeInRelease{2018/04/01}%
398 <latexrelease>                {\UTFviii@invalid}{UTF-8 default}%
```

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

```
399 \ifnum0%
400   \ifx\Umathcode\@undefined\else 1\fi
401   \ifx\mubyte\@undefined\else 1\fi
402   \ifx\charsubdef\@undefined\else 1\fi
403   =\z@

404 \def\saved@space@catcode{10}
405 \let\@inpenc@test\relax
406 \def\IeC{%
407   \ifx\protect\@typeset@protect
408     \expandafter\@firstofone
409   \else
410     \noexpand\IeC
411   \fi
412 }
```

Make characters active for UTF-8 input formats

```
413 \@tempcnta=1
414 \loop
415   \catcode\@tempcnta=13 %
416   \advance\@tempcnta\@ne %
417   \ifnum\@tempcnta<32 %
418     \repeat %
419   \catcode0=15 % null
420   \catcode9=10 % tab
421   \catcode10=12 % ctrl J
422   \catcode12=13 % ctrl L
423   \catcode13=5 % newline
424   \@tempcnta=128
425   \loop
426     \catcode\@tempcnta=13
427     \advance\@tempcnta\@ne
428     \ifnum\@tempcnta<256
429     \repeat
```

`\UseRawInputEncoding` Reset 8 bit characters to catcode 12 so the input encoding matches the “Raw” font encoding. Useful for special behaviours, or for compatibility with older L^AT_EX formats.

```

430 \def\UseRawInputEncoding{%
431 \let\inputencodingname\@undefined % revert
432 \let\DeclareFontEncoding\DeclareFontEncoding@saved % revert
433 \let\DeclareUnicodeCharacter\@undefined % revert
434 \@tempcnta=1
435 \loop
436 \catcode\@tempcnta=15 %
437 \advance\@tempcnta\@ne %
438 \ifnum\@tempcnta<32 %
439 \repeat %
440 \catcode0=15 % null
441 \catcode9=10 % tab
442 \catcode10=12 % ctrl J
443 \catcode12=13 % ctrl L
444 \catcode13=5 % newline
445 \@tempcnta=128
446 \loop
447 \catcode\@tempcnta=12
448 \advance\@tempcnta\@ne
449 \ifnum\@tempcnta<256
450 \repeat
451 }
```

(End of definition for \UseRawInputEncoding.)

`\DeclareFontEncoding@saved` Saved version of `\DeclareFontEncoding@` before `utf8.def` modifies it for use in `\UseRawInputEncoding` above.

```

452 \let\DeclareFontEncoding@saved\DeclareFontEncoding@
```

(End of definition for \DeclareFontEncoding@saved.)

```

453 \edef\inputencodingname{utf8}%
454 \input{utf8.def}
455 \let\UTFviii@undefined@err@@\UTFviii@undefined@err
456 \let\UTFviii@invalid@err@@\UTFviii@invalid@err
457 \let\UTFviii@two@octets@@\UTFviii@two@octets
458 \let\UTFviii@three@octets@@\UTFviii@three@octets
459 \let\UTFviii@four@octets@@\UTFviii@four@octets
460 <2ekernel>\def\UTFviii@undefined@err#1{\@gobble#1}%
461 <2ekernel>\let\UTFviii@invalid@err\string
462 <2ekernel>\let\UTFviii@two@octets\string
463 <2ekernel>\let\UTFviii@three@octets\string
464 <2ekernel>\let\UTFviii@four@octets\string
465 <2ekernel>\everyjob\expandafter{\the\everyjob
466 <2ekernel>\let\UTFviii@undefined@err\UTFviii@undefined@err@@
467 <2ekernel>\let\UTFviii@invalid@err\UTFviii@invalid@err@@
468 <2ekernel>\let\UTFviii@two@octets\UTFviii@two@octets@@
469 <2ekernel>\let\UTFviii@three@octets\UTFviii@three@octets@@
470 <2ekernel>\let\UTFviii@four@octets\UTFviii@four@octets@@
471 <2ekernel>}}
472 \let\@inpenc@test\@undefined
473 \let\saved@space@catcode\@undefined
```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```

474 \else
475 \@tempcnta=0
476 \loop
477   \catcode\@tempcnta=15 %
478   \advance\@tempcnta\@ne %
479   \ifnum\@tempcnta<32 %
480   \repeat %
481 \catcode0=15 % null
482 \catcode9=10 % tab
483 \catcode10=12 % ctrl J
484 \catcode12=13 % ctrl L
485 \catcode13=5 % newline
486 \let\UseRawInputEncoding\relax

```

This ends the skipped code in Unicode engines:

```

487 \fi
488 </2ekernel | latexrelease>
489 <latexrelease>\EndIncludeInRelease
490 <latexrelease>\IncludeInRelease{0000/00/00}%
491 <latexrelease>          {\UTFviii@invalid}{UTF-8 default}%

```

The first block of commands got only introduced in 2019 but we revert all of Unicode support in one go not jump to the intermediate version.

```

492 <latexrelease> \let\UTFviii@two@octets@combine\@undefined
493 <latexrelease> \let\UTFviii@three@octets@combine\@undefined
494 <latexrelease> \let\UTFviii@four@octets@combine\@undefined
495 <latexrelease> \let\UTFviii@two@octets@string\@undefined
496 <latexrelease> \let\UTFviii@three@octets@string\@undefined
497 <latexrelease> \let\UTFviii@four@octets@string\@undefined
498 <latexrelease> \let\UTFviii@two@octets@noexpand\@undefined
499 <latexrelease> \let\UTFviii@three@octets@noexpand\@undefined
500 <latexrelease> \let\UTFviii@four@octets@noexpand\@undefined

501 <latexrelease>\@tempcnta=0
502 <latexrelease>\loop
503 <latexrelease> \catcode\@tempcnta=15
504 <latexrelease> \advance\@tempcnta\@ne
505 <latexrelease>\ifnum\@tempcnta<32
506 <latexrelease>\repeat %
507 <latexrelease>\catcode9=10 % tab
508 <latexrelease>\catcode10=12 % ctrl J
509 <latexrelease>\catcode12=13 % ctrl L
510 <latexrelease>\catcode13=5 % newline
511 <latexrelease>\@tempcnta=128
512 <latexrelease>\loop
513 <latexrelease>\catcode\@tempcnta=12
514 <latexrelease>\advance\@tempcnta\@ne
515 <latexrelease>\ifnum\@tempcnta<256
516 <latexrelease>\repeat
517 <latexrelease>\let\IeC\@undefined
518 <latexrelease>\def\DeclareFontEncoding@#1#2#3{%
519 <latexrelease> \expandafter
520 <latexrelease> \ifx\csname T@#1\endcsname\relax

```

```

521 <latexrelease> \def\cdp@elt{\noexpand\cdp@elt}%
522 <latexrelease> \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
523 <latexrelease> {\default@family}{\default@series}%
524 <latexrelease> {\default@shape}}%
525 <latexrelease> \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
526 <latexrelease> \else
527 <latexrelease> \font@info{Redeclaring font encoding #1}%
528 <latexrelease> \fi
529 <latexrelease> \global\@namedef{T@#1}{#2}%
530 <latexrelease> \global\@namedef{M@#1}{\default@M#3}%
531 <latexrelease> \xdef\LastDeclaredEncoding{#1}%
532 <latexrelease> }
533 <latexrelease> \let\UseRawInputEncoding\@undefined
534 <latexrelease> \let\DeclareFontEncoding@saved\@undefined
535 <latexrelease> \let\inputencodingname\@undefined
536 <latexrelease> \EndIncludeInRelease
537 <*2ekernel>

```

We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```

538 \def\reserved@a#1#2{%
539 \tempcnta#1\relax
540 \tempcntb#2\relax
541 \reserved@b
542 }
543 \def\reserved@b{%
544 \ifnum\tempcnta>\tempcntb\else
545 \reserved@c\tempcnta
546 \advance\tempcnta\@ne
547 \expandafter\reserved@b
548 \fi
549 }

```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that `^^J` has catcode ‘other’ for use in warning messages.

```

550 \catcode'\ =10
551 \catcode'\#=6
552 \catcode'\$=3
553 \catcode'\%=14
554 \catcode'\&=4
555 \catcode'\\=0
556 \catcode'\^=7
557 \catcode'\_ =8
558 \catcode'\{=1
559 \catcode'\}=2
560 \catcode'\-=13
561 \catcode'\@=11
562 \catcode'\^^I=10
563 \catcode'\^^J=12
564 \catcode'\^^L=13
565 \catcode'\^^M=5

```

Set the ‘other’ catcodes.

```

566 \def\reserved@c#1{\catcode#1=12\relax}
567 \reserved@c{'\!}
568 \reserved@c{'\"}
569 \reserved@a{'\'}{'\?}
570 \reserved@c{'\[]
571 \reserved@c{'\]}
572 \reserved@c{'\'
573 \reserved@c{'\|}

```

Set the ‘letter’ catcodes.

```

574 \def\reserved@c#1{\catcode#1=11\relax}
575 \reserved@a{'\A}{'\Z}
576 \reserved@a{'\a}{'\z}

```

All the characters in the range 0–31 and 127–255 are illegal, *except* tab (`^I`), nl (`^J`), ff (`^L`) and cr (`^M`).

1.7 Lccodes and uccodes

We now again set up the default (T1) uc/lccodes. The lower case characters need their `\uccode` and `\lccode` values set. Some of this is a repeat of the set-up before loading hyphenation files. Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (XeT_EX and LuaT_EX) there is no need to do any of this: they use hyphenation data which does not alter any of the set up and so this entire block is skipped.

```

577 \ifnum 0%
578   \ifx\Umathcode\@undefined\else 1\fi
579   \ifx\XeTeXmathcode\@undefined\else 1\fi
580   >\z@
581 \else
582 \def\reserved@c#1{%
583   \count@=#1\advance\count@ by -"20
584   \uccode#1=\count@
585   \lccode#1=#1
586 }
587 \reserved@a{'\a}{'\z}
588 \reserved@a{"A0}{"BC}
589 \reserved@a{"E0}{"FF}

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```

590 \def\reserved@c#1{%
591   \count@=#1\advance\count@ by "20
592   \uccode#1=#1
593   \lccode#1=\count@
594   \sfcode#1=999
595 }
596 \reserved@a{'\A}{'\Z}
597 \reserved@a{"80}{"9C}
598 \reserved@a{"C0}{"DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn’t quite what you’d expect.

```

599 \uccode{'\^Y='^I      % dotless i
600 \lccode{'\^Y='^Y      % dotless i

```

```

601 \uccode'\^^Z='^J      % dotless j, ae in OT1
602 \lccode'\^^Z='^Z      % dotless j, ae in OT1
603 \lccode'\^^9d='^i     % dotted I
604 \uccode'\^^9d='^9d    % dotted I
605 \lccode'\^^9e='^9e    % d-bar
606 \uccode'\^^9e='^d0    % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

607 \lccode'\^^[='^^[     % oe in OT1
608 \fi % End of reset block for 8-bit engines

```

`\BCPdata` A stub for use by babel, polyglossia, etc.

```

609 \ExplSyntaxOn
610 \newcommand*\BCPdata[1]{
611   \str_case:nn {#1}
612   {
613     { language } { en }
614     { region }   { US }
615     { script }   { Latn }
616     { tag }      { en-US }
617   }
618 }
619 \ExplSyntaxOff

```

(End of definition for \BCPdata.)

1.8 Case changing

`\MakeUppercase` And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

`\MakeLowercase` Wrappers around the L3 case changing functions. `\protected` to make them mostly safe as replacements for `\uppercase` and `\lowercase`.

`\MakeTitlecase`

`\NoCaseChange`

`\AddToNoCaseChangeList`

`\CaseSwitch`

`\DeclareCaseChangeEquivalent`

`\DeclareLowercaseMapping`

`\DeclareTitlecaseMapping`

`\DeclareUppercaseMapping`

`\DeclareLowercaseExclusions`

`\DeclareTitlecaseExclusions`

`\DeclareUppercaseExclusions`

`\@uclclist`

In

```

\markboth{\MakeUppercase\contentsname}
{\MakeUppercase\contentsname}

```

then the uppercasing is only done to the first letter of the contents name, since the mark expands out to:

```

\mark{\MakeUppercase Table of Contents}
{\MakeUppercase Table of Contents}

```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it.

Earlier versions needed to process `\@uclclist` in an `\edef` to handle legacy input encodings, but recent (2022) expl3 versions handle non-UTF8 text natively so we simply call the `\text_...case:n` functions.

```

620 \ExplSyntaxOn
621 \keys_define:nn { __kernel }
622 {
623   lang .str_set:N = \reserved@a ,
624   locale .str_set:N = \reserved@a ,
625   words .choices:nn =

```

```

626     { all , first }
627     { \str_set:Nn \reserved@b {#1} }
628   }
629 \cs_new_protected:Npn \@@text@case@aux #1#2#3
630 {
631   \cs_set_nopar:Npn \reserved@a { }
632   \cs_set_nopar:Npn \reserved@b { first }
633   \tl_if_blank:nTF {#2}
634   {
635     \str_set:Ne \reserved@a
636     { \BCPdata { casing } }
637     \str_if_empty:NT \reserved@a
638     {
639       \str_set:Ne \reserved@a
640       { \BCPdata { language } }
641     }
642   }
643   { \keys_set:nn { __kernel } {#2} }
644   \use:c { text_ #1 :Vn } \reserved@a {#3}
645 }

```

The odd use of *three* spaces here is needed as `ltxcmd` uses the name with one and two spaces to give a ‘friendly’ error message for a runaway argument: that means we can’t use it here.

```

646 \exp_args_generate:n { cnx }
647 \cs_set_protected:Npn \reserved@a #1
648 {
649   \cs_generate_variant:cn { text_ \str_lowercase:n {#1} case:nn } { V }
650   \ExpandArgs { cnx } \NewExpandableDocumentCommand
651     { Make#1case }
652     { O{ } +m }
653     { \exp_not:c { Make#1case \c_space_tl \c_space_tl \c_space_tl } [###1] {####2} }
654 }
655 \reserved@a { Upper }
656 \reserved@a { Lower }
657 \reserved@a { Title }

```

These don’t get covered above so we do them manually.

```

658 \cs_generate_variant:Nn \text_titlecase_all:nn { V }
659 \cs_generate_variant:Nn \text_titlecase_first:nn { V }

```

Currently, `babel` uses the equivalence of `\oe` and `\OE` to force casing of some material, most notably in `\today`. To enable that to work, we have to set those commands equal even though the current case changing code does not work using this approach.

```

660 \cs_new_protected:cpn { MakeLowercase \c_space_tl \c_space_tl \c_space_tl } [ #1 ] #2
661 {{
662   \let \OE \oe
663   \@@text@case@aux { lowercase } {#1} {#2}
664 }}
665 \cs_new_protected:cpn { MakeUppercase \c_space_tl \c_space_tl \c_space_tl } [ #1 ] #2
666 {{
667   \let \oe \OE
668   \@@text@case@aux { uppercase } {#1} {#2}
669 }}
670 \cs_new_protected:cpn { MakeTitlecase \c_space_tl \c_space_tl \c_space_tl } [ #1 ] #2

```

```

671 {{
672   \let \oe \OE
673   \@@text@case@aux { titlecase_ \reserved@b } {#1} {#2}
674 }}

```

\NoCaseChange protects its argument from the case change functions.

\AddToNoCaseChangeList Allows new commands to protect their arguments, eg \AddToNoCaseChangeList{\eqref} would protect the argument of \eqref in the same way as the argument of \ref.

```

675 \cs_new_protected_nopar:Npn\AddToNoCaseChangeList
676   {\tl_put_right:Nn \l_text_case_exclude_arg_tl}
677 \AddToNoCaseChangeList{ \NoCaseChange }
678 \cs_new_protected:Npn \NoCaseChange #1 {#1}
679 \cs_new_eq:NN \CaseSwitch \text_case_switch:nnnn
680 \cs_new_eq:NN \DeclareCaseChangeEquivalent
681   \text_declare_case_equivalent:Nn
682 \NewDocumentCommand \DeclareLowercaseMapping { o m m }
683 {
684   \IfNoValueTF {#1}
685     { \text_declare_lowercase_mapping:nn }
686     { \text_declare_lowercase_mapping:nnn {#1} }
687   {#2} {#3}
688 }
689 \NewDocumentCommand \DeclareTitlecaseMapping { o m m }
690 {
691   \IfNoValueTF {#1}
692     { \text_declare_titlecase_mapping:nn }
693     { \text_declare_titlecase_mapping:nnn {#1} }
694   {#2} {#3}
695 }
696 \NewDocumentCommand \DeclareUppercaseMapping { o m m }
697 {
698   \IfNoValueTF {#1}
699     { \text_declare_uppercase_mapping:nn }
700     { \text_declare_uppercase_mapping:nnn {#1} }
701   {#2} {#3}
702 }
703 \cs_new_protected:Npn \DeclareLowercaseExclusions #1
704 {
705   \clist_map_inline:nn {#1}
706     { \text_declare_lowercase_exclusion:n {##1} }
707 }
708 \cs_new_protected:Npn \DeclareTitlecaseExclusions #1
709 {
710   \clist_map_inline:nn {#1}
711     { \text_declare_titlecase_exclusion:n {##1} }
712 }
713 \cs_new_protected:Npn \DeclareUppercaseExclusions #1
714 {
715   \clist_map_inline:nn {#1}
716     { \text_declare_uppercase_exclusion:n {##1} }
717 }
718 \ExplSyntaxOff

```



```

719 \def\@uclclist{\oe\OE\o\O\ae\AE
720 \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\ij\IJ\th\TH}

```

1.9 Automatic insertion of \par tokens

Setting `\partokencontext` to 2 ensures a literal `\par` is used in all contexts, setting to 0 reverts to the classic `TEX` behavior.

1.10 Applying Patch files

Patch file code removed.

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

1.12 Initialize file list

Initialize `\strongemph`. This needs delaying as some of the necessary macros are defined late. By default we alternate between bold and bold italic.

```
786 \DeclareRobustCommand\strongemreset{\bfseries\upshape\ulcshape}  
787 \DeclareStrongEmphSequence{\bfseries,\itshape}
```

1.13 Initialize file list

`\@providesfile` Initialize for use in the document. During initex a modified version has been used which leaves debugging information for `latexbug.tex`.

```
788 \def\@providesfile#1[#2]{%  
789   \wlog{File: #1 #2}%  
790   \expandafter\xdef\csname ver@#1\endcsname{#2}%  
791   \endgroup}
```

(End of definition for \@providesfile.)

`\@filelist` Reset `\@filelist` so files input while making the format are not listed. The list built up so far may take up a lot of memory and so it is moved to `\reserved@a` where it will be overwritten as soon as almost any $\text{\LaTeX}$ command is issued in a class file. However the `latexbug.tex` program will be able to access this information and insert it into a bug report.

```
\@addtofilelist  
\@addtofilelist@aux  
792 \let\@filelist\@gobble  
793 \def\@addtofilelist#1{\xdef\@filelist  
794   {\@filelist,\expanded{\noexpand\@addtofilelist@aux#1},\@nil{#1}}}  
795 \def\@addtofilelist@aux#1,#2\@nil#3{%  
796   \if\relax\detokenize\expandafter{\@gobble#2?}\relax  
797     #1%  
798   \else  
799     {#3}%  
800   \fi  
801 }
```

(End of definition for \@filelist, \@addtofilelist, and \@addtofilelist@aux.)

1.14 Preparation for supporting PDF in backends

At the current point in time, basic support for PDF in backends is not part of $\text{\LaTeX}$ core; it is provided by external packages. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document.

In that code there is a command `\IfPDFManagementActiveTF` which can be used by packages in order to execute different code depending on the whether this basic backend support is loaded.

To make this also work properly when this external package is not loaded at all, we here add this command already in the kernel (with a trivial definition); thus any package can query this loading state in all circumstances. Once this basic PDF backend support gets moved to the kernel, this definition will vanish again from here or, rather, it will be replaced by a real test.

`\IfPDFManagementActiveTF` So long as the code for the basic backend support for PDF is not loaded, the test that is implicit here will always return the false branch. Once this code is loaded, this definition will get replaced by a real test (as it is then possible that the management code is either activated or not activated).

```
802 \let \IfPDFManagementActiveTF \@secondoftwo
```

The T/F variants are defined in a way that they continue to work when the main definition changes.

```
803 \long\def\IfPDFManagementActiveT#1{\IfPDFManagementActiveTF{#1}{}}
```

```
804 \def\IfPDFManagementActiveF{\IfPDFManagementActiveTF{}}
```

(End of definition for \IfPDFManagementActiveTF.)

1.15 Do some temporary work for pre-release

This is a good place to load code that hasn't yet been integrated into the other files ...

1.16 Some last minute initializations ...

Load the first aid set of definitions for external packages that await updates.

```
805 \@input{latex2e-first-aid-for-external-files.ltx}
```

1.17 Dumping the format

Finally we make @ into a letter, ensure the format will be in the 'normal' error mode, and dump everything into the format file.

```
806 \makeatother
```

```
807 \errorstopmode
```

```
808 \dump
```

```
809 \</2ekernel>
```

```
810 <*glyptounicode>
```

```
811 %% A subset of glyhtounicode-cmr.tex
```

```
812 %
```

```
813 %% Copyright (c) 2008, Han The Thanh <thanh@river-valley.org>
```

```
814 %% Copyright (c) 2014, Peter Selinger <selinger@mathstat.dal.ca>
```

```
815 %% Copyright (c) 2018, Ross Moore <ross.moore@mq.edu.au>
```

```
816 %%
```

```
817
```

```
818 %% Glyphs from the cmex fonts:
```

```
819
```

```
820 \pdfglyptounicode{angbracketleftBig}{27E8 FE02}
```

```
821 \pdfglyptounicode{angbracketleftBigg}{27E8 FE04}
```

```
822 \pdfglyptounicode{angbracketleftbig}{27E8 FE01}
```

```
823 \pdfglyptounicode{angbracketleftbigg}{27E8 FE03}
```

```
824 \pdfglyptounicode{angbracketrightBig}{27E9 FE02}
```

```
825 \pdfglyptounicode{angbracketrightBigg}{27E9 FE04}
```

```
826 \pdfglyptounicode{angbracketrightbig}{27E9 FE01}
```

```
827 \pdfglyptounicode{angbracketrightbigg}{27E9 FE03}
```

```
828 \pdfglyptounicode{arrowbt}{2193}
```

```
829 \pdfglyptounicode{arrowdblbt}{21D3}
```

```
830 \pdfglyptounicode{arrowdbltp}{21D1}
```

```
831 \pdfglyptounicode{arrowhookleft}{21AA}
```

832 \pdfglyphtounicode{arrowhookright}{21A9}
 833 \pdfglyphtounicode{arrowtp}{2191}
 834 \pdfglyphtounicode{arrowvertex}{23D0}
 835 \pdfglyphtounicode{arrowvertexdbl}{20E6}% was {ED12}% PUA
 836 \pdfglyphtounicode{backslashBig}{005C FE02}
 837 \pdfglyphtounicode{backslashBigg}{005C FE04}
 838 \pdfglyphtounicode{backslashbig}{005C FE01}
 839 \pdfglyphtounicode{backslashbigg}{005C FE03}
 840 \pdfglyphtounicode{braceex}{23AA}
 841 \pdfglyphtounicode{bracehtipdownleft}{23DF}% was {ED17}% PUA
 842 \pdfglyphtounicode{bracehtipdownright}{23DF}% was {ED18}% PUA
 843 \pdfglyphtounicode{bracehtipupleft}{23DE}% was {ED19}% PUA
 844 \pdfglyphtounicode{bracehtipupright}{23DE}% was {ED1A}% PUA
 845 \pdfglyphtounicode{braceleftBig}{007B FE02}
 846 \pdfglyphtounicode{braceleftBigg}{007B FE04}
 847 \pdfglyphtounicode{braceleftbig}{007B FE01}
 848 \pdfglyphtounicode{braceleftbigg}{007B FE03}
 849 \pdfglyphtounicode{braceleftbt}{23A9}
 850 \pdfglyphtounicode{braceleftmid}{23A8}
 851 \pdfglyphtounicode{bracelefttp}{23A7}
 852 \pdfglyphtounicode{bracerightBig}{007D FE02}
 853 \pdfglyphtounicode{bracerightBigg}{007D FE04}
 854 \pdfglyphtounicode{bracerightbig}{007D FE01}
 855 \pdfglyphtounicode{bracerightbigg}{007D FE03}
 856 \pdfglyphtounicode{bracerightbt}{23AD}
 857 \pdfglyphtounicode{bracerightmid}{23AC}
 858 \pdfglyphtounicode{bracerighttp}{23AB}
 859 \pdfglyphtounicode{bracketleftBig}{005B FE02}
 860 \pdfglyphtounicode{bracketleftBigg}{005B FE04}
 861 \pdfglyphtounicode{bracketleftbig}{005B FE01}
 862 \pdfglyphtounicode{bracketleftbigg}{005B FE03}
 863 \pdfglyphtounicode{bracketleftbt}{23A3}
 864 \pdfglyphtounicode{bracketleftex}{23A2}
 865 \pdfglyphtounicode{bracketlefttp}{23A1}
 866 \pdfglyphtounicode{bracketrightBig}{005D FE02}
 867 \pdfglyphtounicode{bracketrightBigg}{005D FE04}
 868 \pdfglyphtounicode{bracketrightbig}{005D FE01}
 869 \pdfglyphtounicode{bracketrightbigg}{005D FE03}
 870 \pdfglyphtounicode{bracketrightbt}{23A6}
 871 \pdfglyphtounicode{bracketrightex}{23A5}
 872 \pdfglyphtounicode{bracketrighttp}{23A4}
 873 \pdfglyphtounicode{ceilingleftBig}{2308 FE02}
 874 \pdfglyphtounicode{ceilingleftBigg}{2308 FE04}
 875 \pdfglyphtounicode{ceilingleftbig}{2308 FE01}
 876 \pdfglyphtounicode{ceilingleftbigg}{2308 FE03}
 877 \pdfglyphtounicode{ceilingrightBig}{2309 FE02}
 878 \pdfglyphtounicode{ceilingrightBigg}{2309 FE04}
 879 \pdfglyphtounicode{ceilingrightbig}{2309 FE01}
 880 \pdfglyphtounicode{ceilingrightbigg}{2309 FE03}
 881 \pdfglyphtounicode{circledotdisplay}{2A00 FE02}
 882 \pdfglyphtounicode{circledottext}{2A00 FE01}
 883 \pdfglyphtounicode{circlemultiplydisplay}{2A02 FE02}
 884 \pdfglyphtounicode{circlemultiplytext}{2A02 FE01}
 885 \pdfglyphtounicode{circleplusdisplay}{2A01 FE02}

886 \pdfglyphtounicode{circleplustext}{2A01 FE01}
 887 \pdfglyphtounicode{contintegraldisplay}{222E FE02}
 888 \pdfglyphtounicode{contintegraltext}{222E FE01}
 889 \pdfglyphtounicode{coproductdisplay}{2210 FE02}
 890 \pdfglyphtounicode{coproducttext}{2210 FE01}
 891 \pdfglyphtounicode{floorleftBig}{230A FE02}
 892 \pdfglyphtounicode{floorleftBigg}{230A FE04}
 893 \pdfglyphtounicode{floorleftbig}{230A FE01}
 894 \pdfglyphtounicode{floorleftbigg}{230A FE03}
 895 \pdfglyphtounicode{floorrightBig}{230B FE02}
 896 \pdfglyphtounicode{floorrightBigg}{230B FE04}
 897 \pdfglyphtounicode{floorrightbig}{230B FE01}
 898 \pdfglyphtounicode{floorrightbigg}{230B FE03}
 899 \pdfglyphtounicode{hatwide}{02C6 FE01}
 900 \pdfglyphtounicode{hatwider}{02C6 FE02}
 901 \pdfglyphtounicode{hatwidest}{02C6 FE03}
 902 \pdfglyphtounicode{integraldisplay}{222B FE02}
 903 \pdfglyphtounicode{integraltext}{222B FE01}
 904 \pdfglyphtounicode{intersectiondisplay}{22C2 FE02}
 905 \pdfglyphtounicode{intersectiontext}{22C2 FE01}
 906 \pdfglyphtounicode{logicalanddisplay}{22C0 FE02}
 907 \pdfglyphtounicode{logicalandtext}{22C0 FE01}
 908 \pdfglyphtounicode{logicalordisplay}{22C1 FE02}
 909 \pdfglyphtounicode{logicalortext}{22C1 FE01}
 910 \pdfglyphtounicode{mapsto}{21A6}
 911 \pdfglyphtounicode{parenleftBig}{0028 FE02}
 912 \pdfglyphtounicode{parenleftBigg}{0028 FE04}
 913 \pdfglyphtounicode{parenleftbig}{0028 FE01}
 914 \pdfglyphtounicode{parenleftbigg}{0028 FE03}
 915 \pdfglyphtounicode{parenleftbt}{239D}
 916 \pdfglyphtounicode{parenleftex}{239C}
 917 \pdfglyphtounicode{parenlefttp}{239B}
 918 \pdfglyphtounicode{parenrightBig}{0029 FE02}
 919 \pdfglyphtounicode{parenrightBigg}{0029 FE04}
 920 \pdfglyphtounicode{parenrightbig}{0029 FE01}
 921 \pdfglyphtounicode{parenrightbigg}{0029 FE03}
 922 \pdfglyphtounicode{parenrightbt}{23A0}
 923 \pdfglyphtounicode{parenrightex}{239F}
 924 \pdfglyphtounicode{parenrighttp}{239E}
 925 \pdfglyphtounicode{productdisplay}{220F FE02}
 926 \pdfglyphtounicode{producttext}{220F FE01}
 927 \pdfglyphtounicode{radicalBig}{221A FE02}
 928 \pdfglyphtounicode{radicalBigg}{221A FE04}
 929 \pdfglyphtounicode{radicalbig}{221A FE01}
 930 \pdfglyphtounicode{radicalbigg}{221A FE03}
 931 \pdfglyphtounicode{radicalbt}{23B7}% was {221A}
 932 \pdfglyphtounicode{radicaltp}{231C}% was {ED6A}% PUA
 933 \pdfglyphtounicode{radicalvertex}{20D3}% was {ED6B}% PUA
 934 \pdfglyphtounicode{slashBig}{002F FE02}
 935 \pdfglyphtounicode{slashBigg}{002F FE04}
 936 \pdfglyphtounicode{slashbig}{002F FE01}
 937 \pdfglyphtounicode{slashbigg}{002F FE03}
 938 \pdfglyphtounicode{summationdisplay}{2211 FE02}
 939 \pdfglyphtounicode{summationtext}{2211 FE01}

```

940 \pdfglyphtounicode{tie}{2040}
941 \pdfglyphtounicode{tildewide}{02DC FE01}
942 \pdfglyphtounicode{tildewider}{02DC FE02}
943 \pdfglyphtounicode{tildewidest}{02DC FE03}
944 \pdfglyphtounicode{uniondisplay}{22C3 FE02}
945 \pdfglyphtounicode{unionmultidisplay}{2A04 FE02}
946 \pdfglyphtounicode{unionmultitext}{2A04 FE01}
947 \pdfglyphtounicode{unionsqdisplay}{2A06 FE02}
948 \pdfglyphtounicode{unionsqtext}{2A06 FE01}
949 \pdfglyphtounicode{uniontext}{22C3 FE01}
950 \pdfglyphtounicode{vextenddouble}{20E6}% was {ED79}% PUA
951 \pdfglyphtounicode{vextendsingle}{20D3}%% was {23D0}
952 </glyphtounicode>

```

Change History

1985-11-04 ltmath.dtx LaTeX2.09	<code>\mathversion</code> : Test if version defined added.	584
General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number.		900
1989-04-10 ltfssbas.dtx v1.0a		
General: Starting with version numbers! <code>\ifmmode</code> added in <code>\math@group</code>		571
1989-04-10 ltfssbas.dtx v1.0b		
General: <code>\preload@sizes</code> added.		571
<code>\wrong@fontshape</code> changed to define substitution font/shape macro.		571
1989-04-10 ltfssini.dtx v1.0a		
General: Starting with version numbers <code>\newif</code> for <code>\@tempwa</code> added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) <code>\math@famname</code> changed to <code>\math@version</code>		735
1989-04-14 ltfssbas.dtx v1.0c		
General: More documentation added.		571
1989-04-15 ltfssini.dtx v1.0b		
General: <code>\mathfontset</code> renamed to <code>\mathversion</code>		735
1989-04-19 ltfssbas.dtx v1.0d		
General: Even more doc.		571
1989-04-21 ltfssbas.dtx v1.0e		
General: Documentation is fun! Parameters of <code>\define@mathalphabet</code> changed.		571
1989-04-21 ltfssini.dtx v1.0c		
General: Changed to conform to fam.tex.		735
1989-04-23 ltfssbas.dtx v1.0f		
General: % in <code>\getanddefinefonts</code> added.		571
1989-04-26 ltfssini.dtx v1.0d		
General: <code>\xpt</code> added.		735
1989-04-27 ltfssbas.dtx v1.0g		
General: Documentation revised.		571
1989-04-27 ltfssini.dtx v1.0e		
General: Definitions of L ^A T _E X symbols corrected.		735
1989-04-29 ltfssbas.dtx v1.0h		
General: Documented problem with <code>\halign</code> , and <code>\noalign</code>		571
	<code>\mathversion</code> : Test if version defined added.	584
1989-04-29 ltfssbas.dtx v1.0i		
General: Removed the <code>\halign</code> <code>\noalign</code> correction (wasn't bugfree)		571
1989-04-29 ltfssini.dtx v1.0f		
General: Corrections to L ^A T _E X tabular env. added.		735
1989-05-01 ltfssbas.dtx v1.0j		
General: Default for <code>\baselinestretch</code> added.		571
1989-05-22 ltfssbas.dtx v1.0k		
General: Lines longer than 72 characters folded.		571
1989-05-22 ltfssini.dtx v1.0g		
General: Lines shortened to 72 characters		735
1989-09-14 ltfssbas.dtx v1.0m		
General: Global replacement: <code>\group</code> to <code>\mathgroup</code>		571
<code>\mathversion</code> : Corrected typo: <code>\endscname</code> to <code>\endcsname</code>		584
1989-11-07 ltfssini.dtx v1.0i		
General: All family, series, and shape names abbreviated.		735
1989-11-08 ltfssbas.dtx v1.0o		
General: First parameter of <code>\define@mathalphabet</code> and <code>\define@mathgroup</code> changed from string to control sequence.		571
1989-11-14 ltfssbas.dtx v1.0p		
<code>\math@version</code> : Math version prefix 'mv@' added.		584
1989-11-19 ltfssbas.dtx v1.0q		
<code>\define@newfont</code> : Group added.		587
<code>\wrong@fontshape</code> : Instead of calling <code>\family\default@family</code> , etc. we directly set <code>\f@family</code> , etc.		592
1989-11-22 ltfssbas.dtx v1.0r		
<code>\math@version</code> : <code>\def</code> → <code>\edef</code> for <code>\math@version</code>		584
1989-11-25 ltfssbas.dtx v1.0s		
General: All <code>\edef\font@name</code> changed to <code>\xdef\font@name</code> . Necessary after introduction of <code>\begingroup/\endgroup</code> in v1.0q. extra// → + in <code>\extra@def</code>		571

1989-11-26 ltfssbas.dtx v1.0t	1990-01-25 ltfssini.dtx v1.1e
<code>\select@group</code> : <code>\bgroup</code> / <code>\egroup</code>	<code>\nfss@text</code> : Macro added. 760
changed to	1990-01-27 ltfssbas.dtx v1.2d
<code>\begingroup</code> / <code>\endgroup</code> to avoid	<code>\DeclarePreloadSizes</code> : Font identifier
empty Ord atom on math list. . . 595	set to <code>\relax</code> 579
1989-12-02 ltfssini.dtx v1.1b	1990-01-28 ltfssbas.dtx v1.2e
General: <code>\rmmath</code> renamed to	<code>\mathgroup</code> : <code>\newfam</code> let to
<code>\mathrm</code> 735	<code>\new@mathgroup</code> 571
1989-12-03 ltfssini.dtx v1.1c	1990-01-28 ltfssbas.dtx v1.2f
General: Some internal macros	<code>\define@newfont</code> : Added call to
renamed to make them	<code>\curr@fontshape</code> macro to allow
inaccessible. 735	substitution. 588
1989-12-05 ltfssbas.dtx v1.10u	<code>\wrong@fontshape</code> : Warning message
<code>\addto@hook</code> : <code>\addto@hook</code> added. . 599	slightly changed. 592
1989-12-05 ltfssstrc.dtx v1.0u fam.dtx	1990-01-28 ltfssini.dtx v1.2b
<code>\every@math@size</code> : Hook <code>\every@size</code>	<code>\em</code> : Call to <code>\@nomath</code> added. 758
added. 680	1990-02-08 ltfssini.dtx v1.1g
1989-12-13 ltfssstrc.dtx v1.0f	General: Protected the commands
<code>\use@mathgroup</code> : <code>\expandafter</code> added	<code>\family</code> , <code>\series</code> , <code>\shape</code> , <code>\size</code> ,
before final <code>\fi</code> 683	<code>\selectfont</code> , and <code>\mathversion</code> . 735
1989-12-16 ltfssbas.dtx v1.1a	1990-02-16 ltfssbas.dtx v1.2g
<code>\select@group</code> : <code>\relax</code> in front	General: Support for changes of
added. 595	<code>\baselineskip</code> without changing
Now four arguments. 595	the size. 571
Redefinition of alphabet now	<code>\math@version</code> : <code>\@nomath</code> added. . . 584
simpler. 595	1990-02-18 ltfssstrc.dtx v1.0j
Usage of ‘=’ macro added. 595	<code>\selectfont</code> : Redefine unprotected
1989-12-16 ltfssstrc.dtx v1.1a	version <code>\p@selectfont</code> instead of
<code>\selectfont</code> : Changed order of calls. 675	<code>\selectfont</code> 675
<code>\use@mathgroup</code> : Redefinition of	1990-03-14 ltfssstrc.dtx v1.0k
alphabet now simpler. 682	General: Added code for TeX3. . . . 671
Usage of ‘=’ macro added. 682	<code>\extract@font</code> : Added code for
1990-01-18 ltfssstrc.dtx v1.0h	TeX3. 674
General: <code>\tracingfonts</code> meaning	1990-03-30 ltfssbas.dtx v1.2h
changed. 671	<code>\math@egroup</code> : Changed to have one
1990-01-20 ltfssbas.dtx v1.2a	arg. 597
<code>\math@bgroup</code> : Def. placed in this	1990-03-30 ltfssstrc.dtx v1.2h
file. 597	<code>\use@mathgroup</code> : Third argument
<code>\math@egroup</code> : Def. placed in this	removed (see <code>\math@egroup</code>). . . 682
file. 597	1990-04-01 ltfssbas.dtx v1.2i
<code>\select@group</code> : Def for alph id	General: Code added from
changed. 595	tracefnt.dtx. 571
1990-01-21 ltfssbas.dtx v1.2b	Support for TeX3. 571
<code>\select@group</code> : Code moved to	1990-04-01 ltfssstrc.dtx v1.0l
<code>\use@mathgroup</code> 595	General: Part of code moved to
1990-01-21 ltfssstrc.dtx v1.2b	fam.dtx. 671
<code>\use@mathgroup</code> : Macro added to	<code>\tracingfonts</code> : Check if
allow cleaner interface. 682	<code>\tracingfonts</code> already defined. . 672
1990-01-23 ltfssbas.dtx v1.2c	1990-04-01 ltfssstrc.dtx v1.0o
General: <code>\no@version@warning</code>	<code>\tracingfonts</code> : Check if
renamed to <code>\no@alphabet@error</code> . 571	<code>\tracingfonts</code> defined removed
Macro <code>\no@alphabet@help</code> added 571	again. 672
<code>\no@alphabet@error</code> : Changed to	
error call 571	

1990-04-02 ltfssini.dtx v1.1i	1991-08-14 ltpictur.dtx LaTeX2.09
General: <code>\input</code> of files now handled	General: (RmS) inserted extra braces
by docstrip. 735	around entry for NFSS 979
1990-04-05 ltfssstrc.dtx v1.0m	1991-08-14 ltthm.dtx LaTeX2.09
<code>\selectfont</code> : Call <code>\tracingon</code> only if	<code>\endtheorem</code> : Moved <code>\itshape</code> after
<code>\tracingfonts</code> greater than 3. . . 675	<code>\item</code> to make it work with
1990-05-05 ltfssstrc.dtx v1.0n	NFSS 1010
<code>\selectfont</code> : <code>\tracingon</code> with new	1991-08-26 ltfssini.dtx v1.1n
syntax. 675	<code>\reset@font</code> : Macro introduced . . . 761
1990-06-23 ltfssini.dtx v1.1k	1991-08-26 ltmiscen.dtx LaTeX2.09
<code>\nfss@text</code> : Changed to <code>\mbox</code> 761	<code>\verbatim</code> : <code>\@par</code> added 882
1990-06-24 ltfssbas.dtx v1.2j	1991-08-26 ltpictur.dtx LaTeX2.09
<code>\DeclarePreloadSizes</code> : Missing	<code>\endpicture</code> : (RmS & FMI) extra
percent added. 579	boxing level around <code>\picbox</code> to
1990-06-24 ltfssstrc.dtx v1.0o	guard against unboxing in math
<code>\baselinestretch</code> : Moved to	mode (proposed by John Hobby) 977
tracefnt.dtx. 680	1991-08-26 ltplain.dtx LaTeX2.09
<code>\getanddefine@fonts</code> : <code>\Adding</code>	<code>\tracingall</code> : Added
tracing code. 684	<code>\errorcontextlines=\maxdimen</code> ,
<code>\Macro</code> moved from fam.dtx. . . . 683	suggested by J. Schrod 31
Adding debug code. 684	1991-09-29 ltboxes.dtx LaTeX2.09
<code>\use@mathgroup</code> : Tracing code added. 683	<code>\mpfootnotetext</code> : (RmS) added
1990-06-30 ltfssbas.dtx v1.2l	<code>\reset@font</code> 941
<code>\showhyphens</code> : Macro added. 598	1991-09-29 ltfloat.dtx LaTeX2.09
1990-06-30 ltfssstrc.dtx v1.0p	<code>\@footnotetext</code> : (RmS) added
<code>\use@mathgroup</code> : Added <code>\relax</code> after	<code>\reset@font</code> 1047
math group number. 683	1991-09-29 ltmath.dtx LaTeX2.09
1990-07-07 ltfssstrc.dtx v1.0q	<code>\eqnum</code> : RmS: <code>\reset@font</code> added. 900
<code>\getanddefine@fonts</code> : Group number	1991-09-29 ltsect.dtx LaTeX2.09
added to tracing. 684	<code>\dottedtocline</code> : (RmS) added
<code>\math@egroup</code> : Tracing code added. 683	<code>\reset@font</code> for page number . . 1024
<code>\use@mathgroup</code> : Group number	1991-10-17 ltcntrl.dtx LaTeX2.09
added to tracing. 683	<code>\@tfor</code> : (Rms) <code>\xdef</code> replaced by <code>\def</code>
1990-08-27 ltfssstrc.dtx 1.0r	(See FMI's array.doc) 426
<code>\type@restoreinfo</code> : Some extra	1991-10-25 ltbibl.dtx LaTeX2.09
tracing info. 679	<code>\citex</code> : added <code>\reset@font</code> ,
1990-08-27 ltfssstrc.dtx v1.0r	suggested by Bernd Raichle. . . 1055
<code>\getanddefine@fonts</code> : Correcting	1991-11-01 ltfloat.dtx LaTeX2.09
missing name after <code>\tracingon</code> . . 684	<code>\footnote</code> : (RmS) Added
1991-03-28 ltfssini.dtx v1.1m	<code>\let\protect\noexpand</code> in
<code>\copyright</code> : Extra braces added. . . 761	<code>\footnote</code> , <code>\footnotemark</code> , and
1991-03-30 ltfssini.dtx v1.2g	<code>\footnotetext</code> , since <code>\xdef</code> is
<code>\newfont</code> : Definition added. 759	used 1046
<code>\symbol</code> : Definition added. 760	1991-11-04 ltlists.dtx LaTeX2.09
1991-07-24 ltmiscen.dtx LaTeX2.09	<code>\makelabel</code> : (RmS) added default
<code>\@verbatim</code> : Added	definition for <code>\makeatlabel</code> , to
<code>\penalty\interlinepenalty</code> to	produce an error message. 921
definition of <code>\par</code> so that	1991-11-04 ltplain.dtx RmS
<code>\samepage</code> works 882	General: Removed <code>\itemitem</code> since
1991-08-14 ltmath.dtx LaTeX2.09	never needed/useful with L ^A T _E X. . . 29
<code>\cases</code> : (RmS) inserted extra braces	1991-11-06 ltbibl.dtx LaTeX2.09
around entry for NFSS 893	<code>\citex</code> : added code to remove a
	leading blank 1055

1991-11-13 ltbibl.dtx LaTeX2.09	avoid conflicts with other channels allocated by <code>\newread</code>	82
<code>\@bibitem</code> : Changed counter <code>enumi</code> to <code>enumiv</code> , as it says in the comment above	1992-03-18 ltfloat.dtx LaTeX2.09	
1991-11-21 ltfssini.dtx v1.1o	<code>\@xympar</code> : (RmS) added	
<code>\reset@font</code> : Added extra braces for robustness.	<code>\global\@ignorefalse</code>	1040
Changed to protected version of macro.	<code>\end@float</code> : (RmS) changed	
1991-11-22 ltfloat.dtx LaTeX2.09	<code>\@esphack</code> to <code>\@Esphack</code>	1034
<code>\footnote</code> : (RmS) Added	1992-03-18 ltlists.dtx 0.0	
<code>\let\protect\noexpand</code> in	<code>\trivlist</code> : RmS: added	
<code>\@xfootnote</code> , <code>\@xfootnotemark</code> , and <code>\@xfootnotetext</code>	<code>\@nmbrlistfalse</code>	916
1991-11-22 ltlists.dtx LaTeX2.09	1992-03-18 ltmiscen.dtx LaTeX2.09	
<code>\@item</code> : (RmS) Changed second call to <code>\makelabel</code> to	<code>\begin</code> : Changed <code>\@ignoretrue</code> to <code>\@ignorefalse</code> (as documented)	873
<code>\unhbox\@tempboxa</code> . Avoids problems with side effects in <code>\makelabel</code> and is more efficient.	1992-03-21 ltfssini.dtx v1.2d	
1991-11-27 ltfssbas.dtx v1.3a	General: Renamed <code>\text</code> to <code>\nfss@text</code> to make it internal.	735
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.	1992-05-12 ltfssbas.dtx v1.3c	
1991-11-27 ltfssini.dtx v1.2a	<code>\extract@alph@from@version</code> : Macro added.	596
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.	<code>\select@group</code> : Added call to <code>\extract@alph@from@version</code>	595
1992-01-06 ltfssini.dtx v1.2c	1992-07-26 ltfssbas.dtx v1.9a	
General: added <code>slitex</code> code	<code>\curr@fontshape</code> :	587
1992-01-10 ltbibl.dtx LaTeX2.09	<code>\DeclareFontShape</code> : Introduced	
<code>\@bibitem</code> : Changed <code>\c@enumiv</code> to <code>\value of \@listctr</code>	<code>\DeclareFontShape</code>	572
1992-01-10 ltmath.dtx LaTeX2.09	<code>\define@newfont</code> :	587
<code>equation</code> : RmS: put <code>\hbox</code> around <code>\@eqnnum</code> to typeset the equation number in text mode (as in the <code>eqnarray</code> env.)	<code>\math@fonts</code> :	594
1992-01-10 ltthm.dtx LaTeX2.09	<code>\select@group</code> :	595
<code>\@othm</code> : (RmS) Check for existence of theorem environment	<code>\split@name</code> : Added splitting into <code>\f@encoding</code>	587
1992-01-14 ltbibl.dtx LaTeX2.09	<code>\wrong@fontshape</code> :	592
<code>\@biblabel</code> : removed <code>\hfill</code>	1992-07-26 ltfssstrc.dtx v2.0b	
1992-01-14 ltsect.dtx 0.0	<code>\s@fct@</code> :	693
<code>\@starttoc</code> : (RmS) added <code>\immediate</code> to <code>\openout</code> as all <code>\write</code> commands are also executed	<code>\s@fct@sub</code> : documentation fixes	694
1992-02-26 ltbibl.dtx LaTeX2.09	<code>\selectfont</code> :	676
<code>\@lbibitem</code> : Added <code>\hfill</code> to restore left-alignment of bibliography labels in alpha style	<code>\try@simple@size</code> :	687
1992-03-18 ltdefns.dtx LaTeX2.09	<code>\try@size@range</code> :	690
General: (RMS) changed input channel from 0 to <code>\@inputcheck</code> to	<code>\use@mathgroup</code> :	683
	1992-08-14 ltbibl.dtx LaTeX2.09	
	<code>\@citex</code> : added missing argument braces around <code>\hbox</code> , found by Ed Sznyter	1055
	1992-08-14 ltboxes.dtx LaTeX2.09	
	<code>\endminipage</code> : (RmS) replaced <code>\vskip</code> - <code>\lastskip</code> by <code>\unskip</code> (proposed by FMI)	940
	1992-08-17 ltbibl.dtx LaTeX2.09	
	<code>\@citex</code> : simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup)	1055
	1992-08-19 ltsect.dtx 0.0	
	<code>\@xsect</code> : (RmS) corrected bug: stretch and shrink in argument to <code>\hskip</code>	

previously not negated	1017	<code>\footnote:</code> (RmS) Changed all to 'def'protect'noexpand'protect'noexpand	1046
1992-08-19 ltthm.dtx LaTeX2.09		1992-12-03 ltffsini.dtx v?	
<code>\@othm:</code> (RmS) Changed error message to complain about undefined counter	1009	<code>\hexnumber@:</code> Make it accept counters.	760
1992-08-20 ltffsini.dtx v1.4b		1993-03-08 preload.dtx v2.0b	
<code>\@setsize:</code> Added <code>\@currsize.</code> . . .	760	General: Added 12pt preloads	787
1992-08-24 ltdefs.dtx LaTeX2.09		1993-03-18 ltffsbas.dtx v2.0c	
<code>\@ifnextchar:</code> (Rms) <code>\@ifnextchar</code> didn't work if its first argument was an equal sign.	110	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	571
1992-08-24 ltmiscen.dtx LaTeX2.09		1993-03-18 ltfsstrc.dtx v2.1b	
<code>\begin:</code> Added code to <code>\begin</code> to remember line number. Used by <code>\@badend</code> to display position of non-matching <code>\begin.</code>	873	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	671
<code>\verb:</code> Changed <code>\verb</code> and <code>\@sverb</code> to work correctly in math mode . . .	887	Changed all <code>\@tempdimb</code> in <code>\@tempdimx</code> to avoid killing <code>\numberline</code>	671
1992-08-25 ltsect.dtx LaTeX2.09		1993-03-18 ltfsstrc.dtx v2.1c	
<code>\@sect:</code> (FMi) replaced explicit setting of <code>\@svsec</code> by call to <code>\@seccntformat</code>	1016	<code>\DeclareSizeFunction:</code> Added all args to avoid blanks problems . .	690
1992-09-18 ltlists.dtx LaTeX2.09		1993-04-09 lterror.dtx v1.0e	
<code>\item:</code> (RmS) Added warning if <code>\item</code> is used in math mode	919	<code>\@latexerr:</code> Mention The Companion	433
1992-09-18 lttab.dtx LaTeX2.09		1993-04-11 lterror.dtx v1.0f	
<code>\@array:</code> Changed <code>\par</code> to <code>\@empty</code> to avoid starting new row e.g. after <code>\hline</code>	960	<code>\@latexerr:</code> Remove setting of errorcontextlines	433
1992-09-19 ltfsstrc.dtx v2.0c		1993-05-05 ltftncmd.dtx v2.0b	
<code>\try@simple@size:</code>	687	General: Removed all LaTeX related cmds	790
1992-09-21 ltffsini.dtx v1.4d		1993-05-16 ltffsbas.dtx v2.0e	
<code>\not@math@alphabet:</code> Macro defined. .	759	<code>\showhyphens:</code> Use <code>\reset@font</code> . .	598
1992-09-22 ltffsbas.dtx v1.91a		1993-07-16 ltfsstrc.dtx v2.1h	
General: Introduced <code>\tf@size</code> for math size.	571	General: Changed layout of info messages	671
1992-09-22 ltfsstrc.dtx v2.1a		1993-07-17 ltoutenc.dtx 1.0d	
<code>\getanddefine@fonts:</code> Introduced <code>\tf@size</code> for math size.	684	General: changed <code>\catcoding @</code> . . .	512
1992-11-13 ltffsini.dtx v?		1993-08-03 ltmiscen.dtx LaTeX2.09	
<code>\hexnumber@:</code> Made expandable. . . .	760	<code>\enddocument:</code> Changed redefinition of <code>\global</code> to redefinition of <code>\@setckpt.</code>	865
1992-11-23 ltcounts.dtx LaTeX2.09		1993-08-05 ltpictur.dtx LaTeX2.09	
<code>\stepcounter:</code> Replaced <code>{}</code> in <code>\stepcounter</code> by <code>\begingroup</code> <code>\endgroup</code> to avoid adding an empty ord in math mode	559	<code>\circle:</code> (RMS) Added error message if <code>\circle</code> is used in math mode. .	999
1992-11-26 ltboxes.dtx LaTeX2.09		1993-08-05 ltsect.dtx LaTeX2.09	
<code>\@mpfootnotetext:</code> (RmS) added protection for <code>\edef</code>	941	<code>\@sect:</code> (RmS) Made sure that <code>\protect</code> works correctly in expansion of <code>\the</code> counter	1016
1992-11-26 ltfloat.dtx LaTeX2.09		1993-08-05 ltspaced.dtx LaTeX2e	
<code>\@footnotetext:</code> (RmS) added protection for <code>\edef</code>	1047	<code>\@hspace:</code> (RmS) Removed superfluous <code>\leavevmode</code> in <code>\@hspace</code> and <code>\@hspace</code> , as suggested by CAR.	481

1993-08-05 lttab.dtx latex2e		
\tabular*: Replaced		
\expandafter\def by \@namedef.	960	
1993-08-06 ltbibl.dtx LaTeX2.09		
\@citex: Moved writing to .aux file in		
loop over citation keys so that		
leading blanks are removed there		
as well.	1055	
1993-08-13 ltoutenc.dtx 1.0f		
General: Protected against active @		
sign.	512	
1993-08-13 preload.dtx v2.0c		
General: Added \relax at end of font		
names.	788	
1993-08-16 ltoutenc.dtx 1.0g		
General: Needs space after \string	512	
1993-08-18 ltfsdcl.dtx v2.0e		
\new@mathversion: Exchanged names		
of encodings in warning message of		
\SetSymbolFont.	716	
1993-09-02 ltfsstrc.dtx v2.1i		
General: Corrected name of sgen size		
function.	671	
1993-09-03 ltmiscen.dtx LaTeX2.09		
\verbatim@nolig@list: Replaced		
\@noligs by extensible list	887	
1993-09-07 ltmiscen.dtx LaTeX2.09		
\verb@balance@group: (RmS)		
Changed definition of \verb so		
that it detects a missing second		
delimiter.	887	
1993-09-08 ltmiscen.dtx LaTeX2.09		
\enddocument: Added warning in case		
of undefined references.	865	
1993-09-15 ltfsbas.dtx v2.0g		
\DeclareFontEncoding: Corrected:		
\default@T to \default@M.	575	
1993-09-15 ltfsstrc.dtx v2.1j		
General: Corrected spelling of		
\noxpand.	671	
1993-09-19 lterror.dtx LaTeX2.09		
\@invalidchar: (RmS) Error message		
for invalid input characters.	437	
1993-11-02 ltmath.dtx LaTeX2.09		
General: RmS: Corrected description		
of \@eqnset, moved \@eqnset		
accordingly and removed extra		
\tabskip assignment.	900	
1993-11-03 ltmath.dtx LaTeX2e		
General: RmS: Initialized \everycr to		
empty	900	
1993-11-03 ltpictur.dtx LaTeX2.09		
General: (RmS) changed \halign to		
\ialign to initialize \tabskip and		
\everycr	979	
1993-11-11 ltfsini.dtx v2.1a		
\normalfont: Macro added	761	
1993-11-11 ltfsstrc.dtx v2.2a		
General: Option concept added for		
LaTeX2e	671	
1993-11-14 ltclass.dtx v0.2a		
\@current: Name changed from		
\@currentextension	1100	
\@reset@ptions: macro added	1128	
\AtEndDocument: Included extension		
in the generated macro name for		
package and class hooks.	1129	
\documentstyle: Added		
\RequirePackage		
\@unusedoptionlist stuff.	1116	
\load@onefilewithoptions: Moved		
resetting of \default@ds, \ds@ and		
\@declaredoptions here, from the		
end of \ProcessOptions.	1122	
\NeedsTeXFormat: made more robust		
for alternative syntax for other		
formats.	1118	
\ProcessOptions*: Optimize 'empty		
option' code.	1113	
Stop adding the global option list		
inside class files.	1113	
1993-11-14 ltdefs.dtx v0.2a		
\g@addto@macro: Made global	114	
1993-11-15 ltclass.dtx v0.2b		
\documentstyle: Modified to match		
\ProcessOption*	1116	
\ProcessOptions*: Star form		
added.	1113	
1993-11-17 ltclass.dtx v0.2c		
\@fileswith@ptions: Macro added	1128	
\@badrequireerror: Macro added	1130	
\@twoloadclasserror: Macro added	1131	
\CurrentOption: Name changed from		
\@curroption	1099	
\DeclareOption*: Error checking		
added	1112	
\load@onefilewithoptions: Added		
trap for two \LoadClass		
commands.	1124	
\NeedsTeXFormat: Name changed from		
\NeedsFormat	1118	
\ProcessOptions*: restoring		
\@fileswith@ptions added.	1113	
1993-11-18 ltclass.dtx v0.2d		
\documentstyle: Modified		
\RequirePackage stuff.	1116	

\ExecuteOptions: Use		\newcommand: Macro added	84
\CurrentOption not		\newenv: Macro interface changed . .	88
\reserved@a	1116	\xargdef: Macro interface changed .	84
\NeedsTeXFormat: \fmtname		\yargdf: Avoid \@? token	85
\fmtversion not \@.	1118	Macro interface changed	85
1993-11-21 ltfiles.dtx LaTeX2e		\newcommand: Macro reimplemented	
\@missingfileerror: Stop infinite		and extended	84
looping on \@er@ext	505	\renewcommand: Macro reimplemented	
1993-11-21 ltmiscen.dtx v0.9a		and extended	87
\@verbatim: use \@verbatim@font		\renewenvironment: Macro	
instead of \tt	882	reimplemented and extended	88
\verb: Use \@verbatim@font instead of		\two@digits: Macro added	80
\tt.	887	1993-11-23 ltoutput.dtx v0.1a	
\verbatim@font: Macro added	883	\paperheight: Register added . . .	1226
1993-11-22 ltclass.dtx v0.2f		\paperwidth: Register added . . .	1226
\@fileswithoptions: Made the		1993-11-23 ltoutput.dtx v0.1c	
default [] not		\@enlargepage: Command added .	1290
[\@unknownversion]	1120	\@kludgeins: Insert added	1289
\@ifl@ter: Added //00 so parsing		\@makecol: Command changed . . .	1241
never produces a runaway		\@specialoutput: Command	
argument.	1105	changed	1235
General: \@unknownversion		\enlargethispage*: Commands	
removed	1094	added	1289
\load@onefilewithoptions: Made the		1993-11-24 ltfntcmd.dtx v2.1a	
initial version [] not		\maybe@ic: Use \t@st@ic	795
[\@unknownversion]	1122	\t@st@ic: Macro added	796
1993-11-22 ltdefs.dtx LaTeX2e		1993-11-24 ltssini.dtx v2.1a	
\@minus: Macro added	81	General: Removed \xpt stuff	761
\@plus: Macro added	81	1993-11-24 ltlogos.dtx LaTeX2e	
\CheckCommand: Macro added	89	\LaTeX: Macro changed	484
\providecommand: Macro added	89	1993-11-28 ltclass.dtx v0.2h	
1993-11-22 lterror.dtx LaTeX2e		\@twoclasseserror: Macro added	1131
\@errorcontextlines: Macro added	433	General: Assorted commands now in	
1993-11-22 ltfiles.dtx LaTeX2e		the kernel removed.	1098
\listfiles: Removed checking for		Directory syntax checking moved to	
\@unknownversion	507	dircheck.dtx	1098
1993-11-22 ltlength.dtx LaTeX2e		Primitive filenames now terminated	
\@settodim: Macro added	569	by space not \relax.	1098
\@settopoint: Macro added	570	\endfilecontents: Don't globally	
\settodepth: Macro added	569	allocate a write stream (always use	
\settoheight: Macro added	569	15)	1131
1993-11-22 ltlogos.dtx LaTeX2e		1993-11-28 ltfiles.dtx LaTeX2e	
\LaTeXe: Macro added	484	\@missingfileerror: Use filename	
1993-11-23 ltclass.dtx v0.2g		parser from dircheck	505
\@use@option: Name changed from		1993-11-29 ltoutput.dtx v1.0b	
\@executeoption	1115	\@makecol: \@makespecialcolbox	
General: Various macros now moved		added	1241
to latex.tex.	1098	1993-11-29 ltplain.dtx LaTeX2e	
Warnings and errors now directly		General: All accents in decimals;	
coded.	1098	suggested by Paul Taylor	30
1993-11-23 ltdefs.dtx LaTeX2e		1993-11-30 ltoutput.dtx v1.0c	
\@argdef: Macro added	84	\fl@tracemessage: Commands	
\@ifundefined: Redefined to remove a		added	1292
trailing \fi	108		

1993-12-01 fontdef.dtx v2.1a		1993-12-04 ltfiles.dtx v0.9b	
General: Update for LaTeX2e	766	\@iinput: Macro reimplemented . . .	504
1993-12-01 ltoutput.dtx v1.0e		\@input: Macro reimplemented . . .	504
\@reinserts: Command added . . .	1253	\IfFileExists@: Macro added	500
1993-12-03 ltboxes.dtx v0.1a		\input: Macro reimplemented	503
\@argrsbox: macro removed	944	1993-12-05 ltfloat.dtx LaTeX2e	
\@begin@tempboxa: macro added . . .	927	\@dblfloatplacement: Command	
\@end@tempboxa: macro added	927	changed	1036
\@iirsbox: redefined to support		\@xfloat: Command changed . . .	1030
\height	944	1993-12-05 ltoutput.dtx v1.0f	
\@imakebox: macro modified	928	\@addtobot: Command changed . .	1266
\@irsbox: redefined to support		\@addtocurcol: Command changed	1268
\height	944	\@addtodblcol: Command changed	1283
\@isavebox: color support	931	\@addtonextcol: Command changed	1278
extra group	931	\@addtotoporbot: Command	
\@isavepicbox: extra group	931	changed	1267
\@makebox: default changed from x to		\@boxfpsbit: Command added . .	1296
c	927	\@flcheckspace: Command added	1298
\@makepicbox: macro modified	928	\@flsetnum: Command added . . .	1297
\@savebox: default c not x	931	\@flsettextmin: Command added	1297
\bm@b: macros added	927	\@flstop: Commands added	1293
\endlrbox: macro added	932	\@flupdates: Command added . .	1300
\fbbox: extra group	933	\@fpsadddefault: Command added	1294
\lrbox: color support	932	\@getfpsbit: Command added . .	1295
macro added	932	\@opcol: Command changed	1240
\makebox: modified	926	Hook added	1240
\mbox: extra group	927	\@outputpage: Command changed	1254
\minipage: Redefined to support extra		\@resetfps: Command added . .	1296
optional arguments	939	\@setfloattypecounts: Command	
\newsavebox: Pass the whole of arg 1		added	1295
to \@ifdefinable	930	\@setfpsbit: Command added . .	1296
\parbox: Redefined to support extra		\@shipoutsetup: Command added	1254
optional arguments	935	\@startcolumn: Command changed	1262
\raisebox: redefined to support		\@startdblcolumn: Command	
\height	944	changed	1262
\sbox: color support	931	\@testfp: Command added	1296
extra group	931	\@textfloatsheight: Commands	
\set@color: color support	929	added	1294
macro added	929	\@topnewpage: Commands changed	1230
1993-12-03 ltclass.dtx v0.2i		\@tryfcolumn: Command changed	1263
\@cls@pkg: Name changed to avoid		\@writesetup: \@startpagehook	
clash with output routine.	1129	added	1254
General: \@onlypreamble: Many		\output: Command changed	1235
commands declared.	1098	1993-12-06 ltclass.dtx v0.2k	
Removed obsolete		\ExecuteOptions: Preserve	
\@documentclass	1098	\CurrentOption.	1116
1993-12-03 lterror.dtx v1.0b		1993-12-06 ltoutput.dtx v1.0f	
\@latexerr: Set		\@specialoutput: Unboxing of 255	
\@errorcontextlines to -1 . . .	433	added to rescue writes	1235
1993-12-03 ltffsini.dtx v2.1a		1993-12-06 ltoutput.dtx v1.0g	
General: update for LaTeX2e	735	\@topnewpage: \@floatplacement	
1993-12-04 ltfilehook.dtx v0.9b		placement bug fixed	1230
\unqu@tefilef@und: Macro added	1170	1993-12-07 ltclass.dtx v0.2l	
		\ProvidesFile: Macro added	1110

1993-12-07 ltclass.dtx v0.2m		<code>\fix@penalty</code> : Macro added	796
<code>\load@onefilewithoptions</code> : Reset		<code>\maybe@ic</code> : Macro name changed . .	795
<code>\CurrentOption</code>	1122	<code>\maybe@ic@</code> : Macro and name	
1993-12-07 ltoutenc.dtx 1.1		changed	795
General: Protected all special		<code>\sw@slant</code> : Macro changed	796
characters with <code>\string</code>	512	<code>\textup</code> : Macros changed	793
1993-12-07 ltoutenc.dtx v1.1		1993-12-11 ltmath.dtx v0.9g	
General: Made all character numbers		General: Added a group around the	
decimal.	509	first argument of <code>\frac</code> to prevent	
Removed a lot of equal signs and		changes (for example font changes)	
the like.	509	from modifying the contents of the	
1993-12-08 ltboxes.dtx v0.1b		second argument.	900
<code>\@begin@tempboxa</code> : Extra braces for		1993-12-11 ltoutenc.dtx v1.2a	
color support (braces removed		General: Corrected for <code>tlenc</code> , <code>math</code> . .	509
from other macros)	927	1993-12-11 ltsect.dtx LaTeX2e	
<code>\@irsbox</code> : fix typo	944	<code>\author</code> : Added default	1012
<code>\@parboxto</code> : <code>\endgraf</code> added due to		<code>\title</code> : Added default	1012
extra group in <code>\@begin@tempboxa</code>	936	1993-12-11 ltxref.dtx LaTeX2e	
<code>\lrbox</code> : move <code>\@endpfalse</code> out of the		<code>\setref</code> : Macro added	842
inner group	932	<code>\pageref</code> : Macro reimplemented . . .	842
1993-12-08 ltfntcmd.dtx v2.1b		<code>\ref</code> : Macro reimplemented	842
General: Macros <code>\rm</code> , <code>\bf</code> and <code>\sf</code>		1993-12-12 ltoutput.dtx v1.0h	
moved to <code>classes.dtx</code>	798	<code>\@cflb</code> : <code>boxmaxdepth</code> setting moved	1260
1993-12-08 ltlists.dtx LaTeX2e		defs changed to <code>lets</code>	1260
<code>\@item</code> : use <code>\sbox</code> to support color .	921	<code>\@cflt</code> : name changed	1260
1993-12-08 ltspace.dtx LaTeX2e		<code>\@doclearpage</code> : defs changed to <code>lets</code>	1240
<code>\@bsphack</code> : Command reimplemented	470	<code>\@makecol</code> : defs changed to <code>lets</code> . .	1241
Command reimplemented; late		<code>\@resetthfps</code> : Warnings added:	
birthday present for Chris	470	minimal	1296
<code>\@vbsphack</code> : Command added	473	<code>\@startdblcolumn</code> : defs changed to	
1993-12-09 ltboxes.dtx v0.1c		<code>lets</code>	1262, 1263
<code>\@irsbox</code> : fix another typo	944	<code>\@topnewpage</code> : braces removed . . .	1230
1993-12-09 ltclass.dtx v0.2n		<code>\@tryfcolumn</code> : defs changed to <code>lets</code>	1263
<code>\documentstyle</code> : input 209		<code>\fl@tracemessage</code> : Commands	
compatibility file.	1116	changed	1292
1993-12-09 ltfiles.dtx v0.9e		1993-12-13 ltclass.dtx v0.2o	
<code>\document</code> : Hook added	487	General: Removed setting	
1993-12-09 ltmiscen.dtx v0.9e		<code>\errorcontextlines</code> (now in	
<code>\enddocument</code> : Hook added	865	<code>latex.tex</code>)	1098
1993-12-10 ltoutenc.dtx v1.2		<code>\documentstyle</code> : compatibility file	
General: Added source code for		now <code>latex209.sty</code>	1116
<code>tlenc.sty</code>	509	<code>\usepackage</code> : Fixed error handling	1118
1993-12-11 ltfntcmd.dtx v3.0a		1993-12-13 ltdirchk.dtx v0.2a	
General: Complete reworking of all		General: on the ‘docstrip’ pass, do not	
text commands, using just one		check <code>openin</code> path	10
creator function	790	<code>\IfFileExists</code> : Removed interactive	
italic correction now put in front of		prompting for current directory	
penalty before glue	790	syntax	10
newcommands replaced by defs . .	790	<code>\strip@prefix</code> : modified, name	
newfontswitch command corrected		changed from <code>\stripmeaning</code> . . .	5
and changed	790	1993-12-13 ltlists.dtx latex2e	
<code>\DeclareTextFontCommand</code> : Macro		<code>\trivlist</code> : Initialized <code>\@itemlabel</code>	916
changed	792	1993-12-13 ltmiscen.dtx v0.9h	
<code>\emph</code> : Macro changed	793	<code>\@noligs</code> : Readded <code>\@noligs</code>	888

<code>\@verbatim</code> : Readded <code>\@noligs</code> . . .	882	1993-12-17 ltoutenc.dtx 1.3	
Removed optional argument of		General: Added this section	513
<code>\item</code>	882	Removed all the hackery for use in	
<code>center</code> : Removed optional argument		<code>\DeclareFontEncoding</code> , and redid	
of <code>\item</code>	879	everything using	
<code>flushleft</code> : Removed optional		<code>\DeclareTextFoo</code>	525, 528
argument of <code>\item</code>	880	Removed the catcode hackery, since	
<code>flushright</code> : Removed optional		the file is only read as a package in	
argument of <code>\item</code>	881	the preamble, and removed all the	
1993-12-13 ltoutenc.dtx v1.2b		messages on the screen, which just	
General: Corrected file name in driver		confuse users. Replaced them by	
code.	509	the appropriate <code>\ProvidesPackage</code>	
1993-12-13 lttab.dtx latex2e		commands. Added XXXenc.	512
<code>\tabbing</code> : Removed optional argument		1993-12-17 ltoutenc.dtx v1.3	
of <code>\item</code>	954	General: Added	
1993-12-14 ltoutput.dtx v1.0i		<code>\EncodingSpecificAccent</code> ,	
General: Section added to declare all		<code>\EncodingSpecificAccentedLetter</code>	
parameters	1305	and <code>\EncodingSpecificCommand</code>	509
1993-12-15 ltboxes.dtx v0.1d		Made Rokicki's encoding a proper	
<code>\@iminipage</code> : Changed default from		encoding scheme rather than a	
'c' to 's'	939	variant of OT1.	509
<code>\@iparbox</code> : Changed default from 'c'		1993-12-17 ltoutput.dtx v1.0j	
to 's'	935	<code>\@opcol</code> : Hook removed	1240
<code>\minipage</code> : Changed default from 'c'		<code>\@specialoutput</code> : Page room test	
to 's'	939	added	1235
extra space removed.	939	<code>\@topnewpage</code> : check for vsize too	
<code>\parbox</code> : Changed default from 'c' to		small added	1230
's'	935	Page room test added	1231
1993-12-15 ltclass.dtx v0.2p		<code>\@writesetup</code> : —and then removed	1254
General: Removed extra 's from		<code>\fl@tracemessage</code> : tracefloatvals	
<code>\@warnings</code>	1098	made a document command	1292
1993-12-16 ltlogos.dtx LaTeX2e		1993-12-17 ltpage.dtx LaTeX2e	
<code>\LaTeXe</code> : Extended logo by DPC . .	484	<code>\mark</code> : Removed init <code>\mark</code> at begin	
1993-12-16 ltmath.dtx v0.9i		document, since it doesn't work.	1092
<code>\@eqnocr</code> : use <code>\refstepcounter</code>		<code>\rightmark</code> : Stopgap solution to mark	
instead of shortcut	902	<code>\leftmark</code> and <code>\rightmark</code> work	
General: use <code>\refstepcounter</code> instead		without initializing mark until the	
of shortcut	900	problem is solved.	1092
1993-12-16 ltmiscen.dtx v0.9i		1993-12-18 ltoutenc.dtx 1.3b	
General: <code>\literal</code> added	888	General: Fixed typos with	
1993-12-16 ltpage.dtx LaTeX2e		<code>\ProvidesPackage</code> lines. Added	
<code>\mark</code> : Init <code>\mark</code> at begin document	1092	the <code>\NeedsTeXFormat</code> line. Added	
1993-12-16 ltspace.dtx LaTeX2e		the last argument to	
<code>\@bsphack</code> : Corrected optimisation .	470	<code>\DeclareEncoding</code> . Moved the use	
1993-12-16 lttab.dtx latex2e		of the encodings to after their	
<code>\@xhline</code> : Measure from middle of		declaration.	512
vertical rules	971	Replaced the missing last argument	
1993-12-17 ltclass.dtx v0.2q		to <code>\DeclareFontEncoding</code>	525, 528
<code>\@documentclasshook</code> : Macro added	1099	1993-12-18 ltoutenc.dtx 1.3c	
<code>\@fileswithoptions</code> : Add		General: Rewrote for the new syntax	
<code>\@compatibility</code> hook	1120	of <code>\EncodingSpecific</code>	525, 528
<code>\documentstyle</code> : Match Alan's new		Split <code>\EncodingSpecificAccent</code> up	
code.	1116	into <code>\EncodingSpecific</code> and	
		<code>\DeclareAccent</code>	513

1993-12-18 ltoutenc.dtx v1.3a	1994-01-14 ltdirchk.dtx v0.2d
General: Replaced OT3 by XXX . . . 509	\IfFileExists: Close the texsys.aux
1993-12-18 ltoutenc.dtx v1.3b	output stream 10
General: Corrected typos. 509	1994-01-15 ltfiles.dtx v0.9o
Replaced the missing last argument	\document: move \@preamblecmds
to \DeclareFontEncoding. 509	after document hook 489
1993-12-18 ltoutenc.dtx v1.3c	1994-01-17 ltclass.dtx v0.2s
General: A new syntax, separating	\@fileswithoptions: Modify to
accent-definitions from	reduce parameter stack usage . . 1120
encoding-specific definitions, and	General: Added many more
allowing encoding-specific	\@onlypreamble commands . . . 1098
\chardef, \let, etc. 509	Wrapped long lines to column 72 1098
Rewrote for the new syntax of	\load@onefile@withoptions: Modify
\EncodingSpecific. 509	to reduce parameter stack usage 1125
1993-12-18 ltoutenc.dtx v1.3d	1994-01-17 ltfiles.dtx LaTeX2e
General: Some T1 stuff had drifted	\listfiles: New Version, adds 'tex'
into the OT1 file. 509	if needed, and lines up columns . 507
1993-12-18 ltpage.dtx LaTeX2e	1994-01-17 ltfsbas.dtx v2.1a
\sloppy: Added \emergencystretch 1093	General: New math font setup 571
1993-12-19 ltclass.dtx v0.2r	\curr@math@size: New math font
\endfilecontents: Different message	setup 586
when ignoring a file 1131	\everydisplay: New math font setup 585
1993-12-19 ltfntcmd.dtx v3.0b	\everymath: New math font setup . . 585
General: \@pdef command added . . 790	\frozen@everydisplay: New math
Added by ASAJ. 798	font setup 586
Made \@newfontswitch produce an	\frozen@everymath: New math font
error if command already exists,	setup 585, 586
and added \@renewfontswitch,	\math@version: New math font setup 584
ASAJ 790	1994-01-17 ltfsini.dtx v2.1e
Other tidying 790	\not@math@alphabet: Message
Some more tidying done 790	changed 759
Untidying added, so this is now a	1994-01-17 ltfsstrc.dtx v2.3a
TEMPORARY version. 790	General: New math font setup 671
Wording changes by CAR. 798	\check@mathfonts: New math font
\DeclareOldFontCommand: Corrected	setup 681
and tidied 797	\glb@currsize: New math font setup 679
\DeclareTextFontCommand: Corrected	\restglb@settings: New math font
and tidied 792	setup 682
1993-12-19 ltspace.dtx LaTeX2e	1994-01-18 ltbibl.dtx LaTeX2e
\@bsphack: There seem to be problems	\bibliography: Use \@input@ so
with selfmade birthday presents . 471	include files are listed. 1055
1993-12-20 ltdefs.dtx LaTeX2e	1994-01-18 ltclass.dtx v0.2t
\@reargdef: Kept old version of	\@ifclassloaded: Fix typo
\@reargdef, for array.sty 87	\@pkgetension 1104
1993-12-20 ltfiles.dtx v0.9m	1994-01-18 ltfilehook.dtx v0.9p
\@obsoletefile: Added this	\unqu@tefilef@und: New Definition 1170
command, removed	1994-01-18 ltfiles.dtx v0.9p
\@oldfilewarning 506	\@iffileonpath: Macro added 502
1994-01-05 fontdef.dtx v2.1d	\@input: do not use a different
General: Removed nf prefix from file	definition for \input@path 504
names. 768	\@input@: Macro added 505
1994-01-13 ltmath.dtx v0.9o	\IfFileExists@: New Definition . . 500
\@eqnocr: correcting 0.9i 902	\includeonly: Use \@input@ so
General: correcting 0.9i 900	include files are listed. 493

1994-01-18 ltffssini.dtx v2.1f	1994-01-25 ltffssbas.dtx v2.1b
<code>\not@math@alphabet</code> : Message corrected 759	<code>\math@version</code> : Corrections for math setup 585
1994-01-18 ltmiscen.dtx v0.9p	1994-01-25 ltmath.dtx LaTeX2e
<code>\@verbatim</code> : Add	<code>\bordermatrix</code> : Removed <code>\p@renwd</code> . 894
<code>\global\@inlabeledfalse</code> 882	1994-01-26 ltfsstrc.dtx v2.3c
Only add <code>\penalty</code> if in hmode 882	<code>\check@mathfonts</code> : Correct trace info placement 681
1994-01-19 fontdef.dtx v2.1e	<code>\restglib@settings</code> : Correct trace info placement 682
General: Added missing setting for symbols in bold version. 773	1994-01-27 ltfntcmd.dtx v3.1a
1994-01-19 ltdirchk.dtx v0.2e	<code>\nocorrlist</code> : Only <code>.</code> , used as default for cm fonts 797
<code>\IfFileExists</code> : name changed from <code>\test</code> 9	1994-01-29 ltclass.dtx v0.2v
<code>\input@path</code> : No longer check that an empty group is in the path 10	<code>\@unprocessedoptions</code> : Macro added. 1130
<code>\strip@prefix</code> : name changed from <code>\strip@meaning</code> , to match NFSS. 5	<code>\load@onefile@withoptions</code> : All options raise error if no <code>\ProcessOptions</code> appears 1126
1994-01-19 ltmath.dtx v1.0n classes	1994-01-31 ltdefns.dtx v0.2w
<code>\mathindent</code> : Deferred setting of <code>\mathindent</code> 904	<code>\g@addto@macro</code> : Use toks register to avoid ‘hash’ problems 114
1994-01-20 ltdirchk.dtx v0.2f	1994-01-31 ltfiles.dtx v0.9t
General: <code>\@copytexsys</code> and the <code>texsys.new</code> file removed 8	<code>\document</code> : set <code>\@normalsize</code> or <code>\normalsize</code> if necessary 488
Modify all of <code>ltxcheck</code> 13	1994-01-31 ltfntcmd.dtx v3.1b
<code>\IfFileExists</code> : <code>\@copytexsys</code> removed 10	General: <code>\@normalsize</code> no longer defined 790
1994-01-21 ltclass.dtx v0.2u	1994-02-01 ltpage.dtx LaTeX2e
<code>\documentstyle</code> : compatibility file now <code>latex209.def</code> 1116	<code>\pagestyle</code> : (DPC) Modify to get nicer error message 1089
1994-01-21 ltdirchk.dtx v0.2g	<code>\thispagestyle</code> : (DPC) Modify to get nicer error message 1090
General: Improve documentation, reorganize <code>docstrip</code> module 1	1994-02-02 ltclass.dtx v0.2x
<code>\filename@parse</code> : Minor changes, and add Mac version (<code>:</code>) 11	<code>\load@onefile@withoptions</code> : Only run the hook and options check if the file was loaded. 1126
<code>\today</code> : Name changed from <code>\stamp</code> , to save memory 9	1994-02-03 ltoutput.dtx v1.0k
1994-01-21 ltfloat.dtx LaTeX2e	<code>\@make@specialcolbox</code> : correct mistakes in the documentation . . 1244
<code>\xfloat</code> : Added missing percent characters. 1030	1994-02-07 ltclass.dtx v0.2y
1994-01-21 ltmiscen.dtx v0.9s	<code>\@fileswithoptions</code> : Run <code>\@compatibility</code> on the first class to start (not the first to finish) 1120
<code>\verbatim@font</code> : Removed unnecessary category code hackery. 883	<code>\@ifclasswith</code> : Add extra <code>,s</code> so ‘two’ is not matched with ‘twocolumn’ 1106
1994-01-24 ltdirchk.dtx v0.2h	<code>\ProcessOptions*</code> : Add extra <code>,s</code> so ‘two’ is not matched with ‘twocolumn’ 1113, 1114
<code>\IfFileExists</code> : Stop testing once <code>texsys.aux</code> has been found 9	1994-02-07 ltffssbas.dtx v2.1c
1994-01-24 ltpage.dtx LaTeX2e	<code>\DeclareFontEncoding</code> : revert catcode settings earlier 575
<code>\pagestyle</code> : (DPC) Complain if <code>pagestyle</code> is undefined. 1089	<code>\DeclareFontShape@</code> : revert catcode settings earlier 572
1994-01-25 ltdirchk.dtx v0.2i	
General: Protect against looping on <code>\@input</code> and <code>\@end</code> 3	

1994-02-08 ltoutput.dtx v1.0k		1994-03-04 ltvers.dtx v1.0a	
General: Documentation and tasks		General: Initial version, split from	
tidied.	1214	latex.dtx	35
1994-02-10 ltclass.dtx v0.2z		1994-03-07 ltboxes.dtx v0.1a	
\@documentclasshook: Changed the		\@mpfootnotetext: Extra group for	
name from \@compatibility to		color	941
\@documentclasshook, and added		1994-03-07 ltboxes.dtx v1.0a	
the check for whether		General: Unify format with other	
\@normalsize has been defined.		Kernel files	926
ASAJ.	1099	1994-03-07 ltdefs.dtx v1.0a	
\@fileswithoptions: Renamed		\@italiccorr: Macro added	81
\@compatibility to		1994-03-07 ltfiles.dtx v1.0a	
\@documentclasshook. ASAJ. .	1120	General: Initial version, split from	
1994-02-10 ltffsbas.dtx v2.1d		latex.dtx	485
\addto@hook: Made \addto@hook		Long lines wrapped to 72 columns	485
long.	599	1994-03-07 ltfinal.dtx v0.1a	
1994-02-10 ltffscmp.dtx v2.1d		General: Add code from the old	
\scan@fontshape: scan away stuff		dump.dtx	1350
after pt	698	Initial version, split from latex.dtx	1332
1994-02-22 ltffsini.dtx v2.1g		move code here from lhyphen.dtx	1339
General: Correct error message	764	Remove oldcomments	
1994-02-24 ltffsbas.dtx v2.1e		environment	1332
\DeclareFontShape: Separate		use \InputIfFileExists not	
restoration of catcodes for fd		\IfFileExists	1339
cmds	572	1994-03-07 ltfloat.dtx v1.0a	
\define@newfont: Separate		\@endfloatbox: (DPC) Extra group	
restoration of catcodes for fd		for color	1035
cmds	588	\@footnotetext: (DPC) Extra group	
\nffs@catcodes: Separate restoration		for color	1047
of catcodes for fd cmds	588	\@xfloat: (DPC) Extra group for	
1994-02-25 ltdirchk.dtx v0.2j		color	1031
General: Remove need for drv file	1	1994-03-07 lthyphen.dtx v0.1c	
1994-03-01 ltdirchk.dtx v0.2k		General: move the 2ekernel code to	
General: Add unstripped module, so		ltfinal.dtx	1330
that dircheck.dtx may be used		1994-03-07 ltlength.dtx v1.0a	
with initex	1	\@settodim: (DPC) Extra group for	
1994-03-02 ltboxes.dtx v0.1e		color	569
General: Add 2ekernel module	926	1994-03-07 ltlists.dtx v1.0a	
Remove need for drv file	926	General: Initial version, split from	
1994-03-02 ltclass.dtx v0.3a		latex.dtx	908
General: Remove need for driver file	1098	Long lines wrapped to 72 columns	908
1994-03-03 ltboxes.dtx v0.1f		1994-03-07 ltpage.dtx v1.0a	
\@irsbox: Replaced a missing \else	944	General: Initial version, split from	
1994-03-04 ltfloat.dtx v1.0a		ltherest.dtx	1089
General: Initial version, split from		1994-03-07 ltpictur.dtx v0.1a	
latex.dtx	1026	General: Initial version, split from	
1994-03-04 ltsect.dtx v1.0a		latex.dtx	974
General: Initial version, split from		Long lines wrapped to 72 columns	974
latex.dtx	1012	1994-03-07 ltsect.dtx v1.0a	
1994-03-04 lttab.dtx v1.0a		\@hangfrom: (DPC)Extra groups for	
General: Initial version, split from		color	1019
latex.dtx	947	1994-03-07 lttab.dtx v1.0a	
		General: Long lines wrapped to 72	
		columns	947

1994-03-08 ltclass.dtx v0.3b	1994-03-13 ltfiles.dtx LaTeX2e
General: Modify driver code into ‘new style’	\@addtofilelist: Macro added . . .
1994-03-08 ltdirchk.dtx v1.0a	\listfiles: Reset \@addtofilelist at begin document
General: Reorganize driver module into ‘new style’	1994-03-13 ltfssbas.dtx v2.1g
1994-03-08 ltplain.dtx v1.0a	General: add 2kernel module to omit repeated code
General: Remove need for a driver file.	1994-03-13 ltfssdcl.dtx v2.1c
1994-03-10 ltfssbas.dtx v2.2f	General: add 2kernel module to omit repeated code
\math@egroup: Changed	1994-03-14 ltboxes.dtx v1.0b
\begin@group/\endgroup to	\@isavebox: Use \color@setgroup
\bgroup/\egroup.	\@isavepicbox: Use
1994-03-11 ltfssdcl.dtx v2.1b	\color@setgroup
\DeclareSymbolFontAlphabet@:	\color@begin@group: macro added for color support
Added check against use of	\color@end@group: macro added for color support
alphabet switch outside of math mode.	\lrbox: Use \color@setgroup
\SetMathAlphabet@: Changed	\sbox: Use \color@setgroup
parameter template in temporary macro to catch check add below.	1994-03-14 ltfloat.dtx 1.0c
1994-03-12 ltclass.dtx v0.3c	\@xympar: (DPC) Use
General: Change name from docclass to ltclass	\color@begin@group
\ProvidesFile: Add \wlog	1994-03-14 ltfloat.dtx v1.0c
\ProvidesPackage: Add \wlog	\@endfloatbox: (DPC) Use
use \@gtempa	\color@end@group
1994-03-12 ltdefs.dtx v1.0b	\@footnotetext: (DPC) Use
\@reargdef: New defn, in terms of	\color@begin@group, add
\@yargdef	\endgraf
\@yargdef: Name changed from	\@savemarbox: (DPC) Use
\XXX@argdef	\color@begin@group
1994-03-12 ltdirchk.dtx v1.0b	\@xfloat: (DPC) Use
General: Change name from	\color@begin@group
dircheck.dtx	1994-03-15 ltfiles.dtx LaTeX2e
Minor edits to the typeouts in	\@missingfileerror: Quit on x or X just like a real error
ltxcheck	1994-03-15 ltfntcmd.dtx v3.2a
1994-03-12 ltfloat.dtx v1.0b	General: Adapted to mass formatting
\@savemarbox: (DPC) Extra group for color	Changed \/ to \@@italiccorr
\@xympar: (DPC) Extra bgroup for color	Removed \@renewfontswitch
1994-03-12 ltplain.dtx v1.0b	Removed defs of short-forms and all sizes except \normalize
General: Name changed from lplain. The end of an era	1994-03-15 ltoutput.dtx v1.0l
1994-03-12 ltplain.dtx v1.0e	\@addtocurcol: Changed \advspace to \vskip
General: Replaced remaining width, height, depth by L ^A T _E X macro names to save tokens.	\@combinedblfloats: Removed
1994-03-13 ltcntrl.dtx v1.0c	boxmaxdepth setting.
\@tfor: (DPC) Add \@tf@r so a single group is correctly treated.	\@make@normalcolbox: Removed
1994-03-13 ltfilehook.dtx v0.3b	boxmaxdepth setting.
\unqu@tefilef@und: Use new cmd	\@outputbox@append: \maxdepth changed to \@maxdepth
\@addtofilelist	\@topnewpage: Corrected and amended warning message

Warning added: it should be improved	1231	1994-03-28 ltthm.dtx v1.0a	General: Initial version, split from latex.dtx	1007
General: Added some warnings when page gets full of top floats. . . .	1214	1994-03-29 ltcounts.dtx v1.0c	General: Create file from parts of ltmiscen and ltherest.	556
Driver added and further tidying.	1214	1994-03-29 ltlength.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	569
Removed duplicated code and corrected docstrip options.	1214	1994-03-29 ltmiscen.dtx v1.0d	General: Remove counter macros to ltcntlen	864
Some boxmaxdepth settings removed.	1214	1994-03-29 ltpageno.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	839
1994-03-16 ltclass.dtx v0.3f		1994-03-29 ltxref.dtx v1.0c	General: Create file ltcntlen from parts of ltmiscen and ltherest.	840
General: Add pkgindoc package . . .	1147	1994-03-31 ltbibl.dtx v1.0a	General: Initial version of ltidxbib.dtx, split from ltherest.dtx	1053
1994-03-16 ltfiles.dtx LaTeX2e		1994-03-31 ltidxglo.dtx v1.0a	General: Initial version of ltidxbib.dtx, split from ltherest.dtx	1050
\listfiles: Move this code directly into \document	508	1994-04-09 ltcounts.dtx v1.0d	\@newctr: \@nocnterr now has counter name argument	559
1994-03-16 ltfiles.dtx v1.0c			\addtocounter: \@nocnterr now has counter name argument	558
\document: (DPC) directly add file list settings	489		\setcounter: \@nocnterr now has counter name argument	557
1994-03-16 ltmiscen.dtx v1.0b			\stepcounter: Use \addtocounter to have name checked	559
\@verbatim: Remove			1994-04-09 ltthm.dtx v1.0b	
\global\@inlabeledfalse again. . .	882		\@othm: Use standard counter error message (FMi)	1009
1994-03-28 ltalloc.dtx v1.0d			1994-04-11 ltclass.dtx v0.3g	
General: Redefinition of ‘new’ allocations removed.	421		\endfilecontents: Add star form, don’t write \endinput at the end of the file.	1131
1994-03-28 ltdirchk.dtx v1.0d			\ProvidesFile: Protect against weird catcodes.	1110
General: Improve documentation . . .	1		1994-04-11 ltfssbas.dtx v2.1h	
1994-03-28 lterror.dtx v1.0d			General: Added \defaultscriptratio and \defaultscriptscriptratio. ASAJ.	571
\@invalidchar: (DPC) Comment out (use catcode15 instead)	437		\defaultscriptratio: Macro added	597
General: Remove test for			\defaultscriptscriptratio: Macro added	597
\inputlineno undefined.	433		1994-04-12 ltboxes.dtx v1.0c	
1994-03-28 ltfiles.dtx v1.0d			General: Remove \@acci, now defined in ltplain.dtx	938
\document: (DPC) Use \normalsize not \@normalsize	488		Remove \@dischyph, now defined in ltinit.dtx	938
(DPC) remove				
\@normalsize check	488			
1994-03-28 ltfloat.dtx v1.0b				
\@caption: Use \normalsize not \@normalsize	1029			
General: Split further from ltherest.dtx	1026			
1994-03-28 ltlists.dtx v1.0b				
General: Improve documentation . . .	907			
1994-03-28 ltmiscen.dtx v1.0c				
General: Improve Documentation . . .	864			
1994-03-28 ltplain.dtx v1.0c				
\newlanguage: Remove some \outer declarations.	17			
1994-03-28 ltsect.dtx v1.0b				
General: Split further from ltherest.dtx	1012			
1994-03-28 lttab.dtx v1.0b				
General: Improve documentation . . .	947			

1994-04-12 ltdefs.dtx v1.0g	1994-04-18 ltfsstrc.dtx v2.3d
\@dischyph: Define \@dischyph, was previously in ltboxes.dtx	General: Changed to new error/warning scheme
1994-04-12 ltplain.dtx v1.0d	\font@submax: Changed dimen to macro
General: Define \@acci	\fontsubfuzz: Changed dimen to macro
1994-04-12 ltvers.dtx v1.0b	\subst@size: \font@submax and \fontsubfuzz now macros
General: Have version info generated automatically.	1994-04-19 ltpage.dtx v1.0b
1994-04-14 ltfntcmd.dtx v3.2b	General: Improve documentation
General: Macros renamed to non-private forms, JB	1994-04-20 ltfntcmd.dtx v3.3a
\DeclareOldFontCommand: Renamed from \@newfontswitch	General: Documentation up-dated
1994-04-15 ltboxes.dtx v1.0d	New implementation of \nocorr
\@isavebox: Added missing percent character.	\check@nocorr@: Macros added
1994-04-17 ltcounts.dtx v1.0e	\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters
\@newctr: Use \@nocounterr instead of \@nocnterr	1994-04-20 ltmiscen.dtx v1.0e
\addtocounter: Use \@nocounterr instead of \@nocnterr	\@enddocument@kernel@warnings: Changed logic for producing warning messages
\setcounter: Use \@nocounterr instead of \@nocnterr	1994-04-21 ltboxes.dtx v1.0e
1994-04-17 lterror.dtx v1.0h	\@iiiminipage: Extra \bgroup for color
\@nocounterr: New name for error message, old error message (without arg) kept	\@mpfootnotetext: Extra \endgraf for color
1994-04-17 ltthm.dtx v1.0c	\endminipage: Extra \egroup for color
\@othm: Use new std counter error message (FMi)	1994-04-21 ltfinal.dtx v0.1c
\@thm: Use new std counter error message (FMi)	General: Added comments, set the catcodes of 128–255.
1994-04-18 ltfinal.dtx v0.1b	1994-04-22 ltssini.dtx v2.1g
General: Initialize \textheight, \textwidth and page style	\not@math@alphabet: Message changed again
1994-04-18 ltfloat.dtx v1.0d	1994-04-23 ltfinal.dtx v0.1d
\@footnotetext: (DPC) Remove Color support	General: Check that \font@submax is still zero
\@savemarbox: (DPC) Remove Color support	1994-04-24 ltoutput.dtx v1.0m
1994-04-18 ltssbas.dtx v2.1i	\@resetfhps: Number 2 changed to \tw@
General: Macro \no@alphabet@help removed again	Warning changed
\calculate@math@sizes: Changed message to log only	\@specialoutput: Message changed to give more info and ‘top’ removed
\no@alphabet@error: Use std LaTeX error macro	\@topnewpage: Message changed to give more info
1994-04-18 ltssdcl.dtx ???	Warning message removed as it will be generated later
\DeclareMathAlphabet: Pass correct arg (2 not 3)	General: Changed \@normalsize to \normalsize.
1994-04-18 ltssdcl.dtx v2.1d	Corrected unverbbed commands in documentation.
General: Removed surplus \no@alphabet@error (see fam.dtx)	Removed some long lines and other aesthetic changes.

Warning messages changed/corrected.	1214	1994-04-30 ltfontcmd.dtx v3.3b General: Documentation up-dated and tidied	790
1994-04-24 ltpictur.dtx v0.1b General: Removed surplus spaces after <code>\hbox to</code> in several cases	974	Prefix frag@ changed to frag in <code>\@protecteddef</code>	790
1994-04-25 ltclass.dtx v0.3h General: Removed spurious extra 's at the end of error messages	1098	Title changed	790
1994-04-25 ltfloat.dtx v1.0e <code>\@largefloatcheck</code> : Changed warning message to give more info	1035	Warning changed to info message in <code>\@protecteddef</code>	790
Command added	1035	1994-04-30 ltoutput.dtx v1.0n <code>\@activechar@info</code> : <code>\@activechar@warning</code> changed to <code>\@activechar@info</code>	1253
General: Changed warning messages	1026	<code>\@combinedblfloats</code> : Removed rule in topnewpage case	1261
Removed obsolete tracing code	1026	<code>\@emptycol</code> : Empty column action added: <code>\@emptycol</code>	1230
1994-04-27 ltfssstrc.dtx v2.3e General: Corrected item that was forgotten in last change.	671	<code>\@flsetnum</code> : Rogue space removed	1297
1994-04-28 lterror.dtx v1.0j <code>\@inmatherr</code> : Macro added	437	<code>\@specialoutput</code> : Cut-off point changed to <code>2\baselineskip</code>	1235
1994-04-28 lterror.dtx v1.1c <code>\@inmatherr</code> : Replaced <code>\noexpand</code> with <code>\protect</code>	437	Empty column action added: <code>\@emptycol</code>	1235
1994-04-28 ltssdcl.dtx v2.1e General: Removed all <code>\uppercase</code> in hex num parsing macros	702	Extra empty column added for twocolumn case	1235
1994-04-28 ltlists.dtx v1.0c <code>\item</code> : Replaced <code>\@ltxnomath</code> by <code>\@inmatherr</code>	919	Extra empty column added for twocolumn case (wrong, see below)	1235
1994-04-28 ltpictur.dtx v0.1c <code>\@multiput</code> : (DPC) Macro added	978	<code>\@topnewpage</code> : Added setting of <code>\col@number</code>	1230
General: bezier curves added	1002	Cut-off point changed to <code>3\baselineskip</code>	1231
<code>\multiput</code> : (DPC) Ignore spaces between)(.	978	Empty column action added: <code>\@emptycol</code>	1231
<code>\picture</code> : (DPC) Ignore spaces before (.	976	Message changed for Frank	1231
1994-04-28 ltplain.dtx v1.0g General: Turn off overfull box tracing in log	23	General: <code>\@activechar@warning</code> changed to an info message.	1214
1994-04-29 ltclass.dtx v1.0a General: Change version number to 1 (no other change)	1098	Added <code>\col@number</code>	1214
1994-04-29 ltmiscen.dtx v1.0f <code>\@verbatim</code> : <code>\leavevmode</code> added	882	Documentation tidied.	1214
Change to <code>\everypar</code> added	882	Empty column action added.	1214
1994-04-29 ltoutenc.dtx 1.4a General: Removed <code>\EncodingSpecific</code> . Renamed all the commands. Added <code>\DeclareTextGlyph</code> and <code>\UndeclareTextCommand</code>	513	Fixed bug from <code>\dblfigrule</code> with <code>\@topnewpage</code>	1214
Removed Rokicki's OT1 variant encoding. Moved the driver to the top.	512	Full of floats action improved.	1214
		<code>\col@number</code> : Added <code>\col@number</code>	1226
		<code>\onecolumn</code> : Added setting of <code>\col@number</code>	1228
		1994-05-01 lterror.dtx v1.0k <code>\@latexerr</code> : (CAR) Added draft <code>\@latexinfo</code>	433
		1994-05-01 ltoutenc.dtx 1.4a General: Added the <code>\a</code> command.	522
		Added the <code>\SaveAtCatcode</code> and <code>\RestoreAtCatcode</code> commands.	525
		Removed the uc/lc table settings, since the T1 uc/lc table is now the default.	533

Rewrote for the new syntax.	525, 528	<code>\end@float</code> : (CAR) Added	
1994-05-01 ltoutenc.dtx v1.4a		<code>\@largefloatcheck</code>	1033
General: Removed Rokicki's		1994-05-03 ltfsdcl.dtx v2.1f	
encoding.	509	General: Renamed	
Renamed the commands, removed		<code>\@DeclareMathDelimiter</code> to	
the <code>\EncodingSpecific</code> command.		<code>\@DeclareMathDelimiter</code>	702
Turned all slots into decimal.		1994-05-03 ltlists.dtx v1.0d	
Added <code>\a</code>	509	<code>\@item</code> : <code>\hskip</code> changed to <code>\kern</code> . .	920
1994-05-02 ltcntrl.dtx v1.0l		<code>\item</code> : Removed superfluous braces	919
<code>\@break@tfor</code> : Macro added (from		1994-05-03 ltmiscen.dtx v1.0h	
ltfiles.dtx)	426	<code>\@centercr</code> : <code>\@badcrerr</code> replaced by	
1994-05-02 ltdefs.dtx v1.1f		<code>\@nolnerr</code>	878
<code>\renewcommand</code> : Removed surplus		1994-05-03 lttab.dtx v1.0d	
<code>\space</code> in error	87	<code>\@endpbox</code> : Use <code>\@finalstrut</code> based	
<code>\renewenvironment</code> : Removed surplus		on depth of <code>\@arstrutbox</code>	972
<code>\space</code> in error	88	1994-05-04 ltclass.dtx v1.0b	
1994-05-02 ltfiles.dtx v1.0f		<code>\NeedsTeXFormat</code> : Changed wording of	
<code>\@iffilenamepath</code> : <code>\@break@loop</code>		the warning	1119
renamed to <code>\@break@tfor</code>	502	1994-05-04 lterror.dtx v1.0m	
<code>\@obsoletefile</code> : Make		<code>\@badcrerr</code> : Error message removed	436
<code>\@onlypreamble</code>	506	1994-05-05 ltbibl.dtx v1.0c	
1994-05-02 ltfinal.dtx v0.1e		<code>\@citex</code> : Set switch for warning and	
General: Added setting the 'letter'		end of run.	1055
catcodes.	1346	<code>\nocite</code> : Do not write page number in	
Added setting the 'other'		<code>\nocite</code> warning message.	1056
catcodes.	1345	Set switch for warning and end of	
Added setting the special		run.	1056
catcodes.	1345	1994-05-05 ltfinal.dtx v0.1g	
Made slot 127 illegal	1346	General: Added empty errhelp. . . .	1332
Set all the catcodes	1332	<code>\errhelp</code> : Set error help empty. . .	1351
1994-05-02 ltfinal.dtx v0.1f		1994-05-05 ltfntcmd.dtx v3.3c	
General: Set the catcode of		<code>\@math@egroup</code> : Corrected	
control-J.	1345	<code>\@fontswitch</code> and added saved	
1994-05-02 ltmiscen.dtx v1.0g		versions	797
General: Changed 91 to 1991 and		General: Corrected <code>\@fontswitch</code> . .	790
moved some bits	864	1994-05-05 ltmiscen.dtx v1.0i	
1994-05-02 ltoutput.dtx v1.0o		General: Removed braces from	
<code>\@resethfps</code> : Code shortened	1296	<code>\ifnextchar</code> and <code>\ifstar</code> arguments .	864
General: Code of <code>\@resethfps</code>		1994-05-07 lttab.dtx v1.0c	
shortened.	1214	<code>\@maxtab</code> : Changed <code>\@firsttab</code> to	
1994-05-03 ltbibl.dtx v1.0b		<code>\chardef</code>	951
<code>\nocite</code> : Make <code>\nocite</code> issue a		Changed <code>\@maxtab</code> to <code>\chardef</code> . .	951
warning for an undefined citation		General: Removed definition of <code>\+</code> .	947
key.	1056	Removed surplus braces from	
1994-05-03 ltfinal.dtx v0.1f		<code>\@ifnextchar</code> constructs	947
General: Set the catcode of control-J		1994-05-08 ltfntcmd.dtx v3.3d	
to be 'other', for use in messages.	1332	General: Removed	
1994-05-03 ltfloat.dtx v1.0f		<code>\@undefinedfonterror</code>	790
General: (CAR) Added		<code>\normalsize</code> : Removed	
<code>\@largefloatcheck</code>	1026	<code>\@undefinedfonterror</code>	798
Removed unnecessary braces from		1994-05-09 ltfntcmd.dtx v3.3f	
arguments of <code>\@ifnextchar</code> . . .	1026	General: Replaced all <code>\next</code> by	
<code>\end@dblfloat</code> : <code>\@largefloatcheck</code>		<code>\@let@token</code> and undo change	
added	1034	3.3e, whatever that was.	790

1994-05-10 ltdefs.dtx v1.0n			
General: (ASAJ) Added		<code>\@changed@cmd</code> and	
<code>\DeclareProtectedCommand</code>	80	<code>\DeclareProtectedCommand</code>	513
Added <code>\DeclareProtectedCommand</code>	91	Renamed the commands again.	
Removed braces around		Made the encoding part of the	
<code>\@ifundefined</code> argument. ASAJ.	87	command syntax. Added the	
<code>\makeatother</code> : Added <code>\makeatletter</code>		<code>\DeclareTextCommand</code> interface.	
and <code>\makeatother</code> ASAJ.	112	Used	
1994-05-10 lterror.dtx v1.0n		<code>\DeclareProtectedCommand</code>	509
<code>\@latexerr</code> : (ASAJ) Added extra		<code>\DeclareTextAccent</code> : Reimplemented	
blank lines to <code>\@latexerr</code>	433	using <code>\DeclareTextCommand</code>	516
1994-05-10 ltmiscen.dtx v1.0j		1994-05-11 ltspc.dtx v1.0o	
<code>\@sverb</code> : Slight change in error		<code>\hspace</code> : Use	
message text.	885	<code>\DeclareRobustCommand</code> . ASAJ.	481
1994-05-11 ltboxes.dtx v1.0f		1994-05-12 ltboxes.dtx v1.0g	
<code>\@begin@tempboxa</code> : Use new		<code>\@finalstrut</code> : macro added	945
<code>\color@setgroup</code> concept.	927	<code>\fbox</code> : New definition, merged with	
<code>\@iiminipage</code> : Use new		<code>\framebox</code>	933
<code>\color@setgroup</code> concept.	940	<code>\framebox</code> : Merged <code>\fbox</code> and	
<code>\@mpfootnotetext</code> : Use new		<code>\framebox</code>	933
<code>\color@setgroup</code> concept.	941	<code>\normalcolor</code> : macro added for color	
Use new <code>\normalcolor</code> and		support	929
<code>\@finalstrut</code>	941	1994-05-12 ltdefs.dtx v1.0p	
General: Superfluous braces removed		General: (ASAJ) Fixed a bug with	
from several commands	926	<code>\relax</code> which was using <code>\@gobble</code>	
<code>\color@setgroup</code> : macro added for		before defining it.	80
color support	929	Fixed a bug with <code>\relax</code> which	
<code>\endminipage</code> : Use new		was using <code>\@gobble</code> before defining	
<code>\color@setgroup</code> concept.	940	it.	91
1994-05-11 ltclass.dtx v1.0c		1994-05-12 ltfssbas.dtx v2.1j	
<code>\endfilecontents</code> : Add checks for		General: New <code>baselinestretch</code> concept	571
form feed and tab	1131	Replaced hand-protected commands	
1994-05-11 ltdirchk.dtx v1.0e		by <code>\DeclareRobustCommand</code> defs	571
General: Add <code>\ProvidesFile</code> as used		<code>\f@linespread</code> : New macro	583
in fd files.	4	<code>\fontencoding</code> : Use	
1994-05-11 lterror.dtx v1.0o		<code>\DeclareRobustCommand</code>	581
<code>\@latexerr</code> : (ASAJ) Removed one of		<code>\fontfamily</code> : Use	
the extra blank lines to		<code>\DeclareRobustCommand</code>	582
<code>\@latexerr</code>	433	<code>\fontseries</code> : Use	
1994-05-11 ltlogos.dtx v1.0o		<code>\DeclareRobustCommand</code>	582
<code>\LaTeX</code> : Use		<code>\fontshape</code> : Use	
<code>\DeclareProtectedCommand</code> .		<code>\DeclareRobustCommand</code>	582
ASAJ.	484	<code>\fontsize</code> : Redefined to use	
<code>\LaTeXe</code> : Use		<code>\set@fontsize</code>	583
<code>\DeclareProtectedCommand</code> .		<code>\linespread</code> : New macro	583
ASAJ.	484	<code>\mathversion</code> : Use	
1994-05-11 ltoutenc.dtx 1.5a		<code>\DeclareRobustCommand</code>	584
General: Made T1 and OT1 generate		1994-05-12 ltfssdcl.dtx v2.1g	
packages rather than def files.		General: Allow <code>\relax</code> as undefined	
Renamed the ‘package’ module to		command	702
‘teststy’.	512	Allow <code>\relax</code> ’ed cmds to be	
1994-05-11 ltoutenc.dtx v1.5a		declared	702
General: Reimplemented		1994-05-12 ltfssini.dtx v2.1i	
<code>\DeclareTextCommand</code> using		General: Moved <code>\fontencoding</code> to	
		fam.dtx	735

Moved <code>\fontfamily</code> to fam.dtx .	735	1994-05-13 ltfiles.dtx v1.0g	
Moved <code>\fontseries</code> to fam.dtx .	735	<code>\document</code> : Added execution of	
Moved <code>\fontshape</code> to fam.dtx .	735	<code>\every@size</code>	488
Moved <code>\fontsize</code> to fam.dtx .	735	1994-05-13 ltfinal.dtx v0.1h	
Moved <code>\mathversion</code> to fam.dtx .	735	General: Added package <code>otlenc</code> , and	
Moved <code>\selectfont</code> to tracefnt.dtx	735	defined <code>\@acci</code> , <code>\@accii</code> and	
1994-05-12 ltfsstrc.dtx v2.3f		<code>\@acciii</code>	1332
<code>\selectfont</code> : Use		1994-05-13 ltfinal.dtx v1.0h	
<code>\DeclareRobustCommand</code>	675	General: Added output enc stuff .	1350
1994-05-12 ltoutenc.dtx 1.5a		1994-05-13 ltfloat.dtx v1.0g	
General: Removed the		<code>\@footnotetext</code> : (DPC) Add new	
<code>\SaveAtCatcode</code> and		style color support:	
<code>\RestoreAtCatcode</code> commands. .	525	<code>\normalcolor</code>	1047
Rewrote for the new syntax. .	525, 528	(DPC) Use <code>\@finalstrut</code>	1047
1994-05-12 ltoutput.dtx v1.0p		<code>\xfloat</code> : (DPC) Use <code>\normalcolor</code>	1031
<code>\writetsetup</code> : <code>\normalcoloradded</code>	1254	1994-05-13 ltfontcmd.dtx v3.3g	
General: <code>\normalcoloradded</code> in		General: Replaced <code>\@protecteddef</code> by	
various places (DPC).	1214	<code>\DeclareRobustCommand</code>	790
1994-05-13 ltboxes.dtx v1.0h		1994-05-13 ltfsbas.dtx v2.1k	
<code>\@arrayparboxrestore</code> : New accent		General: Remove File identification	
system, use <code>\let</code> not <code>\def</code>	938	‘typeout’	571
1994-05-13 ltcounts.dtx v1.0f		1994-05-13 ltfsbas.dtx v2.1l	
General: Removed <code>\@lalph</code>	566	<code>\DeclareFontEncoding</code> : Init encoding	
Removed <code>\@ialph</code>	566	change command	575
1994-05-13 ltdefns.dtx v1.0q		<code>\define@newfont</code> : Use <code>\@input@</code> for fd	
General: (ASAJ) Renamed		files	588
<code>\DeclareProtectedCommand</code> to		1994-05-13 ltssdcl.dtx v2.1h	
<code>\DeclareRobustCommand</code> .		General: Removed file identification	
Removed <code>\@if@short@command</code> . .	80	typeout	702
(ASAJ) Replaces <code>\space</code> by ‘ ’ in		1994-05-13 ltssini.dtx v2.1j	
<code>\csname</code>	80	General: Removed file identification	
Renamed		typeout	735
<code>\DeclareProtectedCommand</code> to		1994-05-13 ltfsstrc.dtx v2.3g	
<code>\DeclareRobustCommand</code> .		General: Removed typeouts as	
Removed <code>\@if@short@command</code> .		<code>\ProvidesPackage</code> writes to log. .	671
Moved to after the definition of		1994-05-13 ltoutenc.dtx v1.5b	
<code>\gobble</code>	91	General: Added <code>\{</code> , <code>\}</code> and <code>\\$</code>	509
1994-05-13 ltdefns.dtx v1.0r		Renamed	
General: (ASAJ) Added logging		<code>\DeclareProtectedCommand</code> to	
message to		<code>\DeclareRobustCommand</code>	509
<code>\DeclareProtectedCommand</code>	80	Replaces <code>\space</code> by ‘ ’ in <code>\csname</code> . .	509
Added logging message to		1994-05-13 ltpictur.dtx v0.1d	
<code>\DeclareProtectedCommand</code>	91	General: Removed surplus braces from	
1994-05-13 ltdefns.dtx v1.0s		<code>\@if..</code> constructions	974
General: (ASAJ) Added		1994-05-13 lttab.dtx v1.0d	
<code>\@backslashchar</code>	80	<code>\@contfield</code> : Color support	953
(ASAJ) Coded <code>\@ifdefinable</code> more		<code>\@startfield</code> : Color support	953
efficiently.	80	<code>\@stopfield</code> : Color support	953
Coded more efficiently, thanks to		<code>\a</code> : moved to ltoutenc	951
FMi.	87	1994-05-14 fontdef.dtx v2.1f	
1994-05-13 ltfiles.dtx LaTeX2e		General: Removed .def files.	768
<code>\listfiles</code> : Stop <code>\listfiles</code> being		1994-05-14 ltfsbas.dtx v2.1m	
run twice	507	<code>\enc@update</code> : Macro added	582

1994-05-14 ltfsbas.dtx v2.1n	1994-05-16 ltfsbas.dtx v2.1p
General: Set defaults for all <code>\f@...</code> 583	<code>\fontsize</code> : Pass <code>\baselinestretch</code> not
<code>\DeclareErrorFont</code> : Don't set	<code>\f@linespread</code> 583
<code>\f@encoding</code> 591	<code>\linespread</code> : Remove surplus braces 583
<code>\DeclareFontEncoding</code> : Log if	1994-05-16 ltfsini.dtx v2.1m
encoding is redeclared 575	<code>\@acciii</code> : Define saved versions of
Only init enc change cmd when new	accents 765
encoding 575	1994-05-16 ltlogos.dtx v1.1a
1994-05-14 ltfsini.dtx v2.1k	General: (ASAJ) Split from ltinit.dtx. 484
General: Init error font just before	1994-05-16 ltmath.dtx v1.0k
checking for fontdef.cfg 764	<code>\ensuremath</code> : Use
<code>\reset@font</code> : Remove surplus braces 761	<code>\DeclareRobustCommand</code> and add
1994-05-14 ltfsstrc.dtx v2.3h	extra braces in math mode 903
<code>\selectfont</code> : Added <code>\enc@update</code> . . 677	1994-05-16 ltoutenc.dtx 1.5h
1994-05-14 ltoutenc.dtx 1.5d	General: <code>\pounds</code> was still using u
General: Moved the driver to the top. 512	rather than ui shape. 525
1994-05-14 ltoutenc.dtx v1.5c	1994-05-16 ltoutenc.dtx v1.5f
General: Added the fontenc package 553	General: enc files now have uc
Added the fontenc package. 509	encoding name parts (FMi) 509
Fixed a bug which caused an	Revert code so that the encoding
infinite loop if <code>\f@encoding</code> was	given is used in
incorrectly set. 509, 513	<code>\DeclareTextCommand</code> (FMi) . . . 509
Moved fontsmpl to its own dtx file. 509	1994-05-16 ltoutenc.dtx v1.5g
1994-05-14 ltoutenc.dtx v1.5d	General: Made fontenc.sty use the new
General: Rewrote	mixed-case encoding files. 509
<code>\DeclareTextCommand</code> to define its	Removed the lowercasing of the
argument to use the current	filename. 553
encoding by default, rather than	1994-05-16 ltoutenc.dtx v1.5h
the encoding provided to	General: Added <code>\NG</code> , <code>\ng</code> , <code>\TH</code> , <code>\th</code> ,
<code>\DeclareTextCommand</code> 509, 513	<code>\DH</code> , <code>\dh</code> , <code>\DJ</code> and <code>\dj</code> 509
Tidied up the documentation. . . . 509	Added <code>\r</code> (ring accent) and <code>\k</code>
1994-05-14 ltoutenc.dtx v1.5e	(ogonek) accents. 509
General: Replaced <code>\ENC@cmd</code> by	Fixed a bug with <code>\pounds</code> 509
<code>\ENC-cmd</code> 509	Removed <code>\P</code> from the OT1
1994-05-15 ltfsbas.dtx v2.1o	definitions file. 509
General: encoding cmds changed to	1994-05-16 ltoutenc.dtx v1.5i
enc-cmd 571	General: Fixed a bug with <code>\d</code> 509
1994-05-16 fontdef.dtx v2.1g	1994-05-16 ltoutput.dtx v1.0q
General: Removed	<code>\@writsetup</code> : Changed setting of
<code>\DeclareFontEncoding</code> for ot1 and	accents (FMi): with the new
t1 and input .def files instead . . 768	encoding setup they can use <code>\let</code> .
1994-05-16 ltalloc.dtx v1.1a	It could also use the new internal
General: (ASAJ) Split from ltinit.dtx. 421	commands? 1255
1994-05-16 ltctrl.dtx v1.0a	General: Changed setting of accents
General: (ASAJ) Split from ltinit.dtx. 423	(FMi). 1214
1994-05-16 ltdefns.dtx v1.1a	1994-05-16 ltpar.dtx v1.1a
General: (ASAJ) Split from ltinit.dtx. 80	General: (ASAJ) Split from ltinit.dtx. 438
1994-05-16 lterror.dtx v1.1a	1994-05-16 ltplain.dtx v1.0h
General: (ASAJ) Completely new	General: Comment out encoding
error interface. 427	specific commands 29
(ASAJ) Split from ltinit.dtx. . . . 427	Remove <code>\@acci</code> and friends again . 30
1994-05-16 ltfinal.dtx v1.0i	Remove unnecessary def for <code>\item</code> 29
General: moved output enc stuff to	<code>\loop</code> : Use Kabelschacht method . . . 27
lfonts 1350	<code>\m@th</code> : Remove unnecessary space . . . 28

1994-05-16 ltspace.dtx v1.1a	Replaced <code>\defaultencoding</code> with	
General: (ASAJ) Split from ltinit.dtx. 463	<code>\encodingdefault</code>	509
1994-05-17 ltclass.dtx v1.0e	1994-05-19 ltbibl.dtx v1.1a	
<code>\@use@option</code> : Execute option after	General: Initial version of ltbibl.dtx,	
removing from list, not before . 1115	split from ltidxbib.dtx	1053
1994-05-17 ltdefns.dtx 1.1b	1994-05-19 ltcounts.dtx v1.1a	
General: (ASAJ) Added the	General: Extracted file from ltcntlen. 556	
<code>\@protect@...</code> commands.	1994-05-19 ltdefns.dtx v1.1d	
92	General: (RmS) Added definitions for	
1994-05-17 ltdefns.dtx v1.1b	<code>\@namedef</code> and <code>\@nameuse</code> again. . 80	
General: (ASAJ) Added definitions for	1994-05-19 ltfinal.dtx v0.1k	
<code>protect</code>	General: Removed <code>\makeat...</code>	1332
(ASAJ) Removed warnings and	1994-05-19 ltidxglo.dtx v1.1a	
logging to lterror.dtx.	General: Initial version of ltidxglo.dtx,	
80	split from ltidxbib.dtx	1050
Added the discussion of protected	1994-05-19 ltlength.dtx v1.1a	
commands, defined the values that	General: Extract file ltlength from	
<code>\protect</code> should have.	ltcntlen.	569
91	1994-05-19 ltpageno.dtx v1.1a	
1994-05-17 ltdefns.dtx v1.1c	General: Extract file ltpageno from	
General: (ASAJ) Redid definitions for	ltcntlen.	839
<code>protect</code>	1994-05-19 ltplain.dtx v0.1k ltfinal	
80	<code>\showoutput</code> : used <code>\maxdimen</code> not	
1994-05-17 lterror.dtx v1.1b	99999	31
General: (ASAJ) Moved error stuff	<code>\showoverfull</code> : used <code>\@ne</code> not 1	30
from ltdefns.dtx.	1994-05-19 ltxref.dtx v1.1a	
427	General: Extract file ltxref from	
1994-05-17 ltfssini.dtx v2.1n	ltcntlen.	840
<code>\copyright</code> : Really add extra braces	1994-05-20 ltdefns.dtx v1.1e	
761	General: Changed command name	
<code>\nfss@text</code> : Added braces to allow	from <code>\@checkcommand</code> to	
use in subscripts	<code>\CheckCommand</code>	80
761	<code>\CheckCommand</code> : Changed name from	
1994-05-17 ltmath.dtx v1.0i	<code>\@checkcommand</code> to	
General: Replaced <code>\let</code> by <code>\gdef</code> , for	<code>\CheckCommand</code>	89
indirect definition.	1994-05-20 lterror.dtx v1.1c	
896	General: (ASAJ) Added	
1994-05-17 ltoutenc.dtx v1.5j	<code>\@latex@info@no@line</code>	427
General: Added braces to <code>\pounds</code> so	(ASAJ) Added missing full stops. 427	
it works as a subscript.	(ASAJ) Fixed a bug with	
509	<code>\@inmatherr</code>	427
1994-05-18 ltdefns.dtx 1.1c	1994-05-20 ltfinal.dtx v0.1l	
General: (ASAJ) Renamed the	General: Use new font warning	
commands, and removed one	commands	1339
which is no longer needed.	1994-05-20 ltfloat.dtx v1.0h	
92	<code>\@endfloatbox</code> : Restore outer value of	
1994-05-18 ltdefns.dtx v1.1c	<code>@nobreak</code> switch.	1035
General: Redid the discussion and	<code>\outer@nobreak</code> : Macro added:	
definitions, in line with the	default is to do nothing.	1035
proposed new setting of <code>\protect</code>	1994-05-20 ltfntcmd.dtx v3.3h	
in the output routine.	General: Use new error commands . 790	
91	1994-05-20 ltfssbas.dtx v2.1q	
1994-05-18 ltfinal.dtx v0.1j	General: Use new error commands . 571	
General: Corrected the lccode for		
d-bar.		
1332		
1994-05-18 ltlogos.dtx v1.1b		
General: (ASAJ) Added the $\TeX$		
logo.		
484		
(ASAJ) Made the $\text{\LaTeX}$ logo use		
the text font ‘2’ rather than the		
math font ‘2’.		
484		
1994-05-18 ltoutenc.dtx v1.5k		
General: Made dotted-i produce ‘i’. . 509		
Removed braces from <code>\pounds</code> and		
<code>\dollar</code>		
509		

1994-05-20 ltfssstrc.dtx v2.3i	1994-05-22 lterror.dtx v1.2a
General: Use new error command	General: (ASAJ) Made
names 671	\GenericError, \GenericWarning
1994-05-20 ltmiscen.dtx v1.0l	and \GenericInfo robust. 427
\@writefile: Added correct setting of	(ASAJ) Replaced \ and tilde by
\protect. 871	\MessageBreak and \space. 427
1994-05-20 ltmiscen.dtx v1.0m	(ASAJ) Replaced
General: Use new warning commands 864	\@generic@message and
1994-05-20 ltoutput.dtx v1.0s	\@generic@error by
\@writsetup: Added setting of	\GenericError, \GenericWarning
\protect during \shipout. 1254	and \GenericInfo. 427
General: Added setting of \protect	(ASAJ) Replaces \string by
during \shipout. 1214	\protect in some messages. 427
1994-05-20 ltpage.dtx v1.0d	1994-05-22 lterror.dtx v1.2d
\markright: Changed setting for	\GenericError: (DPC) Alternative
\protect. 1090	version added for old TeXs 428
1994-05-20 ltsect.dtx v1.0c	(DPC) New version using long
\addcontentsline: Correct setting of	command name. 428
\protect. 1021	1994-05-22 ltfloat.dtx v1.0i
\addtocontents: Correct setting of	General: Use new warning
\protect. 1022	commands 1026
1994-05-21 ltbibl.dtx v1.1b	1994-05-22 ltoutput.dtx v1.0t
General: Use new warning	General: Changed warnings and infos
commands 1053	to new commands. 1214
1994-05-21 lterror.dtx v1.1d	1994-05-22 ltpictur.dtx v0.1e
General: (ASAJ) Made the error	General: Use new warning cmds 974
commands robust. 427	1994-05-23 ltclass.dtx v1.0h
1994-05-21 ltfiles.dtx v1.0h	\NeedsTeXFormat: Don't stop
General: Use new error commands 485	completely when format is wrong 1119
1994-05-21 ltlists.dtx v1.0f	\usepackage: Remove argument if
General: Use new error commands 907	possible 1118
1994-05-21 ltmiscen.dtx v1.0n	1994-05-23 ltdirchk.dtx v1.0f
General: Use new error commands 864	General: Document \@TeXversion 1
1994-05-21 ltsect.dtx v1.0d	1994-05-23 ltfssstrc.dtx v2.3j
General: Use new error commands 1012	General: Removed def of
1994-05-21 lttab.dtx v1.0f	\@warn@break 690
General: Use new error commands 947	1994-05-23 ltoutput.dtx v1.0u
1994-05-21 ltxref.dtx v1.1b	\@activechar@info: Added
General: Use new warning commands 840	\MessageBreak 1253
\newlabel: Use new warning	\@writsetup: Changed resetting of
commands 843	\protect after shipout to use
1994-05-22 ltclass.dtx v1.0f	\aftergroup 1254
General: Use new warning and error	General: Added \MessageBreak. 1214
commands 1094	Changed resetting of \protect after
1994-05-22 ltdefs.dtx v1.1f	shipout. 1214
General: Use new warning and error	1994-05-24 lterror.dtx v1.2e
cmds 80	\@latex@info@no@line: Macro added 431
1994-05-22 lterror.dtx v1.1e	1994-05-24 lterror.dtx v1.2f
General: (ASAJ) Replaced bgroup by	General: (DPC) wrap long lines 427
begingroup in error messages, to	1994-05-24 ltfmtcmd.dtx v3.3i
stop extra mathords creeping into	General: Tidying and typos fixed 790
math mode. 427	1994-05-24 ltmiscen.dtx v1.0q
	\@currrentvline: Use \@empty as outer
	default 877

1994-05-25 ltdirchk.dtx v1.0g		<code>\filename@parse</code> : Mac parser had "	
	typo for :	12	
1994-05-25 ltfntcmd.dtx v3.3j		General: Insertion of <code>\aftergroups</code> to	
	implement <code>\nocorr</code> moved to the		
	end of the group	790	
	<code>\check@icr</code> : Macros added	794	
	<code>\check@nocorr@</code> : Insertion of		
	<code>\aftergroups</code> moved and defaults		
	set up for efficiency	794	
	<code>\DeclareTextFontCommand</code> :		
	<code>\expandafter</code> inserted	792	
	Insertion of <code>\aftergroups</code> moved	792	
1994-05-25 ltoutput.dtx v1.0v		General: Extra documentation.	1214
1994-05-25 ltsect.dtx v1.0e		<code>\@dottedtocline</code> : Put braces around	
	argument 4 (the actual toc entry)		
	to avoid font (and possibly other)		
	changes leaking out to the		
	leaders.	1024	
1994-05-25 lthtm.dtx v1.0c		General: Modify documentation	1007
1994-05-25 ltvers.dtx v1.0d		General: Remove PRELIMINARY	
	TEST RELEASE from startup		
	banner (spring is here)	35	
1994-05-25 ltxref.dtx v1.1c		General: Modify documentation	840
1994-05-26 ltfiles.dtx LaTeX2e		<code>\@missingfileerror</code> : Modify message	
	format	505	
1994-05-26 ltlogos.dtx v1.1c		General: Remove <code>\SLiTeX</code> logo	484
1994-05-26 ltmiscen.dtx v1.0r		General: <code>\literal</code> removed	888
1994-05-26 ltplain.dtx v1.1m		<code>\iterate</code> : (CAR) added <code>\long</code>	27
	<code>\underbar</code> : (CAR/FMi) changed to		
	use box <code>\tw@</code>	29	
1994-05-26 ltplain.dtx v1.1p		<code>\underbar</code> : (DPC) changed to use	
	<code>\sbox</code>	29	
1994-05-29 ltfssdcl.dtx v2.1j		General: Use new error commands	702
1994-05-31 ltfinal.dtx v1.0n		General: Renamed <code>lthyphen.*</code> to	
	<code>lthyphen.*</code>	1332	
1994-06-01 ltboxes.dtx v1.0i		<code>\@frameb@x</code> : Macro added.	934
	<code>\@ifframebox</code> : New version, so <code>\width</code>		
	is correct in <code>\framebox</code>	933	
	<code>\fbox</code> : New version, using		
	<code>\@frameb@x</code>	933	
	<code>\framebox</code> : New version, so <code>\width</code> is		
	correct in <code>\framebox</code>	933	
1994-06-01 ltlogos.dtx v1.1d		<code>\LaTeX</code> : Add <code>\m@th</code> to force math size	
	calculations	484	
1994-06-01 ltoutput.dtx v1.0w		General: Tidied up typesetting.	1214
1994-06-08 ltfinal.dtx v1.0m		General: Add patch file system	1350
1994-06-09 ltfinal.dtx v1.0n		General: For $\TeX$ 2, do not set codes	
	for higher half of character		
	table.	1337, 1346	
1994-06-09 ltfntcmd.dtx v3.3k		General: Tidying and typos fixed in	
	documentation	790	
1994-06-18 ltfntcmd.dtx v3.3l		General: Added check for empty text	790
	<code>\check@nocorr@</code> : Added check for		
	empty text	794	
1994-06-22 ltfntcmd.dtx v3.3m		General: Removed space from	
	<code>\nfss@text</code>	790	
	Renamed <code>\check@nocorr</code>	790	
	<code>\check@nocorr@</code> : Renamed		
	<code>\check@nocorr</code> to <code>\text@command</code>		
	to improve <code>\long</code> error message	794	
	<code>\DeclareTextFontCommand</code> : Removed		
	space from <code>\nfss@text</code>	792	
1994-06-22 ltmath.dtx v1.2t classes		<code>\mathindent</code> : Set <code>\mathindent</code> at the	
	end of the class instead of at begin		
	document	904	
1994-07-20 ltlogos.dtx v1.1e		<code>\LaTeX</code> : Save a few tokens	484
	<code>\LaTeXe</code> : Save a few tokens	484	
1994-07-20 ltpage.dtx v1.0h		<code>\sloppy</code> : Save a few tokens	1093
1994-09-16 ltfssbas.dtx v2.1s		<code>\nfss@catcodes</code> : Reset [and] as well,	
	just in case	589	
1994-10-07 ltoutenc.dtx v1.5l		General: Moved the ogonek accent.	509
1994-10-11 ltdirchk.dtx v1.0h		<code>\@TeXversion</code> : Check for TeX3.14	13
	General: Modify all of <code>ltxcheck</code> again	13	
1994-10-12 ltsect.dtx v1.0f		General: Doc. typos	1012
1994-10-14 fontdef.dtx v2.2a		General: New coding	766
1994-10-14 ltfssini.dtx v2.2a		General: New coding for <code>cfg</code> files	735

- 1994-10-14 ltmiscen.dtx v1.0s
 General: Move math to other file . . . 864
- 1994-10-14 ltplain.dtx v1.1a
 General: Moved code to other files. . . 14
- 1994-10-15 ltfsbas.dtx v2.1t
 $\backslash extract@alph@from@version$: Warn if math alpha is used outside math 597
- 1994-10-18 ltboxes.dtx v1.0j
 $\backslash @frameb@x$: $\backslash leavevmode$ added . . . 934
 $\backslash @ifframebox$: $\backslash leavevmode$ moved to $\backslash @frameb@x$ 933
 $\backslash @parboxto$: Macro added to remove misuse of $\backslash @empty$ 936
 General: stuff from ltpatch done . . . 926
 $\backslash fbox$: $\backslash long$ added 933
 $\backslash mbox$: $\backslash long$ added 927
 $\backslash sbox$: $\backslash long$ added 931
- 1994-10-18 ltclass.dtx v1.0j
 General: Move $\backslash listfiles$ to ltfiles.dtx 1094
- 1994-10-18 ltdefs.dtx v1.2a
 $\backslash @star@or@long$: macro added 83
 General: Add extra test for $\backslash endgraf$. . . 80
 Add star-forms for all commands . . . 80
 $\backslash renew@environment$: reset end command 88
- 1994-10-18 ltfiles.dtx v1.0i
 $\backslash listfiles$: code moved here from ltclass 507
- 1994-10-18 ltoutenc.dtx v1.5l
 General: Added new definitions of $\backslash patterns$ and $\backslash hyphenation$. . . 521
- 1994-10-18 ltoutenc.dtx v1.5m
 General: Added new definitions of $\backslash patterns$ and $\backslash hyphenation$. . . 509
- 1994-10-18 ltsect.dtx v1.0g
 $\backslash @dottedtocline$: Added $\backslash normalcolor$ for page number . . . 1024
 General: Added $\backslash normalcolor$. . . 1012
- 1994-10-19 ltfsbas.dtx v2.1t
 $\backslash DeclareFontEncoding$: Add missing $\backslash relax$ 575
- 1994-10-23 ltfsstrc.dtx v23.k
 $\backslash every@math@size$: Renamed to $\backslash every@math@size$ 680
- 1994-10-23 ltmath.dtx v1.0l
 $\backslash @eqnnum$: Added $\backslash normalcolor$ since $\backslash eqno$ introduces a subgroup of the displayed math group 900
 $\backslash ensuremath$: Remove extra braces: but see p 168 of Leslie's book . . . 903
- 1994-10-24 ltboxes.dtx v1.0k
 $\backslash fbox$: Inner braces added (to fix latex/1061) 933
- 1994-10-25 fontdef.dtx v2.2c
 General: Added OMSenc.def 768
- 1994-10-25 ltboxes.dtx v1.0l
 $\backslash @isavepicbox$: missing percent (moved from ltpatch) 931
- 1994-10-25 ltdefs.dtx v1.2b
 General: Documentation improvements 80
- 1994-10-25 ltoutenc.dtx 1.6a
 General: Added $\backslash textdollar$, $\backslash textlbrace$, $\backslash textrbrace$, $\backslash textsterling$, $\backslash textunderline$. . . 528
 Removed $\backslash textlbrace$, $\backslash textrbrace$, $\backslash textunderline$ to give them their proper names. . . 528
- 1994-10-25 ltoutenc.dtx v1.6a
 General: Added $\backslash ProvideTextCommand$, $\backslash UseTextSymbol$, $\backslash UseTextAccent$, $\backslash DeclareTextSymbolDefault$, $\backslash DeclareTextAccentDefault$, $\backslash DeclareTextCommandDefault$, and $\backslash ProvideTextCommandDefault$. . . 509
 Added the $\backslash Provide$ commands, and the default definitions. 513
 Added the defaults. 522
 Added the files OTlenc.def, Tlenc.def and OMSenc.def. . . . 522
 Added the OMS encoding. 534
- 1994-10-27 ltoutenc.dtx 1.6b
 General: Added $\backslash textasciicircum$, $\backslash textasciitilde$, $\backslash textbackslash$, $\backslash textbar$, $\backslash textbraceleft$, $\backslash textbraceright$, $\backslash textcompwordmark$, $\backslash textemdash$, $\backslash textendash$, $\backslash textexclamdown$, $\backslash textgreater$, $\backslash texthyphenchar$, $\backslash textthyphen$, $\backslash textless$, $\backslash textquestiondown$, $\backslash textquotedblleft$, $\backslash textquotedblright$, $\backslash textquotedbl$, $\backslash textquoteleft$, $\backslash textquoteright$, $\backslash textunderscore$, $\backslash textvisiblespace$ 528
 Added: $\backslash textemdash$, $\backslash textendash$, $\backslash textexclamdown$, $\backslash texthyphenchar$, $\backslash textthyphen$, $\backslash textquestiondown$, $\backslash textquotedblleft$

<code>\textquotedblright</code>	Added OML encoding.	509, 523
<code>\textquoteleft</code>	Added the OML encoding.	534
<code>\textquoteright</code>	Made <code>\textless</code> and <code>\textgreater</code>	
1994-10-27 ltoutenc.dtx v1.5d	come from OML.	523
General: Rewrote	Moved math commands here from	
<code>\DeclareTextSymbol</code> to define its	<code>lmath</code>	525
argument to use the current	Removed <code>\textregistered</code>	523
encoding by default, to fit with	Rewrote <code>\copyright</code> to use	
<code>\DeclareTextCommand</code>	<code>\textcircled</code>	523
1994-10-27 ltoutenc.dtx v1.6b	1994-10-31 fontdef.dtx v2.2d	
General: Added <code>\textbackslash</code> . . .	General: Added OMLenc.def ...	768
Added more defaults for OT1. . .	1994-10-31 fontdef.dtx v2.2e	
Removed the enc.def files	General: ... and moved further down	768
Removed the files OT1enc.def,	1994-10-31 ltfloat.dtx v1.1a	
OT1enc.def and OMSenc.def.	<code>\@dblfloat</code> : Major changes since	
Renamed <code>\textlbrace</code> to	two-column and one-column cases	
<code>\textbraceleft</code> and <code>\textrbrace</code>	merged	1029
to <code>\textbraceright</code>	<code>\@dblfloatset</code> : Macro added	1029
1994-10-29 ltmath.dtx 1.0m	Major changes to parameter	
General: ASAJ: Added	parsing, setting of local variables,	
<code>\DeclareMathOperator</code>	etc; two-column and one-column	
ASAJ: Tidied up documentation. . .	cases merged; space hacks moved	1029
1994-10-29 ltmath.dtx v1.0m	<code>\@endfloatbox</code> : (DPC/CAR) Extra	
General: ASAJ: Added	box added to remove color	
<code>\mathellipsis</code> , <code>\mathdollar</code> and	resetting from vmode	1035
<code>\mathsterling</code>	<code>\@floatboxreset</code> : Macro added . .	1033
ASAJ: Removed <code>\dag</code> , <code>\ddag</code>	<code>\@footnotetext</code> : (DPC/CAR) Move	
ASAJ: Renamed <code>\S</code> and <code>\P</code> to	color setting to output routine .	1047
<code>\mathsection</code> and	<code>\@savemarbox</code> : (DPC/CAR) Extra	
<code>\mathparagraph</code> and made them	box added for color	1039
<code>\mathchardefs</code>	<code>\@setfps</code> : Macro added	1030
1994-10-29 ltoutenc.dtx v1.6c	<code>\@xdblfloat</code> : Macros removed:	
General: Added commands like <code>\dots</code>	<code>\@dbfl</code> , <code>\@xdblfloat</code>	1035
for use in text and math.	<code>\@xfloat</code> : (DPC/CAR) Extra box	
Renamed <code>\P</code> , <code>\S</code> , <code>\dag</code> and <code>\ddag</code> to	added to remove color resetting	
<code>\textparagraph</code> , <code>\textsection</code> ,	from vmode	1031
<code>\textdagger</code> and	Major changes, removing setting of	
<code>\textdaggerdbl</code>	local variables, space hacks etc;	
1994-10-30 ltdefns.dtx v1.2c	two-column and one-column cases	
<code>\@onelevel@sanitize</code> : Macro added	merged	1030
General: (CAR) <code>\@onelevel@sanitize</code>	Reset hook added	1031
added	<code>\@xympar</code> : (DPC/CAR) Extra box	
1994-10-30 ltdefns.dtx v1.2f	added since needed for floats . .	1040
General: (DPC) <code>\newwrite</code> 's moved to	<code>\fps@dbl</code> : Macro added	1030
ltfiles	1994-10-31 ltoutput.dtx v1.1a	
1994-10-30 ltmath.dtx v1.0n	<code>\@topnewpage</code> : (DPC/CAR) Extra	
General: ASAJ: Moved the new	box added to remove color	
commands to ltoutenc.	resetting from vmode	1230
1994-10-30 ltoutenc.dtx v1.6d	(DPC/CAR) Use	
General: Added	<code>\color@begingroup</code> for color . .	1230
<code>\DeclareTextCompositeCommand</code> . .	(DPC/CAR) Use <code>\normalcolor</code>	1230
Added <code>\textcircled</code>	1994-11-02 ltoutenc.dtx v1.6d	
Added <code>\t</code>	General: Wrapped lines longer than 70	
Added math commands.	characters.	509

1994-11-03 ltclass.dtx v1.0k	1994-11-04 ltpage.dtx v1.0e
General: Move <code>\@missingfileerror</code>	<code>\markright</code> : Added
to <code>ltfiles</code> 1098	<code>\@unexpandable@protect</code> .
1994-11-03 ltdirchk.dtx v1.0i	ASAJ. 1090
General: Generate an error if <code>latex.ltx</code>	1994-11-04 ltsect.dtx 1.0h
not used with <code>clean initex</code> 1	<code>\@sect</code> : (ASAJ) Added
1994-11-03 ltfiles.dtx v1.0j	<code>\protected@edef</code> 1016
<code>\@missingfileerror</code> : Move here from	General: (ASAJ) Added
<code>ltclass</code> 505	<code>\protected@xdef</code> to <code>\thanks</code> . . 1012
1994-11-04 ltboxes.dtx v1.0m	1994-11-04 ltsect.dtx v1.0h
<code>\@mpfootnotetext</code> : Added	<code>\addcontentsline</code> : Added
<code>\protected@edef</code> . ASAJ. 941	<code>\protected@write</code> to
1994-11-04 ltdefns.dtx v1.2e	<code>\addcontentsline</code> . ASAJ. . . . 1021
General: Added	<code>\addtocontents</code> : Added
<code>\set@display@protect</code> to	<code>\protected@write</code> to
<code>\typeout</code> . ASAJ. 80	<code>\addtocontents</code> . ASAJ. 1022
Added commands for setting and	1994-11-04 lttab.dtx v1.0h
restoring <code>\protect</code> . ASAJ. 94	<code>\@mkpream</code> : (ASAJ) Added
Rewrote protected short commands	<code>\@unexpandable@protect</code> to
using <code>\x@protect</code> . ASAJ. 92	<code>\@mkpream</code> 968
1994-11-04 ltterror.dtx v1.2g	<code>\multicolumn</code> : (ASAJ) added
General: Added	<code>\set@typeset@protect</code> 963
<code>\set@display@protect</code> to	1994-11-04 ltxref.dtx v1.1d
<code>\Generic*</code> commands. ASAJ. . . 427	<code>\label</code> : (ASAJ)Added
1994-11-04 ltfiles.dtx v1.0k	<code>\protected@edef</code> 844
<code>\nofiles</code> : Added setting of	(ASAJ)Added <code>\protected@write</code> 844
<code>\protected@write</code> , <code>\makeindex</code>	1994-11-05 ltboxes.dtx v1.0n
and <code>\makeglossary</code> to <code>\nofiles</code> .	<code>\@mpfootnotetext</code> : Color resetting for
ASAJ. 492	footnotes moved to <code>endminipage</code> :
<code>\protected@write</code> : Macro added	as for main page. 941
ASAJ. 492	<code>\color@endbox</code> : macro added for color
1994-11-04 ltfloat.dtx v1.1b	support 929
<code>\@footnotetext</code> : (ASAJ) Added	<code>\color@hbox</code> : macro added for color
<code>\protected@edef</code> 1047	support 929
<code>\footnotemark</code> : Added	<code>\endminipage</code> : Color resetting for
<code>\protected@xdef</code> to	footnotes moved to here: as for
<code>\footnotemark</code> 1048	main page. 940
1994-11-04 ltidxglo.dtx v1.1b	1994-11-05 ltboxes.dtx v1.0o
<code>\@wrglossary</code> : Added	<code>\@mpfootnotetext</code> : Color groups
<code>\protected@write</code> to	restored here. 941
<code>\@wrglossary</code> 1052	1994-11-05 ltfloat.dtx v1.1c
<code>\@wrindex</code> : Added <code>\protected@write</code>	<code>\@dblflset</code> : Add compatibility with
to <code>\@wrindex</code> 1051	old version of <code>\@xfloat</code> 1029
General: Removed <code>\if@files</code> from	<code>\@endfloatbox</code> : Use new <code>\color@hbox</code>
<code>\makeindex</code> 1050	concept. 1035
<code>\makeglossary</code> : Removed <code>\if@files</code>	<code>\@footnotetext</code> : Removed
from <code>\makeglossary</code> 1051	<code>\normalcolor</code> (again) 1047
1994-11-04 ltmiscen.dtx v1.0t	<code>\@savemarbox</code> : Use new <code>\color@hbox</code>
<code>\@writefile</code> : Removed setting of	concept. 1039
<code>\protect</code> . ASAJ. 871	<code>\@setfps</code> : Add compatibility with old
1994-11-04 ltoutenc.dtx v1.6f	version of <code>\@xfloat</code> 1030
General: Added <code>_</code> 524	<code>\@xfloat</code> : Add compatibility with old
Added <code>\mathunderscore</code> 525	version of <code>\@xfloat</code> : but the

arguments, provided at exorbitant cost, are now completely ignored	1030	1994-11-09 ltffsbas.dtx v2.1v	
Use new <code>\color@hbox</code> concept.	1031	<code>\@vpt</code> : (DPC) macros added, from	
<code>\@xympar</code> : Use new <code>\color@hbox</code> concept.	1040	<code>setsize.dtx</code>	599
1994-11-05 ltoutenc.dtx v1.6g		(DPC) reduce save stack usage	
General: Added setting of		<code>latex/1742</code>	599
<code>\@typeset@protect</code> to <code>\patterns</code> and <code>\hyphenation</code> .	521	1994-11-10 ltbibl.dtx v1.1c	
1994-11-05 ltoutput.dtx v1.1b		General: Fix <code>\nocite{*}</code>	1053
<code>\@topnewpage</code> : Use new <code>\color@hbox</code> concept.	1230	<code>\nocite</code> : Fix <code>\nocite{*}</code>	1056
<code>\@writsetup</code> : Change protect settings for new-style, protect-free aux-files.	1254	1994-11-10 ltmath.dtx v1.2v classes	
Use new <code>\color@hbox</code> concept.	1254	<code>eqnarray</code> : Added value of <code>\parskip</code> to <code>\abovedisplayskip</code> to compensate for negative <code>\topsep</code>	906
1994-11-05 ltoutput.dtx v1.1c		1994-11-10 ltoutput.dtx v1.1e	
<code>\@begindivi</code> : Added macro	1260	<code>\@writsetup</code> : Modify <code>\protect</code> setting	1254
<code>\@begindivibox</code> : Added macro	1225	1994-11-10 ltplain.dtx v1.1b	
<code>\@writsetup</code> : Add new <code>\AtBeginDvi</code> concept	1254	General: (CAR) added patch to <code>\loop</code> .	14
<code>\AtBeginDvi</code> : Added macro	1225	<code>\iterate</code> : (CAR) added extra <code>\relax</code>	27
1994-11-06 ltffsbas.dtx v2.1u		1994-11-11 ltspc.dtx v1.2a	
<code>\cf@encoding</code> : New macro	583	<code>\:</code> (DPC) Make robust	467
<code>\DeclareFixedFont</code> : Renamed <code>\every@size</code> to <code>\every@math@size</code> .	573	1994-11-12 ltfontcmd.dtx v3.3o	
1994-11-06 ltffsini.dtx v2.2b		<code>\normalsize</code> : Added <code>\MessageBreak</code>	798
<code>\@setsize</code> : Use <code>\typeset@protect</code>	760	1994-11-12 ltlists.dtx v1.2b ltspc	
1994-11-06 ltfsstrc.dtx v2.3k		<code>\endtrivlist</code> : Changed order of tests to make <code>\@noitemerror</code> correct: end of an era.	916
<code>\glb@currsize</code> : New implementation	679	1994-11-12 ltmiscen.dtx v1.0u	
<code>\try@simples</code> : New implementation	691	<code>center</code> : Changed end macro to <code>\def</code> : safer and consistent	879
<code>\try@size@substitution</code> : New implementation	690	<code>flushleft</code> : Changed end macro to <code>\def</code> : safer and consistent	880
<code>\tryis@simple</code> : New implementation	691	<code>flushright</code> : Changed end macro to <code>\def</code> : safer and consistent	881
1994-11-07 fontdef.dtx v2.2f		1994-11-12 ltplain.dtx v1.1c	
General: (DPC) Add <code>\DeclareMathSizes</code> declarations	773	General: Comment out more encoding specific commands	29
(DPC) Updated to use <code>\ProvidesFile</code>	768	1994-11-12 ltspc.dtx v1.2b	
1994-11-07 ltfiles.dtx v1.0m		<code>\addpenalty</code> : Corrected error message	475
<code>\document</code> : Renamed <code>\every@size</code> to <code>\every@math@size</code> .	488	<code>\addvspace</code> : Corrected error message	474
1994-11-07 ltplain.dtx v1.0l		1994-11-13 ltspc.dtx v1.2c	
<code>\@unused</code> : move here from ltdefs, remove duplicate <code>\@mainaux</code>	21	<code>\addpenalty</code> : Recorrected error message	475
1994-11-07 preload.dtx v2.1e		<code>\addvspace</code> : Recorrected error message	474
General: (DPC) Updated to use <code>\ProvidesFile</code>	787	1994-11-14 ltoutput.dtx v1.1f	
1994-11-09 ltboxes.dtx v1.0p		<code>\@begindivi</code> : Use normal box register: why a box?	1260
<code>\@finalstrut</code> : Revert <code>\finalstrut</code> to 2.09 equivalent (from ltpatch)	945	<code>\@begindivibox</code> : Use normal box register: why a box?	1225
General: more color changes...	926	<code>\@writsetup</code> : Modify new <code>\AtBeginDvi</code> concept	1254
		General: Removed old definition of <code>\@testfp</code> .	1214

1994-11-14 ltspc.dtx v1.2d	1994-11-18 ltfsbas.dtx v2.1x
<code>\:</code> (DPC) Macro modified 467	General: (DPC) use <code>\reserved@f</code> not
1994-11-14 lttab.dtx v1.0i	<code>\next</code> 571
<code>\tabularnewline</code> : (DPC) Macro	1994-11-18 ltfsdcl.dtx v2.1m
added 962	<code>\DeclareMathDelimiter</code> : (DPC)
1994-11-16 fontdef.dtx v2.2h	<code>\expandafter</code> instead of <code>\next</code> . 727
General: (DPC) Removed <code>\{</code> and <code>\}</code> 768	1994-11-18 ltfsstrc.dtx v2.3m
1994-11-17 ltboxes.dtx v1.0q	General: <code>\next</code> to <code>\reserved@f</code> . . . 671
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 926	1994-11-18 ltmath.dtx v1.0p
1994-11-17 ltclass.dtx v1.0l	<code>\phantom</code> : (DPC) color support . . . 891
General: <code>\@tempa</code> to <code>\reserved@a</code> . 1094	(DPC) use <code>\expandafter</code> instead of
1994-11-17 ltcntrl.dtx v1.0b	<code>\next</code> 891
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 423	<code>\prime@s</code> : (DPC) use <code>\@let@token</code>
1994-11-17 ltdefs.dtx v1.0g	instead of <code>\next</code> and
General: <code>\@tempa</code> to <code>\reserved@a</code> . . . 80	<code>\expandafter</code> instead of <code>\nxt</code> . . 896
1994-11-17 ltdirchk.dtx v1.0j	<code>\smash</code> : (DPC) color support 892
General: <code>\@tempa</code> to <code>\reserved@a</code> 1	(DPC) use <code>\expandafter</code> instead of
1994-11-17 ltterror.dtx v1.2h	<code>\next</code> 892
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 427	1994-11-21 ltfloat.dtx v1.1f
1994-11-17 ltfiles.dtx v1.0n	<code>\@endfloatbox</code> : Added reset of
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 485	minipage flag 1035
1994-11-17 ltfinal.dtx v1.0o	Corrected position of
General: <code>\@tempa</code> to <code>\reserved@a</code> . 1332	<code>\outer@nobreak</code> 1035
1994-11-17 ltfloat.dtx v1.1e	<code>\@marginparreset</code> : Macro added . 1039
General: <code>\@tempa</code> to <code>\reserved@a</code> . 1026	<code>\@savemarbox</code> : Added <code>\@setminipage</code>
1994-11-17 ltfntcmd.dtx v3.3p	etc 1039
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 790	Added resetting of size and font 1039
1994-11-17 ltfsbas.dtx v2.1w	Changed to <code>\color@vbox</code> 1039
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 571	Use <code>\@setnobreak</code> etc 1039
1994-11-17 ltfsdcl.dtx v2.1m	<code>\@setminipage</code> : Macro added 1033
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 702	<code>\@setnobreak</code> : Macro added 1033
1994-11-17 ltfsstrc.dtx v2.3l	<code>\@xfloat</code> : Added <code>\@setminipage</code> . 1031
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 671	Added resetting of size and font 1031
1994-11-17 ltmath.dtx v1.0o	Changed to <code>\color@vbox</code> so that
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 889	large floats overflow at the
1994-11-17 ltmiscen.dtx v1.0v	bottom 1031
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 864	Missing percents reinserted after 4,
1994-11-17 ltoutenc.dtx v1.6h	8: these are not numbers. 1030
General: (DPC) <code>\@tempa</code> to	Use <code>\@setnobreak</code> 1031
<code>\reserved@a</code> 509	<code>\@xympar</code> : Changed to <code>\color@vbox</code> 1040
1994-11-17 ltoutput.dtx v1.1h	1994-11-21 ltoutput.dtx v1.1i
General: <code>\@tempa</code> to <code>\reserved@a</code> . 1214	<code>\@addtocurcol</code> : Added <code>\if@nobreak</code>
1994-11-17 ltpictur.dtx v1.0f	test before float box 1270, 1276
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 974	<code>\@specialoutput</code> : Added <code>\if@nobreak</code>
1994-11-17 ltsect.dtx v1.0i	test 1237
General: <code>\@tempa</code> to <code>\reserved@a</code> . 1012	<code>\@topnewpage</code> : Changed to
1994-11-17 lttab.dtx v1.0j	<code>\color@vbox</code> 1230
General: <code>\@tempa</code> to <code>\reserved@a</code> . . 947	1994-11-22 ltfsdcl.dtx v2.1o
1994-11-18 ltboxes.dtx v1.0r	General: wrap long lines 702
<code>\color@vbox</code> : macro added for color	1994-11-22 ltoutenc.dtx v1.6i
support 929	General: Corrected <code>\dots</code> so that
1994-11-18 ltfinal.dtx v1.0n	there's no kerning in monowidth
General: re-allow slots 127–255 . . . 1346	fonts. 509

Corrected typo with <code>\mathunderscore</code>	509	Rewrote <code>\@text@composite</code> so it allows an empty argument, or an argument containing lots of commands.	516
Fixed empty accents. Again. . . .	509		
1994-11-24 ltdefs.dtx v1.2h <code>\@newenv</code> : Added test for <code>\endgraf</code> . .	88	1994-12-01 ltfinal.dtx v1.0p General: Renamed <code>lthyphen.*</code> to <code>hyphen.*</code>	1332
1994-11-25 ltplain.dtx v1.1f General: (DPC) Comment out lots of obsolete code	14	1994-12-01 lthyphen.dtx v1.0g General: Rename <code>lthyphen.ltx/cfg</code> to <code>hyphen.ltx/cfg</code>	1330
1994-11-26 ltfloat.dtx v1.1b <code>\footnote</code> : (ASAJ) Added <code>\protected@xdef</code>	1046	1994-12-01 ltplain.dtx v1.1g General: (DPC) More doc changes . .	14
1994-11-28 ltcntrl.dtx v1.0c General: Documentation improvements	423	1994-12-02 fontdef.dtx v2.2i General: Commented out <code>\ldots</code> . ASAJ.	766
1994-11-30 ltfiles.dtx v1.0o <code>\@dofilelist</code> : Macro added	508	1994-12-02 ltssini.dtx v2.2c <code>\copyright</code> : <code>\copyright</code> is now in <code>ltoutenc</code> . ASAJ	761
<code>\listfiles</code> : Use <code>\@dofilelist</code> . . .	507	1994-12-02 ltlists.dtx v1.0e <code>\@trivlist</code> : RmS: Added check for looping	915
<code>\nofiles</code> : There is no <code>\@gobblethree</code>	492	1994-12-02 ltoutenc.dtx 1.7b General: Redefined <code>\a</code> properly. . . .	522
1994-11-30 ltssbas.dtx v2.1y <code>\fontshape</code> : Use <code>\@current@cmd</code> in <code>\@enc@update</code> . ASAJ.	582	1994-12-02 ltoutenc.dtx v1.7b General: Fixed a bug with <code>\a</code>	509
1994-11-30 ltmath.dtx 1.0q General: ASAJ: <code>\DeclareMathOperator</code> moved to <code>AMSLATEX</code>	889	1994-12-04 lthyphen.dtx v1.0h General: Documentation edits for /1989	1330
1994-11-30 ltmiscen.dtx v1.0w <code>\@enddocument@kernel@warnings</code> : (DPC) Do warnings even for <code>\nofiles</code>	867	1994-12-05 ltoutenc.dtx v1.7c General: Added braces to <code>\textcircled</code>	509
<code>\enddocument</code> : (DPC) Use <code>\@dofilelist</code>	866	1994-12-06 ltssbas.dtx v2.1z <code>\DeclareFontEncoding</code> : use <code>\nfss@catcodes</code>	575
1994-11-30 ltoutenc.dtx 1.7a General: Redefined <code>\a</code> for the new scheme.	522	<code>\nfss@catcodes</code> : Added tab char as well	589
1994-11-30 ltoutenc.dtx v1.6g General: Removed new definitions of <code>\patterns</code> and <code>\hyphenation</code> , since encoding-specific commands now expand in the mouth.	521	1994-12-08 ltoutenc.dtx v1.7d General: Added <code>\null</code> and <code>\sh@ft</code> to <code>\b</code> and <code>\d</code>	509
1994-11-30 ltoutenc.dtx v1.7a General: Added new code for encoding-specific commands. These now expand in the mouth, which means that ligaturing and kerning can happen.	509	1994-12-08 lttab.dtx v1.0k <code>\@array</code> : Add <code>\tabularnewline</code> . . .	961
Always load the <code>enc.def</code> file, so that the default encoding for the commands will change.	553	<code>\tabularnewline</code> : (DPC) Made it <code>\relax</code>	962
Redefined <code>\@changed@cmd</code> to expand in the mouth.	513	1994-12-09 ltbibl.dtx v1.1d <code>\bibliographystyle</code> : (DPC) Allow use in preamble.	1056
Removed <code>\@changed@x@mouth</code> since <code>\@changed@x</code> now expands in the mouth.	513	1994-12-10 ltfloat.dtx v1.1g <code>\@dblfloat</code> : Old version reinstated temporarily	1029
		<code>\@dblflset</code> : Macro removed temporarily	1029
		Old version reinstated temporarily	1029

<code>\setfyps</code> : Macro removed	1995-04-02 ltffssini.dtx v2.2d
temporarily 1030	<code>\not@math@alphabet</code> : add <code>\noexpand</code> to second part of message 759
<code>\xdblfloat</code> : Macros reinserted	1995-04-21 ltclass.dtx v1.0m
temporarily 1035	<code>\DeclareOption*</code> : Made long /1498 1112
<code>\xfloat</code> : Old version reinstated	<code>\endfilecontents</code> : Close input check stream: latex/1487 1131
temporarily 1030	1995-04-21 ltfinal.dtx v1.0q
Sanitization added temporarily . . 1030	General: Allow initial patch level 0 1350
General: Some temps reinserted	1995-04-21 ltoutenc.dtx v1.7h
temporarily 1026	General: Added <code>\null \k</code> latex/1274 509
<code>\fyps@dbl</code> : Macro removed	1995-04-22 ltfiles.dtx v1.0p
temporarily 1030	<code>\includeonly</code> : Allow blanks in argument 493
1994-12-10 ltfntcmd.dtx v3.3q	1995-04-22 ltmiscen.dtx v1.0x
<code>\@math@egroup</code> : Don't read arguments 797	General: Removed extra def of <code>\gobble</code> 864
<code>\check@nocorr@</code> : Use <code>\space</code> command for comparison 794	1995-04-23 ltsect.dtx v1.0j
1994-12-10 ltssdcl.dtx v2.1p	<code>\addcontentsline</code> : Use <code>\contentsline</code> internally. 1021
<code>\document@select@group</code> : Surround with braces (add fourth arg) . . . 708	1995-04-24 ltbibl.dtx v1.1e
<code>\select@group</code> : Surround with braces (add fourth arg) 705	<code>\citex</code> : Add <code>\mbox</code> to undefined case: latex/1239. 1055
1994-12-10 ltoutenc.dtx v1.7e	1995-04-24 ltbibl.dtx v1.1f
General: Added documentation for the OML encoding. 509	<code>\bibcite</code> : Make <code>\onlypreamble</code> /1388. 1054
Replaced width with <code>\@width</code> and ditto height in vrules. 509	1995-04-24 ltctrl.dtx v1.0d
1994-12-14 ltoutenc.dtx v1.7f	<code>\@for</code> : Don't expand second argument with <code>\edef</code> : /1317 (DPC) 426
General: Added braces to <code>\copyright</code> so it works unbraced in subscripts. 509	1995-04-24 ltoutput.dtx v1.1j
Added check for math mode in <code>\@changed@cmd</code> 509	<code>\fl@tracemessage</code> : Do not add to kernel unless 'trace' specified . . 1292
Commented out <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> , <code>\textthyphenchar</code> , <code>\textthyphen</code> and <code>\textless</code> to save memory. 509	1995-04-24 ltoutput.dtx v1.1l
1995-01-12 ltmath.dtx v1.2y classes	<code>\@begindivbox</code> : Add <code>\vbox</code> latex/1392 1225
<code>\@eqnnum</code> : Added <code>\normalcolor</code> . . . 904	<code>\@writeseup</code> : Reset <code>\</code> latex/1451 (DPC) 1255
1995-03-03 ltoutenc.dtx 1.7g	1995-04-24 ltpage.dtx v1.0f
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment. 522	<code>\fussy</code> : reset <code>\emergencystretch</code> latex/1344 1093
1995-04-02 ltfntcmd.dtx v3.3r	1995-04-24 ltplain.dtx v1.1h
<code>\@math@egroup</code> : Read them again to be able to add <code>\relax</code> 797	<code>\newlanguage</code> : Remove remaining <code>\outer</code> declarations. 17
1995-04-02 ltssdcl.dtx v2.1q	1995-04-24 ltxref.dtx v1.1e
<code>\document@select@group</code> : fix problem for pr/1275 708	<code>\newlabel</code> : Make <code>\onlypreamble</code> for /1388. 843
<code>\select@group</code> : fix problem for pr/1275 705	1995-04-25 ltdefs.dtx v1.2i
<code>\set@mathdelimiter</code> : fix pr/1329 . . 730	<code>\@check@c</code> : Make <code>\long</code> for latex/1346 90
	<code>\new@environment</code> : Parse arguments slowly but safely /1507 88
	1995-04-25 ltfiles.dtx v1.0q
	<code>\document</code> : Removed execution of <code>\every@size</code> latex/1407 488

1995-04-25 ltsect.dtx v1.0k	1995-05-07 ltoutput.dtx v1.1m
\@dottedtocline: Added \hbox	General: Use \hb@xt@. 1214
around dots. 1024	1995-05-07 ltpictur.dtx v1.0g
1995-04-27 ltboxes.dtx v1.0s	General: Use \hb@xt@. 974
\@frameb@x: Move \leavevmode for	1995-05-07 ltplain.dtx v1.1j
graphics/1512 934	General: Use \hb@xt@. 14
\@iframebox: Move \leavevmode for	1995-05-07 ltsect.dtx v1.0o
graphics/1512 933	General: Use \hb@xt@. 1012
\@iirsbox: Move \leavevmode for	1995-05-07 lttab.dtx v1.0l
graphics/1512 944	General: Use \hb@xt@. 947
\@irsbox: Move \leavevmode for	1995-05-08 ltbibl.dtx v1.1g
graphics/1512 944	\@citex: Use \@firstofone 1055
\fbbox: Move \leavevmode for	\@bibitem: Removed unnecessary
graphics/1512 933	braces 1054
\raisebox: Move \leavevmode for	\@nocite: Use \@firstofone 1056
graphics/1512 944	1995-05-08 ltdefs.dtx v1.2k
1995-04-27 ltfiles.dtx v1.0r	\typein: Use \@firstofone 82
\document: Added \global to support	1995-05-08 ltdefs.dtx v1.2l
groups in hook 489	\typein: Remove unnecessary braces 82
1995-04-27 ltmiscen.dtx v1.0y	Replace \def by \let 82
\enddocument: \@checkend moved	1995-05-08 ltfstrc.dtx v2.3n
after hook 865	\ifnot@nil: Use \@firstofone 685
1995-04-27 ltplain.dtx v1.1i	1995-05-11 fontdef.dtx v2.2j
General: Move \hang and	General: Updates to some plain
\textindent to latex209.def 29	macros 766
1995-04-29 ltcntrl.dtx v1.0e	1995-05-12 ltclass.dtx v1.0n
General: Moved init of \protect to	\DeclareOption*: Use \toks@ to
ltdefs.dtx 426	remove need to double hash
Removed unused defs for	/1557 1112
\@setprotect and	1995-05-12 ltfloat.dtx v1.1h
\@resetprotect 426	\@footnotemark: Add \nobreak to
1995-04-29 ltdefs.dtx v1.2j	allow hyphenation. latex/1605 1048
\protect: Init \protect here 95	1995-05-12 ltpictur.dtx v1.0h
1995-04-29 ltpar.dtx v1.1b	\pictur@: Macro added for
General: (TO) Comments clean-up. 438	latex/1355 976
1995-05-02 ltsect.dtx v1.0l	1995-05-12 ltvers.dtx v1.0e
\@dottedtocline: Don't reset to	General: Add autoload docstrip guards 35
\rmfamily 1024	Check for format older than 1 year 35
1995-05-03 ltsect.dtx v1.0m	1995-05-13 ltfstrc.dtx v2.3o
General: TO: Promoted	General: Use single hash mark in
documentation to doc.sty	\DeclareOption 672
standard 1012	1995-05-16 ltfloat.dtx v1.1i
1995-05-06 ltsect.dtx 1.0n	\@makefnmark: Now use
\@secCNTformat: Use \quad instead of	\textsuperscript. 1044
\hskip 1018	\textsuperscript: Command
\@sect: Added \relax after	added./pr1503 1044
\@secCNTformat just in case 1016	\thefootnote: Streamlined parts of
1995-05-07 ltboxes.dtx v1.0t	code. 1043
General: Use \hb@xt@. 926	1995-05-17 ltboxes.dtx v1.0u
1995-05-07 ltdefs.dtx v1.2k	\@irsbox: Removed surplus braces 944
\hb@xt@: Macro added 81	1995-05-17 ltdefs.dtx v1.0o
1995-05-07 ltmath.dtx v1.0r	\g@addto@macro: Make long for
General: Use \hb@xt@. 889	latex/1522 114

1995-05-17 ltlists.dtx v1.0g	1995-05-23 ltffssini.dtx v2.2e
\@item: Removed surplus braces . . . 921	\newfont: Font assignment made local
\@nbitem: Removed surplus braces . . 922	again. 759
enumerate: Use \thr@@ and remove	1995-05-24 ltdefs.dtx v1.1l
surplus braces 923	\newif: (DPC) New implementation . 89
itemize: Use \thr@@ 923	1995-05-24 ltdefs.dtx v1.2m
1995-05-18 ltfloat.dtx v1.1j	\typein: (DPC) New implementation 82
\makefnmark: Added \normalfont. 1044	1995-05-24 ltfloat.dtx v1.1l
\thempfootnote: Added \itshape. 1043	\@textsuperscript: Command
1995-05-19 ltpictur.dtx v1.1a	added. 1044
General: Support autoloading feature 974	General: Moved definition of \footins
1995-05-20 ltcounts.dtx v1.1b	and \footnoterule from ltplain. 1043
\@definecounter: Streamlined code 559	\textsuperscript: Use
\@fnsymbol: Allowing both text and	\@textsuperscript 1044
math 567	1995-05-24 ltffssbas.dtx v3.0a
\fnsymbol: Streamlined code 566	General: (DPC) Make file from
1995-05-20 ltcounts.dtx v1.1c	previous file, fam.dtx 1995/05/20
\@definecounter: And do it right . 559	v2.2d 571
1995-05-20 ltfloat.dtx v1.1k	\mathgroup: (DPC) No need to
\makefnmark: Moved \normalfont	redefine \newfam as not outer . . 571
back and use	1995-05-24 ltffsscmp.dtx v3.0a
\@textsuperscript 1044	General: (DPC) Make file from
Moved \normalfont to	previous file, fam.dtx 1995/05/20
\textsuperscript 1044	v2.2d 697
\textsuperscript: Use	1995-05-24 ltffssdcl.dtx v3.0a
\normalfont. 1044	General: (DPC) Make file from
1995-05-21 ltffssdcl.dtx v2.1t	previous file, latint.dtx 1995/05/21
\DeclareMathRadical: Allow for	v2.1t 702
undefined cs names 731	1995-05-24 ltffssini.dtx v3.0a
1995-05-21 ltlists.dtx v1.0f	General: (DPC) Make file from
General: Moved to doc.sty standard 907	previous file, lfonts.dtx 1995/05/23
1995-05-21 ltmath.dtx v1.0r	v2.2e 735
\@sqrt: Use \sqrtsign 900	\cal: (DPC) Remove definition . . . 765
General: Remove \mathhexbox from	\mit: (DPC) Remove definition . . . 765
this file 894	1995-05-24 ltffsstrc.dtx v3.0a
Update some plain macros 889	General: (DPC) Make file from
\lefteqn: Use \rlap 903	previous file, tracefnt 1995/05/16
\@@t: Use \sqrtsign instead of	v2.3o 671
\sqrt 891	1995-05-24 ltffsstrc.dtx v3.0b
1995-05-21 ltoutenc.dtx v1.7h	General: (DPC) Fix \ProvidesFile
\@inmathwarn: Added several	usage 671
\@onlypreamble 514	1995-05-25 ltclass.dtx v1.0p
1995-05-21 ltoutenc.dtx v1.7j	\endfilecontents: Delete
General: Updated some plain macros 526	\filecontents after preamble . 1131
1995-05-21 ltplain.dtx v1.1j	1995-05-25 ltfilehook.dtx v1.0t
General: Moved some code to other	\unqu@tefilef@und: (CAR) added
files 14	\long 1170
1995-05-22 ltplain.dtx v1.1k	1995-05-25 ltfiles.dtx v1.0s
General: Definitions of \footins and	\document: Added check for \topskip
\footnoterule moved to ltfloat. . 30	zero 489
1995-05-22 lttab.dtx v1.1a	1995-05-25 ltfiles.dtx v1.0t
General: Support autoloading feature 947	\@iffilenamepath: (CAR) added \long 502
	\document: Corrected typo 489
	\IfFileExists@: (CAR) added \long 500

<code>\nofiles:</code> (CAR) added <code>\long</code>	492	Save some tokens in	
<code>\protected@write:</code> (CAR) added		<code>\textvisiblespace</code> and	
<code>\long</code>	492	<code>\textunderscore.</code>	523
1995-05-25 <code>ltfloat.dtx</code> v1.1m		1995-06-06 <code>ltfinal.dtx</code> v1.0s	
<code>\@savemarbox:</code> (CAR) Resettings		General: Made <code>\MakeUppercase</code> and	
moved to hook	1039	<code>\MakeLowercase</code> brace their	
<code>\@xfloat:</code> (CAR) Resettings moved to		argument.	1332
hook	1031	1995-06-09 <code>ltoutenc.dtx</code> v1.7l	
1995-05-25 <code>ltlists.dtx</code> v1.0i		<code>\DeclareTextComposite:</code> Rewrote	
<code>\endtrivlist:</code> Macros moved from		<code>\DeclareTextComposite</code> to define	
<code>ltspace.dtx</code>	916	the composite as a no-argument	
1995-05-25 <code>ltmath.dtx</code> v1.3c classes		command rather than a	
<code>\@eqnnum:</code> replace		two-argument command.	517
<code>\reset@font\rmfamily</code> with		1995-06-11 <code>ltspace.dtx</code> v1.2g	
<code>\normalfont</code> (PR 1578)	904	<code>\restorecr:</code> (CAR) <code>\relax</code> added to	
1995-05-25 <code>ltspace.dtx</code> v1.2f		stop silent eating of *.	483
<code>\@vbsphack:</code> (CAR) not used so		1995-06-13 <code>ltfinal.dtx</code> v1.0t	
‘removed’.	473	General: Add patch level string more	
<code>\@vspacer:</code> (CAR) <code>\@restorepar</code>		carefully	1350
added to avoid possible infinite tail		Call <code>\errorstopmode</code>	1353
recursion caused by a typo in the		1995-06-13 <code>ltpictur.dtx</code> v1.1b	
argument.	478	General: Use <code>\ProvidesFile</code> in	
(CAR) macros modified to be more		autoload	974
efficient	478	1995-06-14 <code>lftab.dtx</code> v1.1b	
General: Macros moved to <code>ltlists.dtx</code>	463	General: Use <code>\ProvidesFile</code> in	
1995-05-26 <code>ltdefns.dtx</code> v1.2n		autoload	947
<code>\@gobblefour:</code> (CAR) Added <code>\longs</code>	90	1995-06-15 <code>ltfssbas.dtx</code> v3.0c	
1995-05-26 <code>ltmath.dtx</code> v1.0s		General: (DPC) minor documentation	
<code>\@eqnnum:</code> Removed <code>\rmfamily</code> (PR		changes	571
1578), replaced <code>\reset@font</code> with		1995-06-15 <code>ltfsscmp.dtx</code> v3.0b	
<code>\normalfont</code>	900	General: (DPC) minor documentation	
1995-05-26 <code>ltpage.dtx</code> v1.0g		edits	697
<code>\ps@plain:</code> removed <code>\rmfamily</code> (PR		1995-06-15 <code>ltfssdcl.dtx</code> v3.0b	
1578)	1090	General: (DPC) minor documentation	
1995-05-27 <code>ltfssbas.dtx</code> v3.0b		changes	702
<code>\mathgroup:</code> (FMi) But a need to		1995-06-19 <code>ltbibl.dtx</code> v1.1h	
define <code>\new@mathgroup</code>	571	<code>\biblecite:</code> Call <code>\@newl@bel</code> so	
1995-06-05 <code>fontdef.dtx</code> v2.2k		repeated keys produce better	
General: Moved math commands from		warning.	1054
<code>ltoutenc.dtx.</code>	784	1995-06-19 <code>ltclass.dtx</code> v1.0q	
1995-06-05 <code>ltfinal.dtx</code> v1.0r		<code>\documentclass:</code> Don’t redefine	
General: Added <code>\MakeUppercase</code> and		<code>\usepackage</code> in compat mode for	
<code>\MakeLowercase.</code>	1332	/1634	1116
1995-06-05 <code>ltoutenc.dtx</code> v1.7k		1995-06-19 <code>ltxref.dtx</code> v1.1e	
<code>\@inmathwarn:</code> Removed		<code>\newlabel:</code> Use <code>\@newl@bel</code> to share	
<code>\protected@cmd</code> and replaced with		code with <code>\biblecite</code>	843
explicit <code>\noexpand.</code>	514	1995-06-28 <code>ltfssini.dtx</code> v3.0b	
General: Allowed		General: (DPC) Fix documentation	
<code>\ProvideTextCommandDefault</code>		typos	735
after the preamble.	515	1995-06-28 <code>ltmath.dtx</code> v1.0t	
Commented out <code>\textless</code> and		General: minor doc edits	889
<code>\textgreater.</code>	523	1995-07-02 <code>ltplain.dtx</code> v1.1n	
Moved math commands to		General: Removed surplus ‘by’ and ‘=’	
<code>fontdef.dtx.</code>	525	in various places	14

<code>\offinterlineskip</code> : Replaced 1000 by <code>\@m</code>	27
<code>\showoutput</code> : Use <code>\showoverfull</code> to save space	31
<code>\tracingall</code> : Use <code>\showoutput</code> to save space	31
1995-07-03 <code>ltdefns.dtx</code> v1.2o <code>\set@typeset@protect</code> : Use <code>\@typeset@protect</code> for init	94
1995-07-03 <code>ltfntcmd.dtx</code> v3.3s <code>\tst@ic</code> : Use clean interface for jump	796
1995-07-05 <code>ltfntcmd.dtx</code> v3.3s <code>\tst@ic</code> : Renamed from <code>\test@next</code>	796
1995-07-05 <code>ltspace.dtx</code> v1.2h <code>\@gnewline</code> : Use <code>\break</code>	469
<code>\@no@pgbk</code> : Macro replaces <code>\@pgbk</code> and <code>\@nopgbk</code>	467
<code>\nopagebreak</code> : Reimplemented both using <code>\@no@pgbk</code>	466
1995-07-09 <code>ltntrl.dtx</code> v1.0f <code>\iforloop</code> : Reimplemented using Kabelschacht method	426
<code>\iwhiledim</code> : Reimplemented using Kabelschacht method	424
<code>\iwhilenum</code> : Reimplemented using Kabelschacht method	424
<code>\iwhilesw</code> : Reimplemented using Kabelschacht method	424
<code>\@tfor</code> : Reimplemented using Kabelschacht method	426
1995-07-09 <code>ltlists.dtx</code> v1.0j <code>enumerate</code> : Use <code>\expandafter</code>	923
<code>itemize</code> : Use <code>\expandafter</code>	924
1995-07-12 <code>ltpictur.dtx</code> v1.1d General: allow 2e commands in 209 mode. <code>latex/1737</code>	974
1995-07-13 <code>ltdefns.dtx</code> v1.0p General: Updates to documentation	80
1995-07-13 <code>ltfiles.dtx</code> v1.0u General: Updates to docu	485
1995-07-13 <code>ltfssbas.dtx</code> v3.0d <code>\@defaultsubs</code> : macro added	594
<code>\@defaultsubs</code> : macro added	594
General: minor documentation changes	571
<code>\wrong@fontshape</code> : Change a macro not a switch to flag default font substitutions	593
1995-07-13 <code>ltmiscen.dtx</code> v1.0z <code>\@centercr</code> : Use <code>\nobreak</code>	878
<code>\@enddocument@kernel@warnings</code> : Use <code>\@defaultsubs</code> instead of switch	867
<code>\@writefile</code> : Added missing percent and use <code>\relax</code> in the THEN case	871
<code>\@xobeysp</code> : Use <code>\nobreak</code>	881
General: Improve Documentation	864
<code>\enddocument</code> : Set <code>\@setckpt</code> to <code>\@gobbletwo</code> instead of defining it by hand	865
Shorten redefinition of <code>\bibcite</code> and <code>\newlabel</code>	866
1995-07-14 <code>ltbibl.dtx</code> v1.1i <code>\bibcite</code> : Remove <code>\@onlypreamble</code> so still defined in new <code>\enddocument</code>	1054
1995-07-14 <code>ltxref.dtx</code> v1.1g <code>\newlabel</code> : Remove <code>\@onlypreamble</code> so still defined in new <code>\enddocument</code>	843
1995-07-19 <code>ltfssini.dtx</code> v3.0d General: (DPC) TeX2 support	764
1995-07-20 <code>ltboxes.dtx</code> v1.0v <code>\@isavebox</code> : Use <code>\sbox</code>	931
<code>\@isavepicbox</code> : Use <code>\sbox</code>	931
1995-07-21 <code>ltoutput.dtx</code> v1.1o <code>\@writesetup</code> : Command added	1254
New, experimental, versions: need in-lining	1254
1995-08-09 <code>ltmath.dtx</code> v1.0u General: Added code for class options <code>leqno</code> and <code>fleqn</code>	904
1995-08-11 <code>ltlength.dtx</code> v1.1b General: Doc typos fixed for <code>latex/753</code>	569
1995-08-16 <code>ltntrl.dtx</code> v1.0g <code>\@break@tfor</code> : Made long	426
<code>\@forloop</code> : Made defs long	426
<code>\@fornoop</code> : Made defs long	426
<code>\@iforloop</code> : Made defs long	426
<code>\@iwhiledim</code> : Made defs long	424
Removed <code>\@whilenoop</code>	424
<code>\@iwhilenum</code> : Made defs long	424
Removed <code>\@whilenoop</code>	424
<code>\@iwhilesw</code> : Removed <code>\@whileswnoop</code>	424
<code>\@tfor</code> : Made defs long	426
1995-08-16 <code>ltfiles.dtx</code> v1.0v <code>\document</code> : set <code>\@maxdepth</code>	489
set <code>\do</code> globally	489
set <code>\topskip</code> globally	489
1995-08-24 <code>ltfssbas.dtx</code> v3.0f General: Added autoload code	571
1995-08-24 <code>ltfstrc.dtx</code> v3.0c General: Macro <code>\gobble@font@spec</code> removed	685
<code>\tryis@simple</code> :	692

1995-08-25 ltoutput.dtx v1.1p	1995-10-10 ltssdcl.dtx v3.0c
General: Support autoloading feature (FMi).	\@non@alpherr: (DPC) autoload error message
1995-09-01 lterror.dtx v1.2i	1995-10-10 ltplain.dtx v1.1r
General: Add autoload support . . .	General: Autoload tracing code
1995-09-01 ltplain.dtx v1.1m	1995-10-10 ltthm.dtx v1.0f
\empty: Use \let to save space	General: Make \newtheorem ‘only preamble’
\I: Use \let to save space	1995-10-11 ltoutput.dtx v1.1r
1995-09-14 ltplain.dtx v1.1o	\clearpage: Added a check so that it does not lose the argument of \twocolumn[...]
General: Moved \multispan to lttab.dtx	1995-10-16 ltbibl.dtx v1.1j
1995-09-14 lttab.dtx v1.1c	\cite: (DPC) Make robust
\cline: (DPC) New implementation	1995-10-16 ltboxes.dtx v1.0w
1995-09-15 ltssini.dtx v3.0e	General: Clarify makebox description
General: (DPC) Modify TeX2 message	1995-10-16 ltdefs.dtx v1.2u
1995-09-19 ltmiscen.dtx v1.1a	\@ifstar: (DPC) New implementation, for /1910
\verb: Put \@noligs after \verbatim@font where it belongs.	\new@command: (DPC) Use \@testopt /1911
1995-10-01 ltfiles.dtx LaTeX2e	\new@environment: (DPC) Use \@testopt /1911
\@addtofilelist: Macro added . . .	\typein: (DPC) Use \@testopt /1911
1995-10-02 ltdefs.dtx v1.2q	1995-10-16 ltssini.dtx v3.0f
\@ifnch: Use \@let@token for internal/924, save \reserved@e .	\reset@font: Added \relax after \usefont, as the latter eats up spaces.
\@ifnextchar: Use \@let@token . .	1995-10-16 ltmath.dtx v1.0y
\@protected@testopt: Macro added .	\@yeqncr: (DPC) Use \@testopt /1911
\@testopt: Macro added	\sqrt: (DPC) Make robust /1808 . .
\@xargdef: New implementation, using \@test@opt	1995-10-16 ltspc.dtx v1.2j
1995-10-03 fontdef.dtx v2.2l	\@nolinebreak: (DPC) Use \@testopt /1911
General: \@@sqrt from patch file for /1701	\@nolinebreak: (DPC) Use \@testopt /1911
1995-10-03 ltdefs.dtx v1.2r	1995-10-16 ltthm.dtx v1.0g
\typein: Add missing \@typein for /1710 (from patch file)	General: Revert to previous \newtheorem behaviour
1995-10-03 ltpictur.dtx v1.1e	1995-10-17 ltclass.dtx v1.0r
General: New autoload code	\@providesfile: Delay definition of \ProvidesFile till ltfinal
1995-10-04 ltssbas.dtx v3.0g	\ProcessOptions*: Reset \CurrentOption for graphics/1873
General: Modify autoload code	1995-10-17 ltdirchk.dtx v1.0l
1995-10-04 ltssstrc.dtx v3.0d	General: Modify initex version of \ProvidesFile
General: (DPC) Modify autoload code	1995-10-17 ltfinal.dtx v1.0v
1995-10-04 lttab.dtx v1.1d	\@providesfile: reset macro
General: Modify autoload support . .	\reserved@b: reset here after the \input above
1995-10-06 ltfiles.dtx v1.0w	
\@missingfileerror: Autoload error	
1995-10-09 lterror.dtx v1.2j	
General: Modify autoload support . .	
1995-10-09 ltoutenc.dtx v1.7m	
\@inmathwarn: Autoload error	
1995-10-10 ltssbas.dtx v3.0h	
\showhyphens: Use \normalfont and make color safe, and autoloadable	

1995-10-17 ltplain.dtx v1.1s	Use <code>\@refundefined</code> instead of switch	867
<code>\eject</code> : Move <code>\supereject</code> to compat file		28
1995-10-17 lttab.dtx v1.1e	<code>\@cline</code> : (DPC) Use <code>\@multicnt</code>	971
<code>\@multispan</code> : (DPC) Macro added.		972
1995-10-19 ltfinal.dtx v1.0w	<code>\@filelist</code> : Move after <code>\reserved@a</code> setting	1352
1995-10-20 ltbibl.dtx v1.1k	<code>\@citex</code> : Removed <code>refundefined</code> flag	1055
<code>\nocite</code> : Removed <code>refundefined</code> flag		1056
1995-10-20 ltclass.dtx v1.0s	<code>\@begindocumenthook</code> : Make setting conditional, for autoloader version	1128
1995-10-20 ltffsbas.dtx v3.0i	General: (DPC) Modify autoloader code, change <code>\undefined</code>	571
1995-10-20 ltffsstrc.dtx v3.0e	General: (DPC) Modify autoloader code	671
1995-10-22 ltffsbas.dtx v3.0j	General: (RmS) New size function macro <code>\genb@sfcnt</code> needs to be disabled at <code>\document.</code>	571
1995-10-22 ltffsstrc.dtx v3.0f	General: Added ‘genb’ and ‘sgenb’ size functions to support new DC font naming scheme.	671
1995-10-23 lttab.dtx v1.1f	<code>\@settab</code> : (CAR)Ensure that <code>\@hightab</code> increases by at most one	954
<code>\@startline</code> : (CAR)Ensure that <code>\@nxttabmar</code> is never larger than <code>\@hightab</code>		952
<code>\@poptabs</code> : (CAR)Ensure that <code>\@curtab</code> is never larger than <code>\@hightab</code>		955
<code>\@tabbing</code> : (CAR)Make <code>\@hightab</code> consistently a local variable		954
1995-10-24 ltfiles.dtx v1.1a	<code>\document</code> : Removed <code>multiplelabels</code> switch	488
Removed <code>refundefined</code> switch		488
1995-10-24 ltffsbas.dtx v3.0k	<code>\@defaultsubs</code> : macro removed	594
<code>\wrong@fontshape</code> : Make this code inline since it happens only here		593
1995-10-24 ltmiscen.dtx v1.1b	<code>\@enddocument@kernel@warnings</code> : Changed logic for producing warning messages and removed switch	867
1995-10-24 ltxref.dtx v1.1h	<code>\@multiplelabels</code> : Switch for <code>multiplelabels</code> removed	844
<code>\@newl@bel</code> : Switch for <code>multiplelabels</code> replaced by inline code		843
<code>\@refundefined</code> : Switch for <code>refundefined</code> replaced		841
<code>\@setref</code> : Switch for <code>refundefined</code> renamed		842
<code>\if@multiplelabels</code> : Macro removed		844
1995-10-25 ltalloc.dtx v1.1b	General: General doc improvements	421
1995-10-25 ltfloat.dtx v1.1n	<code>\@endfloatbox</code> : (CAR) macro added: to unify code for double and single versions	1035
<code>\end@dblfloat</code> : (CAR) unify code for double and single versions		1034
<code>\end@float</code> : (CAR) unify code for double and single versions		1033
1995-10-25 ltidxglo.dtx v1.1d	General: Doc cleanup	1050
1995-10-25 ltsect.dtx v1.0q	<code>\subparagraphmark</code> : Use <code>\let</code> not <code>\def</code> to save space.	1020
1995-10-27 ltpictur.dtx v1.1f	General: Move initialization to kernel from autoloader file	1001
1995-10-31 ltboxes.dtx v1.0x	<code>\@finalstrut</code> : Add <code>\nobreak</code> in horizontal mode to allow hyphenation.	945
1995-11-01 fontdef.dtx v2.2m	General: add <code>\nfss@catcodes</code> for internal/1932	769
1995-11-01 ltdirchk.dtx v1.0n	General: Initialize <code>\@addtofilelist</code> to <code>\@gobble</code>	4
1995-11-01 ltfinal.dtx v1.0x	General: (DPC) Switch meaning of <code>\@addtofilelist</code> for cfg files	1339
1995-11-01 ltffsbas.dtx v3.0m	<code>\DeclareFontShape@</code> : (DPC) Test for <code>\relax</code> not <code>\undefined</code> , internal/1933	572
1995-11-01 ltffsini.dtx v3.0g	General: (DPC) Switch meaning of <code>\@addtofilelist</code> for cfg files	764
1995-11-02 ltffsbas.dtx v3.0n	<code>\wrong@fontshape</code> : (DPC) Remove extra space with <code>\string</code> for latex/1676	593

1995-11-02 ltoutenc.dtx v1.7n	1995-11-28 ltfloat.dtx v1.1n
General: Changed internal name <code>\a</code> to <code>\@tabacckludge</code> to protect against redefinition by malicious users.	General: documentation fixes 1026
1995-11-07 ltlists.dtx v1.0k	1995-11-28 ltfsstrc.dtx v3.0g
<code>\@doendpe</code> : Enclosed <code>\setbox0</code> assignment by a group so that it leaves the contents of box 0 intact.	General: documentation fixes 671
1995-11-07 ltoutenc.dtx v1.7o	1995-11-28 ltoutenc.dtx v1.7r
General: Added <code>\leavevmode</code> at start of <code>\c</code> , otherwise the output routine might be invoked within the macro.	General: Added math mode checks to text commands. 513
Changed <code>\char32</code> to <code>\@xxxii</code> (two tokens less).	doc fixes 509
Replaced octal number 27 by decimal number 23 to protect against the quote character being active.	Renamed <code>\@changed@x@err</code> to <code>\TextSymbolUnavailable</code> 513
Replaced some 0's by <code>\z@</code> (faster). 527	1995-11-29 ltoutenc.dtx v1.7t
1995-11-10 ltoutput.dtx v1.1s	General: Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> and <code>\textless</code> 529
<code>\@shipoutsetup</code> : Command removed 1254	Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textregistered</code> and <code>\texttrademark</code> 523
<code>\@writesetup</code> : Command removed . 1254	Added <code>\textbackslash</code> and <code>\textbar</code> 523, 534
In-lined 1254	Added <code>\textless</code> and <code>\textgreater</code> 523, 534
1995-11-14 ltclass.dtx v1.0t	1995-12-01 ltoutenc.dtx v1.7u
<code>\@unprocessedoptions</code> : Allow empty option 1130	General: Made <code>\SS</code> a Default, rather than having the default point to the OT1 definition. 523
<code>\@loadwithoptions</code> : macro added . 1117	1995-12-04 ltspc.dtx v1.2k
<code>\LoadClassWithOptions</code> : macro added 1117	<code>\nobreakspace</code> : Macro added 480
<code>\RequirePackageWithOptions</code> : macro added 1118	1995-12-04 ltspc.dtx v1.2l
1995-11-17 ltfsbas.dtx v3.0m	<code>\xobeysp</code> : braces added to definition of tilde 480
<code>\@wrong@font@char</code> : (DPC) Macro added. latex/1676 594	1995-12-04 preload.dtx v2.4e
<code>\define@newfont</code> : Redefine <code>\typeout</code> latex/1676 587	General: Ulrik Vieth. added 12pt OMS and OML preloads /1989 . . 789
<code>\wrong@fontshape</code> : Support <code>\@wrong@font@char</code> latex/1676 . . 593	1995-12-05 ltdefs.dtx 1.2w
1995-11-17 ltoutenc.dtx v1.7p	<code>\@unexpandable@protect</code> : Removed <code>\unexpandable@noexpand</code> as never used. internal/1733 92
<code>\UseTextSymbol</code> : Support <code>\@wrong@font@char</code> latex/1676 . . 519	1995-12-05 ltfiles.dtx v1.1c
1995-11-18 ltoutenc.dtx v1.7q	<code>\document</code> : <code>\ignorespaces</code> added for latex/1933 489
<code>\UseTextSymbol</code> : Modify message slightly 519	1995-12-05 ltfloat.dtx v1.1n
1995-11-21 fontdef.dtx v2.2n	<code>\@textsuperscript</code> : Use <code>\ensuremath</code> for latex/1984. 1044
General: Incorporate changed figures, as in plain.tex 783	1995-12-05 ltoutenc.dtx v1.7v
1995-11-27 ltfsbas.dtx v3.0n	<code>\@inmathwarn</code> : Changed <code>\TextSymbolUnavailable</code> text . . 515
<code>\nfss@catcodes</code> : Reset hash, for definitions in fd files 589	1995-12-06 ltfsbas.dtx v3.00
	<code>\nfss@catcodes</code> : Reset hat, for typeouts etc in fd files 589
	1995-12-07 ltbibl.dtx v1.1l
	<code>\@citex</code> : Restored name of <code>\G@refundefinedtrue</code> 1055

1995-12-07 lfloat.dtx v1.1m	1996-05-17 fontdef.dtx v2.2o
\@textsuperscript: Move \m@th out	General: \@@sqrt removed, at
of the \ensuremath for	last 766, 782
latex/1984. 1044	1996-05-17 ltfiles.dtx v1.1f
1995-12-07 ltxref.dtx v1.1i	\nofiles: added \write to
\@setref: Switch for reundefined	\protected@write for latex/2146 492
restored 842	1996-05-18 ltoutenc.dtx v1.7x
\G@refundefinedtrue: Renamed	General: Produce error if encoding not
(back) from \G@refundefined .. 841	found. pr/2054 553
1995-12-11 ltoutenc.dtx v1.7w	1996-05-21 ltoutenc.dtx v1.7y
General: Modified \copyright 523	General: Corrected error message
1995-12-13 ltdefs.dtx 1.2x	(CAR) 553
\:- Documentation changed. 112	1996-05-21 ltsect.dtx v1.0s
1996-01-10 ltfiles.dtx v1.1d	\@sect: (DPC) Added extra braces for
\@iffileonpath: Change argument	internal/2148 1016
handling to not require doubled	(DPC) Moved brace to allow
hash. latex/2024 502	commands like \MakeUppercase in
1996-01-20 ltidxglo.dtx v1.1e	6th argument. Changed \par to
\makeglossary: Make no-op after use	\endgraf to allow non-long
pr/2048 1051	commands. internal/2148 1016
\makeindex: Make no-op after use	\@ssect: (DPC) Added extra braces
pr/2048 1051	for internal/2148 1019
1996-01-20 ltspace.dtx v1.2m	(DPC) Moved brace to allow
\vspacer: Made robust 478	commands like \MakeUppercase in
1996-03-25 ltmath.dtx v1.1a	4th argument. Changed \par to
\@ensuredmath: Macro added for	\endgraf to allow non-long
amslatex/2104 903	commands. internal/2148 1019
\ensuremath: Reimplement for	1996-05-23 ltoutenc.dtx v1.7z
amslatex/2104 903	\@strip@args: \expandafter added to
1996-04-18 ltpage.dtx v1.0i	match other changes for
General: Improve documentation . 1089	latex/2133 518
1996-04-22 ltmiscen.dtx v1.1c	\add@accent: macro added.
General: Improve Documentation .. 864	latex/2133 516
1996-04-22 ltspace.dtx v1.2n	\DeclareTextAccent: Reimplemented
General: Documentation	using \add@accent to save space
Improvements 463	latex/2133 516
1996-04-22 lttab.dtx v1.1g	\DeclareTextCompositeCommand:
\@tabclassz: (DPC) Extra \hskip	Modified to cope with new
keeps tabcolsep in empty columns	\add@accent command: required
internal/2122 968	removal of check for one
1996-04-23 ltcounts.dtx v1.1d	argument-command 517
General: Documentation	1996-05-24 ltoutput.dtx v1.1t
improvements 556	\@specialoutput: Check that
1996-04-24 ltfiles.dtx v1.1e	\@colroom is less than \vsize,
\document: (DPC) Reset	indicating that a float has been
\AtBeginDocument eg for	added 1235
latex/1297 488	Cut-off point changed to
1996-05-08 ltfstsrc.dtx v3.0h	1.5\baselineskip 1235
\math@egroup: Use \bgroup instead of	\@topnewpage: Cut-off point changed
\begingroup to match a kernel	to 2.5\baselineskip 1231
change made in 1994!! 683	1996-05-25 ltoutput.dtx v1.1u
1996-05-09 ltfntcmd.dtx v3.3t	\@specialoutput: Correct the above
\check@icr: Default definitions	check 1235
added 794	

1996-06-03 ltmiscen.dtx v1.1d	1996-07-26 ltfloat.dtx v1.1n
\@verbatim: Exchanged the following two code lines so that \dospecials cannot reset the category code of characters handled by \@noligs. 882	\@endfloatbox: remove unnecessary \global before \@minipage... 1035
General: Move setting of verbatim font and \@noligs. 864	\@savemarbox: remove unnecessary \global before \@minipage... 1039
\verb: Put setting of verbatim font after \dospecials so that \dospecials cannot reset the category code of characters handled by \@noligs. 887	\@setminipage: remove unnecessary \global before \@minipage... 1033
1996-06-10 ltboxes.dtx v1.0y	\@setnobreak: remove unnecessary \global before \@nobreak... 1033
\@parboxto: (DPC) Changed \endgraf to \@@par 936	1996-07-26 ltfsbas.dtx v3.0p
1996-06-10 ltsect.dtx v1.0t	\@DeclareMathSizes: use faster \if test 580
\@sect: (DPC) Changed \endgraf to \@@par 1016	\@nfss@catcodes: omit \relax as not needed 589
\@ssect: (DPC) Changed \endgraf to \@@par 1019	1996-07-26 ltfsdcl.dtx v3.0e
1996-06-13 ltdirchk.dtx v1.0r	\@init@restore@version: Removed \ifrestore@version switch and replaced by \@init@restore@version 706
General: documentation improvements mainly from internal/2174 1	1996-07-26 ltfsstrc.dtx v3.0i
1996-06-14 lttab.dtx v1.1h	\@init@restore@glb@settings: macro added replacing \if@inmath switch 682
\@tabclassz: (DPC) Change both\z@skip to 1sp for latex/2160 968	1996-07-26 ltlists.dtx v1.0l
1996-06-22 ltspace.dtx v1.2o	\@item: Remove unnecessary \global before \@minipage... 920
General: Documentation of problems added 463	Remove unnecessary \global before \@nobreak... 921
1996-07-10 ltfinal.dtx v1.0y	1996-07-26 ltmath.dtx v1.1b
\toks: Free up memory from scratch registers /2213 1351	General: Removed \global before \@ignoretrue in various places. 889
1996-07-19 ltoutenc.dtx v1.8a	1996-07-26 ltmiscen.dtx v1.1e
\@strip@args: Use char 0 not @ as carrier for \lowercase /2197 519	\@ignorefalse: put \global into definition 865
1996-07-26 ltboxes.dtx v1.0z	\@begin: remove \global before \@ignore... 873
\if@minipage: put \global into definition 939	\@end: remove \global before \@ignore... 875
1996-07-26 ltclass.dtx v1.0u	\@ignorespacesafterend: user level macro added 865
\@classoptionslist: made only preamble 1099	1996-07-26 ltoutput.dtx v1.1v
\@unusedoptionlist: made only preamble 1099	\@testfp: remove \global before \@test... 1296
1996-07-26 ltdefs.dtx v1.2y	\@xtryfc: remove \global before \@test... 1264
\@reargdef: third arg picked up by \@yargdef 87	\@ztryfc: remove \global before \@test... 1265
\@renew@command: use \noexpand instead of \string 87	General: put \global into definition remove \global before \@test... 1224
use \relax in place of empty arg 87	\@clearpage: add number of missing percents 1227
\@renew@environment: use \relax in place of empty arg 88	1996-07-26 ltplain.dtx v1.1t
	\sh@ft: replace \dimen\z@ by \dimen@ 30

1996-07-26 ltsect.dtx v1.0u	tests	1228
\@starttoc: removed \global before	1996-10-04 ltclass.dtx v1.0v	
\nobreak	\RequirePackageWithOptions: Reset	
\@xsect: Removed \global before	\@unprocessedoptions for	
\nobreak	/2269	1118
1996-07-26 ltspace.dtx v1.2p	1996-10-05 ltfiles.dtx v1.1h	
\if@nobreak: put \global inside	\clubpenalty: Added setting its	
definition	value	487
1996-07-27 ltfssbas.dtx v3.0q	1996-10-08 ltfntcmd.dtx v3.3u	
General: \if@inmath switch removed	\DeclareTextFontCommand: Removed	
1996-07-27 ltspace.dtx v1.2q	\check@icr when in vmode since	
General: Further documentation of	it causes various errors (see	
problems	pr/2157)	792
1996-07-27 ltspace.dtx v1.2r	1996-10-21 lttab.dtx v1.1i	
General: Correct documentation of	\@array: Use \set@typeset@protect	961
problems	General: Moved the code associated	
1996-08-02 ltfloat.dtx v1.1o	with \mkpream into the group	
\@xympar: Remove \global before	provided by the box, for	
\@ignore	robustness (latex/2183)	960
1996-08-02 ltsect.dtx v1.0v	\multicolumn: Make \multicolumn	
\@afterheading: Removed \global	long (latex/2180)	963
before \nobreak	\tabbing: Moved the \indent so that	
1996-08-02 ltspace.dtx v1.2s	the \everypar can remove it when	
\@Esphack: Remove \global before	necessary; this is needed because	
\@ignore	the code for items in lists has	
1996-08-25 ltfssbas.dtx v3.0r	changed (see pr/22111)	954
\nfss@catcodes: Reset the acute,	1996-10-23 ltlists.dtx v1.0m	
grave and double quote chars as	\@item: \@nobreak . . . moved into the	
well	\everypar and not executed	
1996-09-21 ltoutput.dtx v1.1w	unconditionally, see above	921
\@writesetup: Added	\kern . . . changed to \setbox . . .	920
\@parboxrestore and made	Added setting of \clubpenalty and	
consequent deletions: wait for the	set \nobreakfalse only when	
howls of protest	necessary	921
1996-09-25 ltdirchk.dtx v1.0t	1996-10-23 ltsect.dtx v1.0x	
General: Move ltxcheck to separate file	\@xsect: Replaced \hskip . . . with	
1996-09-28 ltmiscen.dtx v1.1f	\setbox . . . as used in	
\@xobeysp: Moved to ltspace.dtx . .	\@afterheading	1018
1996-09-28 ltspace.dtx v1.2t	1996-10-24 ltboxes.dtx v1.1a	
\@xobeysp: Moved from ltmiscen.dtx	\@arrayparboxrestore: Added local	
and redefined to use	settings of flags: dangerous!	938
\nobreakspace	\@iiminipage: Use it or lose it	
1996-09-29 ltfiles.dtx v1.1g	(@setminpage): Frank will want to	
\document: Added disabling of	lose it	940
\nodocument	1996-10-24 ltfloat.dtx v1.1p	
1996-09-29 ltoutput.dtx v1.1x	\@floatboxreset: Added local	
\newpage: Checks for noskipsec and	settings of flags: dangerous!	1033
inlabel added	\@marginparreset: Added local	
1996-09-29 ltsect.dtx 1.0w	settings of flags: dangerous!	1039
\@noskipsectrue: Added	\@xfloat: Added \nodocument to	
documentation	trap floats in the preamble . . .	1030
1996-09-30 ltoutput.dtx v1.1y	1996-10-24 ltoutput.dtx v1.1z	
\newpage: Checks for noskipsec and	\@addtocurcol: Added \nobreak, etc	
inlabel removed pending further	as appropriate	1270, 1276

<code>\@specialoutput</code> : Added <code>\nobreak</code> as appropriate	1237	1996-11-04 <code>ltlists.dtx</code> v1.0q	
<code>\@topnewpage</code> : Added <code>\@nodocument</code> to trap <code>\twocolumn</code> in the preamble	1230	<code>\@trivlist</code> : Moved check for missing item: only checked when not inlabel flag is false	915
<code>\newpage</code> : Better checks for noskipsec and inlabel added, plus nobreak	1228	1996-11-05 <code>ltfiles.dtx</code> v1.1i	
1996-10-25 <code>ltlists.dtx</code> v1.0n		<code>\nofiles</code> : Standard <code>\if@nobreak</code> test added	492
<code>\endtrivlist</code> : Change <code>\indent</code> to <code>\leavevmode</code>	916	1996-11-09 <code>ltmath.dtx</code> v1.1c	
Reset flags explicitly	916	<code>\@ensuredmath</code> : Made long, as it was before. /2104	903
1996-10-25 <code>ltoutput.dtx</code> v1.2a		1996-11-18 <code>ltfssbas.dtx</code> v3.0s	
<code>\newpage</code> : Reset all flags explicitly	1228	<code>\define@newfont</code> : (DPC) lowercase fd file names. internal/1044	588
1996-10-26 <code>ltlists.dtx</code> v1.0o		1996-11-18 <code>ltoutenc.dtx</code> v1.8d	
<code>\endtrivlist</code> : Correct typo	916	General: (DPC) lowercase external file names. internal/1044	553
1996-10-27 <code>ltoutenc.dtx</code> v1.8c		1996-11-20 <code>fontdef.dtx</code> v2.2p	
<code>\@strip@args</code> : Removed macro	517	General: lowercase fd and enc.def file names /1044	766
General: Added <code>\r A</code>	527	1996-11-20 <code>ltvers.dtx</code> v1.0f	
Added		General: Check for old format modified /2319	35
<code>\textasteriskcentered</code>	523, 534	1996-11-23 <code>ltoutenc.dtx</code> v1.8e	
Corrected syntax descriptions	510	General: Corrected description	510
Removed <code>\aa</code> and <code>\AA</code>	522, 527, 529	Extended description	511
1996-10-28 <code>ltplain.dtx</code> v1.1u		1996-11-28 <code>ltvers.dtx</code> v1.0g	
General: (CAR) More doc changes	14	General: Check for old format modified /2319	35
<code>\dotfill</code> : Removed math mode	30	1996-12-06 <code>ltdirchk.dtx</code> v1.0u	
1996-10-29 <code>ltplain.dtx</code> v1.1v		<code>\IfFileExists</code> : *** removed from various messages for GNU Make. internal/2338	10
<code>\dotfill</code> : Got arithmetic correct (CAR)	30	1996-12-06 <code>ltfloat.dtx</code> v1.1r	
1996-10-29 <code>ltspace.dtx</code> v1.2u		<code>\@caption</code> : Call <code>\@setminpage</code> if needed. latex/2318	1029
<code>\@gnewline</code> : Added macro	469	1996-12-06 <code>ltfssini.dtx</code> v3.0h	
<code>\@no@lnbk</code> : Macro replaces <code>\@lnbk</code> and <code>\@nolnkb</code>	467	General: (DPC) Remove *** from messages internal/2338	764
<code>\:</code> : Corrected and rationalized code	467	1996-12-17 <code>ltdefns.dtx</code> v1.0w	
<code>\nolinebreak</code> : Reimplemented both using <code>\@no@lnbk</code>	466	<code>\g@addto@macro</code> : Use <code>\begingroup</code> to save making a mathord	114
1996-10-31 <code>ltfinal.dtx</code> v1.0z		1996-12-20 <code>ltsect.dtx</code> v1.0z	
General: Added extra <code>\lcode</code> , hoping it does no harm in T1 (pr/1969)	1338, 1347	<code>\@dottedtocline</code> : Added <code>\nobreak</code> for latex/2343	1024
1996-10-31 <code>ltlists.dtx</code> v1.0p		1997-01-08 <code>fontdef.dtx</code> v2.2q	
<code>\@trivlist</code> : Added check for missing item in outer list	915	General: Use <code>\DeclareMathDelimiter</code> to set delimiter codes	776
1996-10-31 <code>ltsect.dtx</code> v1.0y		<code>\mathparagraph</code> : Define using <code>\DeclareMathSymbol</code>	784
General: Corrected and tidied documentation; removed long lines	1012	1997-01-08 <code>ltfiles.dtx</code> v1.1j	
1996-11-03 <code>ltplain.dtx</code> v1.1w		<code>\@include</code> : reset <code>\deadcycles</code> latex/2365	496
<code>\dotfill</code> : Saved tokens by using <code>\hb@xt@</code>	30		
1996-11-04 <code>lterror.dtx</code> v1.2m			
<code>\@nodocument</code> : Always define <code>\@nodocument</code> in kernel, so that it can be cleared by <code>\document.</code>	434		

1997-01-08 ltmath.dtx v1.1d	1997-05-29 ltlogos.dtx v1.1f
<code>\root</code> : (DPC) Remove spurious space tokens from plain <code>T_EX</code> definition /2359 891	<code>\LaTeXe</code> : Added <code>\m@th</code> so that the $\text{\LaTeX 2\epsilon}$ logo works with non-zero values of <code>\mathsurround</code> 484
1997-02-05 ltdefns.dtx v1.0x	1997-06-16 ltdirchk.dtx v1.0v
<code>\g@addto@macro</code> : missing percent /2402 114	General: documentation improvements mainly from internal/2520 1
1997-02-21 ltlists.dtx v1.0r	1997-06-16 ltfloat.dtx v1.1s
<code>\@item</code> : <code>\ifvoid</code> check added for <code>\noindent</code> . latex/2414 920	General: documentation fixes 1026
1997-03-21 ltcounts.dtx v1.1e	1997-06-16 ltfntcmd.dtx v3.3v
<code>\fnsymbol</code> : Use <code>\mathsection</code> and <code>\mathparagraph</code> . latex/2445 ... 566	General: Fix typo in documentation. 790
1997-04-14 ltfiles.dtx v1.1k	1997-08-05 ltoutenc.dtx v1.9e
<code>\document</code> : Set the document space factor defaults. latex/2404 488	General: Corrected order of arguments in <code>\UseTextSymbol</code> example. ... 510
<code>\normalsfcodes</code> : Macro added (from patch file) latex/2404 492	1997-08-29 ltoutenc.dtx v1.9f
1997-04-14 ltoutput.dtx v1.2b	General: Added OT4 encoding, provided by Marcin Woliński. .. 509
<code>\@writeseup</code> : Call <code>\normalsfcodes</code> (from patch file) latex/2404 ... 1256	1997-09-09 ltdefns.dtx v1.2z
Move <code>\label</code> and <code>\index</code> (from patch file) 1257	<code>\provide@command</code> : Use <code>\begingroup</code> to avoid generating math ords if used in math mode. pr/2573 89
1997-04-24 ltbibl.dtx v1.1m	1997-09-15 ltpictur.dtx v1.1g
<code>\@citex</code> : <code>\@empty</code> to avoid primitive error on empty cite keys. latex/2432 1055	<code>\@getcirc</code> : Warn if lines become invisible pr/2524 995
1997-04-30 ltoutenc.dtx v1.9a	<code>\@picture@warn</code> : Macro added pr/2524 996
General: Changed <code>\textsc</code> to <code>\scshape</code> 524	<code>\@sline</code> : Warn if lines become invisible pr/2524 985
Introduced <code>\textcopyright</code> and modified <code>\copyright</code> 523	1997-10-06 ltcounts.dtx v1.1f
Introduced <code>\textcopyright</code> and modify <code>\copyright</code> 524	<code>\@Roman</code> : Change <code>\@Roman</code> to be fully expandable, so that the result is written properly to files. 566
Modified <code>\textunderscore</code> , removing <code>\mathunderscore</code> 523	<code>\@slowromancap</code> : Macro added. ... 566
Modified <code>\underscore</code> , removing <code>\mathunderscore</code> 524	1997-10-08 ltlogos.dtx v1.1h
1997-04-30 ltoutenc.dtx v1.9b	<code>\LaTeX</code> : Simplify macro (force loading of suitable math fonts once). ... 484
General: Added <code>\leavevmode</code> to <code>\textunderscore</code> 523	1997-10-10 ltclass.dtx v1.0y
1997-05-04 ltoutenc.dtx v1.9c	<code>\endfilecontents</code> : <code>\@currenvir</code> in banner 1133
General: Added ‘hex index tabs’ ... 531	<code>\reserved@c</code> not <code>\verbatim@out</code> to save a csname 1132
Added TS1 encoding v2.2.beta .. 537	Check for text before or after <code>\end</code> environment. latex/2636 1134
1997-05-07 ltoutenc.dtx v1.9d	Use <code>\@gobbletwo</code> 1133
General: Added <code>\leavevmode</code> to <code>\textcompwordmark</code> 523	1997-10-17 ltfntcmd.dtx v3.3w
1997-05-07 ltspace.dtx v1.2v	<code>\check@nocorr@</code> : Check for vertical mode moved here, from <code>\DeclareTextFontCommand</code> (see PR/2646). 794
<code>\newline</code> : Made completely robust. 468	<code>\DeclareTextFontCommand</code> : Reinstalled <code>\check@icr</code> as check is now done in <code>\check@nocorr@</code> (see PR/2646). 792
1997-05-29 ltfstrc.dtx v3.0j	
General: Replaced <code>\</code> by <code>\MessageBreak</code> , as suggested by Donald Arseneau. 673	

1997-10-20 ltfinal.dtx v1.1a	Added section.	553
\@uclclist: Removed \aa and \AA from \@uclclist as these are macros.	Added textcomp.sty.	509
1997-10-21 ltdefs.dtx v1.2z1	As in OT1, Added \leavevmode at start of \c, otherwise the output routine might be invoked within the macro.	528
\renewcommand: Use \beginngroup/\endgroup rather than braces for grouping, to avoid generating empty math atom. . . .	Changed to decimal codes in \ooalign.	539
1997-10-21 ltssbas.dtx v3.0t	Changed to decimal codes.	534
\define@newfont: Move \makeatletter to \nfss@catcodes.	Documentation changes and additions.	509
\nfss@catcodes: Moved \makeatletter from \try@load@font@shape.	Example corrected, braces removed.	509
1997-11-09 ltoutput.dtx v1.2c	Removed default settings, see next section.	537
\@specialoutput: Remove incorrect code: only one \@emptycol is needed here	1997-12-19 ltoutenc.dtx v1.9i General: Documentation corrections. . . .	509
\@topnewpage: Documentation of vsize check enhanced	1997-12-20 fontdef.dtx v2.2s General: Added documentation	768
1997-11-13 ltssdcl.dtx v3.0f	1997-12-31 ltoutenc.dtx v1.9k General: Further correction	510
\DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661	1998-01-12 ltoutenc.dtx v1.9k General: Added \ProvidesPackage for textcomp.sty	509
\stepcounter: (DPC) Remove as never used. (Re)defined in ltxcounts	Adding missing braces and \ushape.	539
1997-11-19 ltfloat.dtx v1.1t	1998-01-16 ltoutenc.dtx v1.9m General: fixed decimal codes. latex/2734	534
\@footnotetext: Missing percent, again	1998-03-04 ltdefs.dtx v1.2z2 \@xargdef: Unnecessary \expandafter removed: pr/2758 . . .	85
1997-11-19 ltoutput.dtx v1.2d	1998-03-05 ltoutenc.dtx v1.9n General: Added masc/fem ords as in pr/2579	524
\@vtryfc: Reindent code, to be understandable(DPC).	1998-03-20 ltdefs.dtx v1.2z3 \@thirdofthree: Macro added	90
1997-11-20 ltssdcl.dtx v3.0g	1998-03-20 ltoutenc.dtx v1.9o General: Documentation added about order of decls	512
\document@select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list)	Documentation added for pr/2783	511
\select@group: (DPC) inline use of \stepcounter (faster, and saves a csname per math version as no reset list)	\UndeclareTextCommand: Macro added for pr/2783	520
1997-11-23 ltoutenc.dtx v1.9g	1998-03-20 lttextcomp.dtx v1.9o General: Added various \UndeclareTextCommand declarations for pr/2783	827
General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textpertenmill and \textfraction. /2673	Load decls after defaults for speed. . . .	827
1997-12-17 ltoutenc.dtx v1.9h	1998-03-21 ltclass.dtx v1.0z General: Added to documentation of filecontents	1094
General: Added \textperthousand and \textpertenthousand	1998-03-21 ltclass.dtx v1.1a \@providesfile: Allow &. Internal/2702	1110
Added code for textcomp.sty. . . .		553

General: Correct to new onlypreamble command list	1147	1998-07-06 lttab.dtx v1.1l	General: Small correction to documentation	947
1998-03-25 ltfssbas.dtx v3.0u		1998-08-17 ltboxes.dtx v1.1e	General: (RmS) Minor Documentation fixes.	925
\showhyphens: Suppress unnecessary error when used in preamble . . .	598	1998-08-17 ltclass.dtx v1.1c	General: (RmS) Minor documentation fixes.	1094
1998-04-11 fontdef.dtx v2.2t		1998-08-17 ltdirchk.dtx v1.0w	General: (RmS) Documentation improvements.	1
General: Added \mathring accent (pr2785)	782	1998-08-17 ltfntcmd.dtx v3.3x	General: (RmS) Minor documentation fixes.	790
1998-04-15 fontdef.dtx v2.2u		1998-08-17 ltfssbas.dtx v3.0v	General: (RmS) Documentation fixes.	571
General: Use new syntax for \DeclareMathDelimiter	776	1998-08-17 ltfssdcl.dtx v3.0i	General: (RmS) Corrected minor glitches in changes entries.	702
1998-04-15 ltfssdcl.dtx v3.0h		1998-08-17 ltfssini.dtx v3.0i	General: (RmS) Minor documentation fixes.	735
\@xxDeclareMathDelimiter: Macro added (pr/2662)	727	1998-08-17 ltlogos.dtx v1.1i	General: (RmS) Minor documentation fixes.	484
1998-04-17 fontdef.dtx v2.2v		1998-08-17 ltmath.dtx v1.1c	General: (RmS) Minor documentation fixes.	889
General: Reinsert symbol defs for < and > chars.	776	1998-08-17 ltmiscen.dtx v1.1g	General: (RmS) Minor documentation fixes.	864
1998-04-18 fontdef.dtx v2.2w		1998-08-17 ltspc.dtx v1.2w	General: Documentation fixes.	463
General: Reinsert symbol def for / char.	776	1998-08-17 preload.dtx v2.1g	General: (RmS) Minor documentation fixes.	787
1998-05-07 ltclass.dtx v1.1b		1998-09-19 ltoutenc.dtx v1.9r	\a: Added \string (pr/2878)	522
\@onefilewithoptions@clashchk: Modify help message for latex/2805	1125	1998-11-13 lttab.dtx v1.1m	\@array: Check for hmode to see if something went wrong during parsing (pr/2884)	961
1998-05-18 lttab.dtx v1.1j		1999-01-05 fontdef.dtx v2.2x	General: Need special protection for character > in \changes entry. . .	766
\@endpbox: Use \setlength to set \hspace, so that the changes in the calc package apply here.	972	1999-01-06 ltfssbas.dtx v3.0w	\DeclareFontEncoding: Added \LastDeclaredEncoding to support cyrillic integration (pr/2988)	575
\tabular*: Use \setlength, so that calc extensions apply.	960		\LastDeclaredEncoding: Added \LastDeclaredEncoding to	
1998-05-20 ltfinal.dtx v1.1b				
General: Set up lccodes before loading hyphenation files: pr/2639	1337			
Set up uc/lccodes after loading hyphenation files: pr/2639	1346			
1998-05-28 lterror.dtx v1.2n				
\@notdefinable: Added message re 'end...' pr/1555	434			
1998-06-04 ltboxes.dtx v1.1c				
\@rule: Support calc-expressions . .	942			
1998-06-12 ltoutenc.dtx v1.9p				
General: Corrected 130 and 131, see pr/2834	539			
Renamed \textmacron pr/2840 . .	540			
1998-06-12 ltoutenc.dtx v1.9q				
\add@accent: Explicitly set \spacefactor after \accent (pr/2877)	516			
1998-06-12 lttextcomp.dtx v1.9p				
General: Renamed \textmacron pr/2840	823			
1998-06-18 lttab.dtx v1.1k				
General: Small addition to documentation	947			

support cyrillic integration (pr/2988)	575	1999-04-29 ltdefs.dtx v1.3f <code>\@yargd@f</code> : Full expansion and conversion needed for digit in new version, see pr/3013	85
1999-01-06 ltoutenc.dtx v1.9r <code>\@strip@args</code> : New impl for latex/2930	518	New macro added	85
General: Minor documentation fix. . .	539	1999-06-10 ltoutenc.dtx v1.9u General: Ensure that we also forget old options (pr/2888)	555
1999-01-07 ltdefs.dtx v1.3a <code>\@ifnextchar</code> : made long	110	1999-06-12 ltoutenc.dtx v1.9v General: Extend <code>\@uclclist</code> only once	554
<code>\@newenvb</code> : made long and brace optional arg. latex/2896	88	1999-10-09 ltmath.dtx v1.1e <code>\active@math@prime</code> : Macro added, see PR 3104.	896
<code>\@testopt</code> : made long and brace optional arg. latex/2896	85	<code>\prime@s</code> : Introduce <code>\active@math@prime</code>	896
1999-01-07 ltdefs.dtx v1.3b <code>\@ifnextchar</code> : extra <code>\long</code> . latex/2902	110	1999-10-09 ltoutput.dtx 1.2f <code>\@activechar@info</code> : Reset definition of active prime character (used in math mode)	1253
1999-01-07 ltoutenc.dtx v1.9r General: Hackery to allow using fontenc several times	555	1999-10-28 ltoutenc.dtx v1.9w <code>\add@accent</code> : Give <code>\accent@spacefactor</code> a default definition (pr/3084)	516
Hackery to temp support cyrillic uc/lc	553	1999-12-08 ltoutenc.dtx v1.9x General: Changed <code>\CYRRHOOK</code> and <code>\cyrrhook</code> to <code>\CYRRHK</code> and <code>\cyrrhk</code> as name changed in the cyrillic bundle for naming consistency with other “hook” glyphs.	553
1999-01-13 ltoutenc.dtx v1.9s <code>\@strip@args</code> : Simplified solution for latex/2930	518	2000-01-07 ltmiscen.dtx v1.1h <code>\@verbatim</code> : Disable hyphenation even if the font allows it.	882
1999-01-18 ltdefs.dtx v1.3c <code>\@yargd@f</code> : New implementation DPC /2942	85	2000-01-15 ltpictur.dtx v1.1i <code>\@upvector</code> : Removed space at end-of-line, CAR	987
1999-02-09 ltdefs.dtx v1.3d <code>\@yargd@f</code> : catch bad argument forms by re-inserting #3	85	2000-01-30 ltfntcmd.dtx v3.3y <code>\DeclareTextFontCommand</code> : Use <code>\hmode@bgroup</code> now (pr/3160) . .	792
1999-02-12 lttextcomp.dtx v3.0j <code>\legacyoldstylenums</code> : Use <code>\rmdefault</code> instead of <code>cmm</code> (pr/2954)	799	2000-01-30 ltoutenc.dtx v1.9y General: Use <code>\hmode@bgroup</code> where applicable (pr/3160)	526–528, 534–537, 539
1999-02-24 ltoutenc.dtx v1.9t General: Corrected hackery cyrillic uc/lc list	553	<code>\add@accent</code> : Use <code>\hmode@bgroup</code> where applicable (pr/3160)	516
1999-03-01 ltdefs.dtx v1.3e <code>\@ifnextchar</code> : remove extra <code>\long</code> . internal/2967	110	<code>\hmode@bgroup</code> : Macro added	516
1999-04-15 ltpictur.dtx v1.1h <code>\@getlarrow</code> : Replaced octal number, CAR	986	2000-01-30 ltoutenc.dtx v1.9z <code>\@use@text@encoding</code> : Macro reimplemented (pr/3160)	519
<code>\@upvector</code> : Replaced octal number, CAR	987	<code>\add@accent</code> : Macro reimplemented (pr/3160)	516
General: Replaced octal number, CAR	986, 987	<code>\hmode@start@before@group</code> : Macro added (pr/3160)	520
Replaced octal numbers, CAR . .	974		
1999-04-19 ltfloat.dtx v1.1u <code>\caption</code> : Made caption an error outside a float: latex/2815	1029		
1999-04-27 ltboxes.dtx v1.1f <code>\@parboxto</code> : (CAR) Changed <code>\@empty</code> to <code>\relax</code> as flag for natural width: pr/2975	936		

2000-05-19 ltmiscen.dtx v1.1i	2001-02-16 ltxref.dtx v1.1k
\enddocument: Reset \AtEndDocument for latex/3060 865	\@newl@bel: Added an extra grouplevel (PR3250), jlb 843
2000-05-26 ltpage.dtx v1.0j	2001-05-25 ltclass.dtx v1.1d
\@markright: Reimplementation to fix expansion error (pr/3203). 1092	\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334) 1110
\leftmark: Use \@empty instead of brace group (pr/3203). 1092	2001-05-25 ltdirchk.dtx v1.0x
\markright: Reimplementation to fix expansion error (pr/3203). 1090	General: Explicitly set catcode of \endlinechar to 10 (pr/3334) 4
\rightmark: Use \@empty instead of brace group (pr/3203). 1092	2001-05-28 ltoutenc.dtx v1.93
2000-06-02 ltpage.dtx v1.0k	General: Added composites for compatibility with T1, pr/3295 527
\@markright: Small adjustment to give slightly less expansion, CAR 1092	Changed the effect of \.i, pr/3295 531
\markright: Small adjustment to give slightly less expansion, CAR 1090	2001-06-02 fontdef.dtx v2.2y
Tidied 1.0j reimplementation, CAR 1090	General: Provide default cfg files (pr/3264) 785
2000-07-11 ltmiscen.dtx v1.1j	2001-06-04 fontdef.dtx v2.2z
\@enddocument@kernel@warnings: Fix typo in warning 867	General: Guard against math active equal and pipe sign in \models (pr/3333) 781
2000-07-12 ltoutput.dtx 1.2g	Guard against math active equal sign in \Relbar (pr/3333) 781
General: Ensure that rule is in \normalcolor 1303	2001-06-04 ltclass.dtx v1.1e
2000-07-19 ltoutput.dtx v1.2h	\@providesfile: But only if it is a char (pr/3334) 1110
\@writsetup: Reset and restore \@if@newlist for internal/3231 1255	2001-06-04 ltdirchk.dtx v1.0y
2000-08-23 ltfinal.dtx v1.1c	General: But only if it is a char (pr/3334) 4
General: Fix typo in warning 1339	2001-06-04 ltpictur.dtx v1.1j
2000-08-30 ltoutenc.dtx v1.9l	\@sline: Don't warn for exactly zero pr/3318 985
\@use@text@encoding: Rearranged but no change to final code, CAR (pr/3160) 519	2001-06-04 ltvers.dtx v1.0i
\add@accent: Rearranged but no change to final code, CAR (pr/3160) 516	General: Check for old format disabled 35
2000-09-01 ltfinal.dtx v1.1d	2001-06-05 ltoutenc.dtx v1.94
\@errhelp: Set error help empty at very end (pr/449 done correctly). 1351	General: Text composite Commands need kludges for ‘,’ – see tlb1903.lvt 527
2000-09-24 ltfloat.dtx v1.2b	2001-08-26 ltclass.dtx v1.1f
\end@dblfloat: FMI: use output routine to defer float 1034	\@providesfile: Readded setting of space char (pr/3353) 1110
2000-09-24 ltoutput.dtx v1.2b	2002-02-24 ltplain.dtx v1.1x
\@docclearpage: FMI: ensure \docclearpage is called again until all floats are output. 1239	\loggingall: Macro added 31
2000-09-24 ltoutput.dtx v1.2n	\loggingoutput: Macro added 31
\@addtocurcol: FMI: test for wide float was in wrong place 1269	\showoutput: Use newly added \loggingoutput 31
2001-01-07 ltoutput.dtx v1.2j	\tracingall: Use newly added \loggingoutput 31
\@writsetup: And do it in the right macro (pr/3286) 1255	2002-06-16 ltoutenc.dtx v1.95
	General: Added \textbardbl (pr/3400) 534
	Added default for \textbardbl (pr/3400) 523

2002-06-17 ltoutenc.dtx v1.95	2004-01-03 ltoutenc.dtx v1.99b
General: Corrected \c for T1	General: Added \textogonekcentered
(pr/3442) 528	(pr/3532) 528
Definition of \textexclamdown	Added composites for \k (pr/3532) 533
changed (pr/3368) 526	Use \ooalign for \k (pr/3532) . . 528
Definition of \textquestiondown	2004-01-04 ltbibl.dtx v1.1p
changed (pr/3368) 526	\nocite: Changed error message . 1056
2002-06-18 ltoutenc.dtx v1.95	2004-01-04 ltoutenc.dtx v1.99c
General: Changed def for	General: More adjustments for ogonek
\textregistered to avoid small	(pr/3532) 528
caps (pr/3420) 524	2004-01-23 ltdefs.dtx v1.1g
2002-10-01 ltfloat.dtx v1.1v	\@newenva: Use kernel version of
\thempfootnote: Use braces around	\@ifnextchar (pr/3501) 88
\itshape to keep font change local	\@testopt: Use kernel version of
(pr/3460). 1043	\@ifnextchar (pr/3501) 85
2002-10-02 ltfsbas.dtx v3.0x	\@xargdef: Use kernel version of
\DeclareFontSubstitution: Adding	\@ifnextchar (pr/3501) 84
\LastDeclaredEncoding	\@xdblarg: Use kernel version of
introduced a bug as on some	\@ifnextchar (pr/3501) 111
occasions that macro name was	2004-01-23 ltdefs.dtx v1.3g
stored in the internal lists instead	\kernel@ifnextchar: Added macro
of the actual encoding. (pr/3459) 575	(pr/3501) 110
2002-10-28 ltlists.dtx v1.0s	2004-01-28 ltclass.dtx v1.1g
\endtrivlist: Check for math mode	\@providesfile: Use kernel version of
(pr/3437) 916	\@ifnextchar (pr/3501) 1111
2002-10-28 ltoutenc.dtx v1.96	2004-01-28 ltvers.dtx v1.0k
General: coding change, to follow bug	General: Check for old format made 5
fix by DEK in plain.tex	years (pr/3601) 35
(pr/3469) 527, 536	2004-02-02 fontdef.dtx v2.3
2002-12-13 ltbibl.dtx v1.1n	General: Many things from here on
\@citex: Added \leavevmode in case	made robust 780
citation is at start of paragraph	2004-02-02 ltoutenc.dtx v1.99
(pr/3486) 1055	General: Added \textbigcircle . . 534
2003-01-01 ltfntcmd.dtx v3.3z	2004-02-04 fontdef.dtx v2.3a
General: Code checked and	General: Added bigtriangle synonyms
documentation extended by Chris 792	for stmaryrd 778
2003-05-18 ltbibl.dtx v1.1o	2004-02-04 ltspaced.dtx v1.3
\nocite: Check if we are after	\nobreakdashes: Macro added 479
\document 1056	2004-02-06 ltoutenc.dtx v1.99d
2003-08-27 ltpictur.dtx v1.1k	\@inmathwarn: New command added
\@bezier: added missing displacement	to fix severe bug: pr/3563 514
pr/3566 1004	2004-02-07 ltoutput.dtx v1.2l
\@sline: check for \@linechar being	\@doclearpage: Empty kludgeins box
empty pr/3570 984	if necessary, pr/3528 1239
2003-10-13 ltfinal.dtx v1.1e	2004-02-13 ltoutenc.dtx v1.99e
General: Added extra \lccode for \-	General: Documentation fixes: typos 509
and \textcompwordmark 1338	2004-02-15 ltbibl.dtx v1.1q
2003-12-16 ltoutput.dtx v1.2k	\@cite@ofmt: Added hook with
\@makecol: Ensure that \@elt has a	default value \hbox 1057
defined state (pr/3586) 1241	\@citex: Changed to use a hook with
2003-12-30 ltpictur.dtx v1.1j	default value \hbox 1055
\@getcirc: issue warning if circle size	2004-02-15 ltspaced.dtx v1.3a
can't be met pr/3473 995	\nobreakdashes: Added spacefactor
	setting 479

2005-07-27 ltfssdcl.dtx v3.0j		
\DeclareMathAlphabet: (MH) Make document commands robust ...	720	
\DeclareSymbolFontAlphabet: (MH) Make document commands robust	732	
\new@mathalphabet: (MH) Make document commands robust ...	721	
\non@alpherr: (MH) Change because command is now properly robust	706	
\SetMathAlphabet: (MH) Make document commands robust ...	721	
2005-09-27 ltoutenc.dtx v1.99g		
General: Replace \sh@ft by \ltx@sh@ft	526, 528, 535	
2005-09-27 ltplain.dtx v1.1y		
\ltx@sh@ft: New macro	30	
\sh@ft: Macro no longer used but left for compatibility	30	
2005-11-08 ltoutenc.dtx v1.99h		
General: Added \ij and \IJ from babel. (pr/3771)	523, 527, 529	
2005-11-10 ltmath.dtx v1.1g		
\l: (MH) Fixed potential problem in \l (pr/3399).	897	
General: (MH) Minor documentation fixes.	889	
2006-05-18 ltboxes.dtx v1.1g		
\@parboxto: Ensure \@parboxto holds the value of \@tempdima not the register itself (pr/3867)	936	
2006-09-13 ltoutput.dtx v1.1m		
General: Ensure that rule is in \normalcolor	1303	
2007-08-05 ltclass.dtx v1.1h		
\@fileswithoptions: Prevent loss of brackets PR/3965	1120	
2007-08-06 ltcntrl.dtx v1.0h		
\@fornoop: Really make defs long	426	
2007-08-31 ltfssdcl.dtx v3.0l		
\SetSymbolFont@: Font warning changed to info for encoding change (pr/3975)	719	
2009-09-24 ltvers.dtx v1.0l		
General: Stop checking for old format	35	
2009-10-20 ltfssdcl.dtx v3.0m		
\in@: More robust thanks to Heiko.	702	
2009-10-28 lttextcomp.dtx v1.99k		
General: Added Latin Modern and TeX Gyre subsets	829	
2009-11-04 lttextcomp.dtx v1.99l		
General: Added more Latin Modern and TeX Gyre subsets	829	
2009-12-14 ltfntcmd.dtx v3.4a		
\ifmaybe@ic: Macro added	795	
\maybe@ic@: Use switch \ifmaybe@ic instead of \if@tempswa	795	
\t@st@ic: Use switch \ifmaybe@ic instead of \if@tempswa	796	
2010-08-17 ltmiscen.dtx v1.1k		
\enddocument: Use braces around \input arg (pr/4124)	866	
2010-08-17 ltmiscen.dtx v1.1l		
\enddocument: Change of plan: use \@@input instead (pr/4124)	866	
2011-05-08 ltfssdcl.dtx v3.0n		
\in@: Simplified thanks to Bruno.	702	
2011-08-19 ltclass.dtx v1.1i		
\@ifclasswith: Re-jig definition after more stringent \in@ test.	1106	
2011-09-03 ltfssdcl.dtx v3.0o		
\new@mathversion: (Will) Remove \global before \newcount (unnecessary and caused etex bug).	715	
2012-01-20 ltplain.dtx v2.0b		
\loggingall: etex tracing if available	31	
2013-07-07 ltclass.dtx v1.1i		
General: Correctly describe how the date in \@ifpackagelater is used	1097	
2014-04-24 ltoutput.dtx v1.2n		
\fl@tracemessage: Renamed internal trace commands; provide as package	1292	
2014-04-27 ltfloat.dtx v1.2b		
\end@dblfloat: Inline the code to allow some coexistence with packages that hook into \end@float and do not know about the algorithm change	1034	
2014-06-10 ltfloat.dtx v1.2b		
\end@dblfloat: missing \fi added	1034	
2014-12-30 ltfinal.dtx v2.0a		
\newmarks: macro added	1332	
\newXeTeXintercharclass: macro added	1333	
2014-12-30 ltfloat.dtx v1.2a		
\@textsubscript: Command added (latexrelease)	1045	
\textsubscript: Command added (latexrelease)	1045	
2014-12-30 ltfsbas.dtx v3.0y		
\mathgroup: move allocation to ltplain.	571	
2014-12-30 ltoutput.dtx v1.2m		
General: Command updated (latexrelease)	1303	

2014-12-30 ltplain.dtx v2.0a		<code>\@stpelt</code> : Reset all within counters in one go (latexrelease)	559
<code>\e@alloc</code> : macro added	18	2015-01-11 ltcounts.dtx v1.1h	
<code>\e@alloc@chardef</code> : macro added . . .	18	<code>\TextOrMath</code> : Add command to solve robustness issues (pr/3752) (latexrelease)	567
<code>\e@alloc@top</code> : macro added	18	2015-01-11 ltfloat.dtx v1.2b	
<code>\e@ch@ck</code> : macro added	18	<code>\dblfloatplacement</code> : float order in 2-column (latexrelease)	1036
<code>\extrafloats</code> : macro added	18	<code>\xfloat</code> : Check for valid option (latexrelease)	1030
<code>\newlanguage</code> : New engine-specific allocation scheme (latexrelease) . .	17	<code>\end@dblfloat</code> : float order in 2-column (latexrelease)	1034
2014-12-30 ltspace.dtx v1.3b		2015-01-11 ltfsbas.dtx v3.0y	
<code>\@</code> : <code>\@</code> discards spaces when moving (pr3039)(latexrelease)	481	<code>\@DeclareMathSizes</code> : Allow arbitrary units (latexrelease)	580
2015-01-03 ltdefs.dtx v1.4a		2015-01-11 ltspace.dtx v1.3d	
<code>\typein</code> : use modified definition in luatex	82	<code>\@Esphack</code> : Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	473
2015-01-03 ltdirchk.dtx v1.1		<code>\@esphack</code> : Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	471
General: Enable extra primitives when Lua _T _E _X is used	3	2015-01-14 ltoutput.dtx v1.2n	
2015-01-03 ltfinal.dtx v2.0a		<code>\@addtocurcol</code> : float order in 2-column (latexrelease)	1268
General: Skip resetting codes with Unicode engines	1346	<code>\@addtodblcol</code> : float order in 2-column (latexrelease)	1283
Unicode data loading added . . .	1336	<code>\@addtonextcol</code> : float order in 2-column (latexrelease)	1278
2015-01-07 ltvers.dtx v1.0n		<code>\@doclearpage</code> : Empty kludgeins box if necessary, pr/3528	1239
<code>\@check@IncludeInRelease</code> : macro added	37	float order in 2-column (latexrelease)	1238
2015-01-08 ltboxes.dtx v1.1h		<code>\@startdblcolumn</code> : float order in 2-column (latexrelease)	1262
<code>\framebox</code> : Make Robust (latexrelease)	933	<code>\@xtryfc</code> : float order in 2-column (latexrelease)	1264
<code>\makebox</code> : Make Robust (latexrelease)	926	<code>\@ztryfc</code> : float order in 2-column (latexrelease)	1265
<code>\parbox</code> : Make Robust (latexrelease)	935	2015-01-14 ltspace.dtx v1.3e	
<code>\raisebox</code> : Make Robust (latexrelease)	944	<code>\addpenalty</code> : Avoid adding redundant skips (DPC)	476
<code>\rule</code> : Make Robust (latexrelease) .	942	2015-01-17 ltvers.dtx v1.0m	
<code>\savebox</code> : Make Robust (latexrelease)	931	<code>\@check@IncludeInRelease</code> : modified with <code>\@currname</code>	37
2015-01-08 ltdefs.dtx v1.4a		2015-01-19 ltvers.dtx v1.0o	
<code>\MakeRobust</code> : Added macro	95	<code>\@check@IncludeInRelease</code> : Optional argument	37
2015-01-08 ltlength.dtx v1.1c		2015-01-20 ltoutput.dtx v1.2m	
<code>\setlength</code> : to ensure first length argument is terminated. (latexrelease)	569	<code>\fl@tracemessage</code> : Reset <code>\IncludeInRelease</code> flags	1293
2015-01-08 ltmath.dtx v1.1h			
<code>\]</code> : Make Robust (latexrelease)	897		
<code>\]</code> : Make Robust (latexrelease)	897		
2015-01-09 ltfsini.dtx v3.1a			
<code>\em</code> : Allow <code>\emph</code> to produce small caps (latexrelease)	758		
<code>\eminershape</code> : macro added (latexrelease)	758		
2015-01-09 ltspace.dtx v1.1h			
<code>\addpenalty</code> : Donald Arseneau's fix from PR/377703 (latexrelease) . .	475		
2015-01-10 ltcounts.dtx v1.1h			
<code>\@fnsymbol</code> : Unse <code>\TextOrMath</code> (latexrelease)	567		

2015-01-22 ltvers.dtx v1.0p	2015-02-21 ltplain.dtx v2.0e
General: Preserve any <code>\everyjob</code>	General: Removed autoload code . . . 14
material inserted by a loader (<code>.ini</code>	2015-02-21 lttab.dtx v1.1n
file) 36	General: Removed autoload code . . . 947
2015-01-23 ltfinal.dtx v2.0b	2015-02-21 ltvers.dtx v1.0r
<code>\newmarks</code> : use reserved count 256 . . . 1332	General: Removed autoload code . . . 35
<code>\newXeTeXintercharclass</code> : use	2015-02-21 ltvers.dtx v1.0w
reserved count 257 1333	<code>\@check@IncludeInRelease</code> : set
2015-01-23 ltplain.dtx v2.0c	<code>\@currname</code> empty here (in case
<code>\extrafloats</code> : reserve counts 256–265 . . . 18	<code>\IncludeInRelease</code> input early) . . . 37
2015-01-24 ltfinal.dtx v2.0c	2015-02-22 ltfsscnp.dtx v3.0e
General: Skip T1-code entirely with	General: Moved all code into
Unicode engines 1336	<code>latexrelease</code> - obsolete commands
2015-02-03 ltfinal.dtx v2.0d	are no longer automatically part of
General: Set <code>\lccode</code> for - with	the kernel 697
Unicode engines 1337	2015-03-02 ltplain.dtx v2.0f
2015-02-16 ltoutenc.dtx v1.99m	<code>\e@mathgroup@top</code> : macro added . . . 18
General: Added <code>\textcommabelow</code>	<code>\newlanguage</code> : allow 255 math groups
<code>latex/4414</code> 524	in Unicode engines 17
2015-02-16 ltoutenc.dtx v1.99n	2015-03-10 ltplain.dtx v2.0g
General: Added <code>\textcommaabove</code> . . . 525	<code>\hideoutput</code> : macro added 33
Added composites for <code>ç</code> 533	<code>\loggingall</code> : Reorganize to be less
Added composites for <code>\c</code> 528	noisy 31
2015-02-16 lttextcomp.dtx v1.99m	<code>\tracingnone</code> : macro added 32
General: Added <code>lmtt</code> (Heiko Oberdiek)	2015-03-12 ltoutput.dtx v1.2m
<code>latex/4415</code> 829	General: initialize <code>\@dbldeferlist</code>
2015-02-19 ltvers.dtx v1.0q	again 1225
<code>\@check@IncludeInRelease</code> : Swap	2015-03-18 ltfssdcl.dtx v3.0q
argument order 37	<code>\DeclareSymbolFont</code> : Restrict Symbol
2015-02-20 ltplain.dtx v2.0d	fonts to 0–15 717
<code>\loggingall</code> : Spell commands	<code>\document@select@group</code> : Introduce
correctly :-)) 31	<code>\e@mathgroup@top</code> 707
2015-02-21 ltdefs.dtx v1.4b	<code>\select@group</code> : Introduce
General: Removed autoload support . . . 80	<code>\e@mathgroup@top</code> 705
2015-02-21 ltterror.dtx v1.2o	2015-03-26 ltfinal.dtx v2.0d
General: Removed autoload support . . . 427	General: Use renamed
2015-02-21 ltfiles.dtx v1.1m	<code>unicode-letters.def</code> 1336
General: Removed autoload support . . . 485	2015-04-07 ltfssbas.dtx v3.1a
2015-02-21 ltfssbas.dtx v3.0z	<code>\wrong@fontshape</code> : Try loading fd file
General: Removed autoload code . . . 571	if family has changed 592
2015-02-21 ltfsscnp.dtx v3.0d	2015-04-28 ltfinal.dtx v2.0f
General: Removed autoload code . . . 697	<code>\newXeTeXintercharclass</code> : define
2015-02-21 ltfssdcl.dtx v3.0p	<code>\xe@alloc@intercharclass</code> for
General: Removed autoload code . . . 702	compatibility with older xelatex
2015-02-21 ltfsstrc.dtx v3.0k	initialization 1333
General: Removed autoload code . . . 671	2015-05-10 ltlists.dtx v1.0t
2015-02-21 ltoutenc.dtx v1.99m	<code>\@doendpe</code> : Explicitly reset
General: Removed autoload code . . . 509	<code>\clubpenalty</code> before clearing
2015-02-21 ltoutput.dtx v1.2n	<code>\everypar</code> ; see also pr/0462 and
General: Removed autoload code . . . 1214	pr/4065 918
<code>\f@depth</code> : macro added(<code>latexrelease</code>) . . . 1238	2015-06-19 ltfinal.dtx v2.0g
2015-02-21 ltpictur.dtx v1.1k	<code>\e@alloc@intercharclass@top</code> : Use
General: Removed autoload code . . . 974	–1 for first range to get contiguous
	allocation 1333

<code>\newmarks</code> : Use <code>-1</code> for first range to get contiguous allocation	1332	<code>module_warning</code> : Function added . . .	54
2015-06-19 <code>ltpplain.dtx</code> v2.0h		<code>modules</code> : Function modified	53
General: delete spurious old definition of <code>\newtoks</code>	20	<code>create_callback</code> : Function added . .	65
<code>\e@alloc</code> : extra braces in case arguments not single token	18	<code>provides_module</code> : Function added . .	53
<code>\newlanguage</code> : Use <code>-1</code> for first range to get contiguous allocation	17	<code>luatexbase</code> : Table added	52
2015-06-23 <code>ltfinal.dtx</code> v2.0h		2015-10-02 <code>ltdirchk.dtx</code> v1.2a	
General: set <code>\patch@level</code> in <code>ltvers</code> rather than in <code>ltfinal/ltpatch</code> . .	1350	General: Allow backing out of unprefixed names	3
2015-06-23 <code>ltvers.dtx</code> v1.0t		2015-10-02 <code>ltluatex.dtx</code> v1.0b	
General: set <code>\patch@level</code> in <code>ltvers</code> rather than in <code>ltfinal/ltpatch</code>	35	General: Fix backing out of $\TeX$ code	51
2015-08-06 <code>ltpplain.dtx</code> v2.0i		2015-10-02 <code>ltluatex.dtx</code> v1.0c	
<code>\extrafloats</code> : Add <code>\string</code> in case argument is not an unexpandable primitive	19	General: Allow backing out of Lua code	51
2015-08-23 <code>ltdirchk.dtx</code> v1.2		2015-10-02 <code>ltluatex.dtx</code> v1.0e	
General: Do not use <code>luatex</code> prefix	3	<code>uninstall</code> : Function added	70
2015-08-23 <code>ltvers.dtx</code> v1.0v		2015-10-03 <code>ltluatex.dtx</code> v1.0f	
General: Allow negative <code>patchlevel</code> for pre-release	36	<code>provides_module</code> : use <code>luatexbase_log</code>	53
2015-08-30 <code>ltpplain.dtx</code> v2.1a		2015-10-27 <code>ltpplain.dtx</code> v2.1b	
<code>\newinsert</code> : new <code>\newinsert</code> implementation	20	<code>\extrafloats</code> : Use global assignment when switching to extended range	19
2015-09-205 <code>ltoutput.dtx</code> v1.3a		2015-11-07 <code>ltspace.dtx</code> v1.3f	
General: extended <code>\@freelist</code>	1224	<code>\@esphack</code> : Only space if there is no space at the end of the hlist <code>latex/4443</code>	471
2015-09-24 <code>ltluatex.dtx</code> v1.0a		2015-11-14 <code>ltluatex.dtx</code> v1.0g	
<code>call_callback</code> : Function added	66	General: Track Lua $\TeX$ changes for (<code>new</code>) <code>token.create</code>	54
<code>callback.register</code> : Function modified	62	2015-11-18 <code>ltpplain.dtx</code> v2.2a	
<code>callback_identifiers</code> : Function added	70	<code>\newlanguage</code> : Extended stream allocation in <code>luatex</code> (0.85)	17
<code>\catcodetable@atletter</code> : Macro added	48	2015-11-19 <code>ltpplain.dtx</code> v2.2b	
<code>\catcodetable@initex</code> : Macro added	48	<code>\newlanguage</code> : Only extend allocation of write streams (see <code>luatex</code> list) .	17
<code>\catcodetable@latex</code> : Macro added .	48	2015-11-27 <code>ltluatex.dtx</code> v1.0h	
<code>\catcodetable@string</code> : Macro added	48	<code>callback_identifiers</code> : Match test in in-callback <code>latex/4445</code>	70
<code>add_to_callback</code> : Function added . .	67	<code>in_callback</code> : Guard against undefined list <code>latex/4445</code>	66
<code>remove_from_callback</code> : Function added	69	2015-11-29 <code>ltluatex.dtx</code> v1.0i	
<code>new_attribute</code> : Function added	55	General: Declare this as local before used in the module error definitions (PHG)	52
<code>disable_callback</code> : Function added .	70	<code>call_callback</code> : Check name is not nil in error message (PHG)	66
<code>in_callback</code> : Function added	66	<code>create_callback</code> : Check name is not nil in error message (PHG)	65
<code>\newattribute</code> : Macro added	47	2015-12-02 <code>ltluatex.dtx</code> v1.0j	
<code>\newcatcodetable</code> : Macro added . . .	47	General: Adjust <code>hashtokens</code> to store the result of <code>tex.hashtokens()</code> , not the function (PHG)	54
<code>\newluabytecode</code> : Macro added	50	Assorted typos fixed (PHG)	45
<code>\newluachunkname</code> : Macro added . . .	50	Declaration/use of <code>first_head</code> fixed (PHG)	53
<code>\newluafunction</code> : Macro added	49		
<code>\newwhatsit</code> : Macro added	50		
<code>module_error</code> : Function added	54		
<code>module_info</code> : Function added	54		

Remove nonlocal iteration variables (PHG)	45	<code>\DeclareMathAccent</code> : Check for <code>mathaccent</code> not <code>\mathaccent</code> . . .	723
Remove unreachable code after calls to <code>error()</code> (PHG)	45	<code>\DeclareMathRadical</code> : Check for <code>radical</code> not <code>\radical</code>	731
2015-12-02 ltluatex.dtx v1.0k		<code>\DeclareMathSymbol</code> : Check for <code>mathchar</code> not <code>\mathchar</code>	726
General: resolve name and <code>i.description</code> (PHG)	64	2016-03-13 ltluatex.dtx v1.0n	
<code>call_callback</code> : Give more specific error messages (PHG)	66	General: contribute <code>_filter</code> added	61
<code>add_to_callback</code> : Give more specific error messages (PHG)	67	<code>insert_local_par</code> added	61
<code>remove_from_callback</code> : adjust initialization of <code>cb local</code> (PHG) . .	69	2016-03-29 ltpictur.dtx v1.1l	
Give more specific error messages (PHG)	69	<code>\@oval</code> : add setting of line tests . . .	997
<code>create_callback</code> : Give more specific error messages (PHG)	65	<code>initialize tests</code>	996
2015-12-10 ltfinal.dtx v2.0i		<code>\@ovhorz</code> : use glue not leaders if horizontal line not required	999
General: Use new common Unicode data loaders	1336	<code>\@ovvert</code> : use glue not leaders if vertical line not required	998
2015-12-18 ltluatex.dtx v1.0l		<code>\if@ovhline</code> : macro added (<code>latex/4452</code>)	996
General: Load Unicode data from source	47	<code>\if@ovvline</code> : macro added (<code>latex/4452</code>)	996
2016-01-04 ltfinal.dtx v2.0j		2016-04-22 ltfinal.dtx v2.0q	
General: Do not set up inter character classes for XeTeX	1336	<code>\e@alloc@intercharclass@top</code> : XeTeX 0.99996 has 4096 char classes not 256	1333
<code>\e@alloc@intercharclass@top</code> : Start allocation at one not three	1333	2016-06-19 ltoutenc.dtx v1.99m	
2016-01-05 ltfinal.dtx v2.0k		General: OT1 definition (was duplicate T1 definition)	528
<code>\e@alloc@intercharclass@top</code> : Remove duplicated code	1333	2016-06-20 ltclass.dtx v1.1j	
2016-01-05 ltfinal.dtx v2.0l		General: don't declare as <code>\@onlypreamble</code>	1105
General: Correct <code>latexrelease</code> guards	1336	2016-07-29 ltplain.dtx v2.2c	
Ensure old definitions for inter-character class toks are available using <code>latexrelease</code>	1336	<code>\extrafloats</code> : use <code>\global \chardef</code> . .	19
Missing brace	1336	<code>\newinsert</code> : fix for <code>tlb-newinsert-001</code> .	20
2016-01-05 ltfinal.dtx v2.0m		2016-10-02 ltclass.dtx v1.2a	
General: Undefine XeTeX classes when using patching an older kernel . .	1336	<code>\@ifclasswith</code> : Ignore spaces while checking for option clash	1106
2016-01-05 ltfinal.dtx v2.0p		<code>\ExecuteOptions</code> : Ignore spaces in argument	1116
General: Only apply XeTeX change if XeTeX is in use	1336	2016-10-15 ltdirchk.dtx v1.2b	
2016-02-11 ltluatex.dtx v1.0m		General: Require eTeX	3
General: <code>pdf_stream_filter_callback</code> removed	62	2016-10-15 lterror.dtx v1.2p	
<code>process_rule</code> , <code>[hv]pack_quality</code> <code>append_to_vlist_filter</code> added . . .	61	General: Require eTeX	427
<code>read_cidmap_file</code> added	61	2016-10-15 ltfinal.dtx v2.0r	
<code>show_warning_message</code> added . . .	61	General: Require eTeX	1332
<code>token_filter</code> removed	61	2016-10-15 ltfinal.dtx v2.0s	
2016-02-18 ltfssdcl.dtx v3.0r		General: Tidy up status of char 127 .	1332
<code>\DeclareMathDelimiter</code> : Check for <code>delimiter</code> not <code>\delimiter</code>	728	2016-10-15 ltfssini.dtx v3.1b	
		General: Require eTeX	735
		2016-10-15 ltplain.dtx v2.2d	
		General: Require eTeX	14
		2016-10-16 ltplain.dtx v2.3a	
		<code>\newlanguage</code> : Allow languages up to 16383 in <code>luatex</code>	17

2016-10-19 ltcounts.dtx v1.1j		declare composites with empty base	
\TextOrMath: Test directly for		for hat and tilde, use same slots for	
\protected	567	\textasciicircum ans	
2016-11-06 ltplain.dtx v2.3b		\textasciitilde	541
General: Drop \outer entirely	14	declare straight quotes using new	
2016-11-09 ltclass.dtx v2.1b		\remove@tlig command	541
\@fileswithoptions: Improve \ifx		2017-02-22 ltoutenc.dtx v2.0g	
tests PR/4497	1120	General: Fix typo introduced at 2.0f	541
2016-11-17 ltluatex.dtx v1.0p		2017-02-24 ltoutenc.dtx v2.0h	
General: call _edit added	61	General: introduce	
2016-12-03 fontdef.dtx v3.0a		\DeclareUnicodeAccent	541
General: (DPC) Default to TU		\DeclareTextCompositeCommand: add	
encoding for Unicode TeX engines	768	check whether the accent command	
\shapedefault: (DPC) Default to TU		is defined for this encoding	517
encoding for Unicode TeX engines	772	2017-03-08 ltclass.dtx v1.2c	
2016-12-04 ltoutenc.dtx v2.0a		General: add \@parse@version@dash	
General: Added TU encoding	541	to support yyyy-mm-dd as well as	
2017-01-01 ltoutput.dtx v1.3b		yyyy/mm/dd	1105
General: make fpmn negative so		2017-03-09 ltfinal.dtx v2.0t	
ignored even if float height is		\l@nohyphenation: ensure	
negative	1302	\l@nohyphenation is defined. .	1339
2017-01-10 ltfssbas.dtx v3.2a		2017-03-09 ltmiscen.dtx v1.1m	
\showhyphens: Add version of		\@verbatim: Use \language not	
\showhyphens that works with		\hyphenchar	882
XeTeX.	598	\verb: Use \language to stop	
2017-01-23 ltoutenc.dtx v2.0b		hyphenation	887
General: Added TU specific commands		2017-03-10 ltfiles.dtx v1.1n	
in ASCII range pr/4500	541	\document: Save language default ..	488
2017-01-24 ltoutenc.dtx v2.0c		2017-03-10 ltoutput.dtx v1.3c	
General: Declare TU composites for i		\@writsetup: Reset \language ..	1254
and j	541	2017-03-13 ltdefs.dtx v1.5a	
Make \textasteriskcentered		\:-: Define \- in terms of \hyphenchar	112
U+2217 not U+204E	541	2017-03-27 ltdefs.dtx v1.5b	
TeX ligature syntax for xetex and		\@dischyp: Define \@dischyp after	
luatex reversed	541	\-	112
2017-01-24 ltoutenc.dtx v2.0d		2017-03-28 ltluatex.dtx v1.1e	
General: Declare macron composites		General: glyph_stream_provider	
for YyGg	541	added	62
2017-02-12 ltoutenc.dtx v2.0e		2017-03-29 ltboxes.dtx v1.3a	
General: Declare fallback code for		\@arrayparboxrestore: Reset	
\textasteriskcentered	541	\lineskiplimit	938
2017-02-18 ltluatex.dtx v1.1c		2017-04-05 ltoutenc.dtx v2.0i	
new_attribute: Parameterize count		\DeclareTextCompositeCommand:	
used in tracking	55	Declare accent command if not	
new_bytecode: Parameterize count		already declared when declaring a	
used in tracking	56	composite.	517
new_chunkname: Parameterize count		2017-04-10 ltplain.dtx v2.3c	
used in tracking	57	\newlanguage: Correction to code to	
new_whatitsit: Parameterize count used		skip write18 in luatex	17
in tracking	56	2017-04-11 ltoutput.dtx v2.4a	
2017-02-19 ltoutenc.dtx v2.0f		\newpage: account for the depth of the	
General: add \@empty to guard against		last row of the page	1228
3rd argument being empty	527	2017-12-17 ltoutput.dtx v1.4b	
		\@addtonextcol: fix doc guards ..	1278

2018-01-06 ltdefs.dtx 1.5c		2018-05-29 ltclass.dtx v1.2j	
\@ifundefined: Avoid defining		\endfilecontents: use \curname not	
undefined commands to \relax .	108	\@undefined	1134
2018-02-18 ltclass.dtx v1.2d		2018-08-11 ltoutenc.dtx v2.0j	
\@ifl@ter: Added 0 up front to make		General: Provide \guillemetleft and	
bad data come out as 0.	1105	\guillemetright	530, 535, 545
General: Introduce rollback concept	1139	2018-08-18 ltluatex.dtx v1.1h	
2018-03-08 ltcounts.dtx v1.1k		General: append_to_vlist_filter is	
\@ifbothcounters: Interface added	563	exclusive	61
\@removefromreset: Interface added	562	2018-08-24 ltfinal.dtx v2.1f	
\counterwithin: Interface added . .	564	\document@default@language: Add to	
\counterwithout: Interface added .	565	latexrelease (github/68)	1339
2018-03-24 ltclass.dtx v1.2e		2018-09-02 ltsect.dtx v1.1b	
\pkgcls@use@this@release: Use full		\@dottedtocline: Prevent protrusion	
file name for old release	1145	(https://tex.stackexchange.com/q/172785/10109)	1024
2018-03-25 ltfinal.dtx v2.1a		2018-09-24 fontdef.dtx v3.0b	
General: default to UTF-8	1342	General: Start LR-mode if necessary	
\UseRawInputEncoding: Macro		(gh/49)	785
added	1343	2018-09-24 ltmath.dtx v1.2b	
2018-03-27 ltclass.dtx v1.2f		\smash: Start LR-mode if necessary	
\endfilecontents: Use full file name		(gh/49)	893
for old release	1133	\phantom: Start LR-mode if	
2018-04-06 ltfinal.dtx v2.1b		necessary (gh/49)	892
\UseRawInputEncoding: Undo changes		2018-09-24 ltspace.dtx v1.3h	
to \DeclareFontEncoding@ and		\enspace: Start LR-mode if necessary	
definition of		(gh/49)	482
\DeclareUnicodeCharacter . . .	1343	\leavevmode@ifvmode: Macro added	
2018-04-07 ltfinal.dtx v2.1c		(gh/49)	482
\UseRawInputEncoding: Undefine		2018-09-26 ltdefs.dtx v1.5e	
\inputencodingname	1343	\renew@command: Always explicitly	
2018-04-08 ltclass.dtx v1.2g		generate a space after the curname	
\@ifl@ter: Strip leading spaces from		and not rely on \noexpand to save	
dates.	1105	tokens (gh/41)	87
2018-04-08 ltclass.dtx v1.2h		2018-09-26 ltmiscen.dtx v1.1n	
\@onefilewithoptions: Pass		\@writefile: Sometimes mask the	
expanded date	1141	endline char when writing to files	
2018-04-08 ltfinal.dtx v2.1d		(github/73)	871
General: Delay full UTF-8 handling to		\add@percent@to@temptokena:	
\everyjob	1343	Sometimes mask the endline char	
2018-04-11 ltcounts.dtx v1.1l		when writing to files (github/73)	871
\counterwithin: Correct default		\protected@file@percent: Sometimes	
(issue/38)	564	mask the endline char when	
2018-05-02 ltluatex.dtx v1.1g		writing to files (github/73)	871
General: find_sfd_file removed	60	2018-09-26 ltsect.dtx v1.1c	
finish_synctex_callback added . . .	61	\addcontentsline: Sometimes mask	
glyph_not_found added	62	the endline char when writing to	
read_sfd_file removed	61	files (github/73)	1021
2018-05-08 ltclass.dtx v1.2i		2018-10-10 ltspace.dtx v1.3i	
\pkgcls@parse@date@arg: Make		\@esphack: Don't introduce	
suspicious rollback a warning not		breakpoints if @nobreak is true	
error: github issue 43	1143	and after sections	471
2018-05-11 ltfinal.dtx v2.18			
General: Make invalid UTF-8 also safe,			
for legacy filesystem encodings .	1343		

2018-10-11 ltmiscen.dtx v1.1o	font_descriptor_objnum_provider added	62
\@@sverb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	make_extensible added	62
\@setupverbvisibleospace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	new_graf added	61
\@verbvisibleospacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	page_objnum_provider added	62
\asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	process_pdf_image_content added	62
verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	wrapup_run added	61
\verbvisibleospace: Provide \verbvisibleospace such that it is usable in normal text (github/70)	2019-07-01 ltclass.dtx v1.3a	
Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	\endfilecontents: Support UTF8 and spaces in filecontents environment file name	1131
2018-10-21 ltuatex.dtx v1.1i	2019-07-01 ltfiles.dtx v1.2a	
new_luafunction: Function added	\IfFileExists: Support UTF-8	500
2018-11-09 ltbib1.dtx LaTeX2e	\includeonly: Support UTF-8	493
\bibliography: Zap spaces in the argument as BibTeX doesn't support them (github/88)	\setcurrfile: Support UTF-8	499
2018-11-18 ltoutenc.dtx v2.0k	2019-07-09 ltfssbas.dtx v3.2c	
General: Provide \Hwithstroke and \hwithstroke	\DeclareErrorFont: Don't set any \f@... macros	591
2018-11-19 ltoutenc.dtx v2.0k	2019-07-09 ltfssini.dtx v3.1c	
General: Added \Hwithstroke and \hwithstroke	General: Explicitly set some defaults	764
2018-11-28 ltoutput.dtx v1.4d	2019-08-22 ltxref.dtx v1.11	
\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94)	\labelformat: Commanded moved from varioref.sty	847
2018-12-30 lttab.dtx v1.1p	\Ref: Commanded moved from varioref.sty	847
\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102)	\refstepcounter: Allow \p@... to have an argument	846
2019-02-07 ltfilehook.dtx v1.1o	2019-08-27 fontdef.dtx v3.0c	
\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109)	General: Various commands made robust throughout the file	778
2019-02-07 ltfiles.dtx v1.1o	2019-08-27 ltboxes.dtx v1.3b	
\@swaptwoargs: Helper macro added	General: Various commands made robust	925
2019-02-07 ltfssbas.dtx v3.2b	2019-08-27 ltclass.dtx v1.3b	
\define@newfont: Changed wording of warning (github/107)	\endfilecontents: Make various commands robust	1131
2019-06-18 ltuatex.dtx v1.1j	2019-08-27 ltdefs.dtx v1.5f	
General: finish_synctex_callback renamed finish_synctex	General: Make various commands robust	113
	\MakeRobust: Make the assignments global as we may need to apply them inside a group	95
	2019-08-27 ltfilehook.dtx v1.2b	
	\unqu@tefilef@und: Make command robust	1170
	2019-08-27 ltfiles.dtx v1.2b	
	\IfFileExists: Make command robust	500
	2019-08-27 ltfssbas.dtx v3.2d	
	General: Make various commands robust	571
	2019-08-27 ltfssdcl.dtx v3.0s	
	\DeclareMathAccent: Make math accents robust	723

<code>\set@mathdelimenter</code> : Make math delimiters robust	730	2019-09-11 ltclass.dtx v1.3c	
2019-08-27 ltffssini.dtx v3.1d		<code>\endfilecontents</code> : Support optional argument for filecontents	1131
General: Make various commands robust	735	2019-09-14 ltfinal.dtx v2.1h	
2019-08-27 ltidxglo.dtx v1.1f		<code>\@uclclist</code> : Expand UTF8 chars when case changing (github/177)	1347
General: Make <code>\index</code> and <code>\glossary</code> robust	1050	2019-09-16 ltxref.dtx v1.1m	
2019-08-27 ltlength.dtx v1.1d		General: Correctly revert the <code>\p@...</code> change	847
General: Make various command robust	569	2019-09-21 fontdef.dtx v3.0d	
2019-08-27 ltlogos.dtx v1.1j		General: Distangle alias (gh/184)	779, 783
<code>\TeX</code> : Make <code>\TeX</code> command robust	484	2019-10-02 ltexpl.dtx v0.0	
2019-08-27 ltmath.dtx v1.2c		General: Initial version	72
General: Make various commands robust	889	2019-10-02 ltfinal.dtx v2.2	
2019-08-27 ltmiscen.dtx v1.1p		General: Load ltexpl	1351
General: Make various commands robust	864	2019-10-02 ltluatex.dtx v1.1k	
<code>\begin</code> : Make command robust	873	General: <code>linebreak_filter</code> is exclusive	61
<code>\end</code> : Make command robust	875	<code>mlist_to_hlist</code> is exclusive	61
2019-08-27 ltoutput.dtx v1.4e		<code>process_rule</code> is exclusive	61
<code>\@begindvibox</code> : Make <code>\AtBeginDvi</code> robust	1225	2019-10-07 lttab.dtx v1.1q	
2019-08-27 ltpage.dtx v1.0l		<code>\extracolsep</code> : This needs to expand	959
General: Make various commands robust	1089	2019-10-11 ltfiles.dtx v1.2c	
2019-08-27 ltpictur.dtx v1.1m		<code>\set@curr@file</code> : Remove one brace group	499
General: Make various commands robust	974	2019-10-11 ltfsstrc.dtx v3.0l	
Remove several unnecessary <code>\gdef</code> definitions	974	<code>\@font@aliasinfo</code> : Added 'alias' size function	694
2019-08-27 ltsect.dtx v1.1d		2019-10-18 ltclass.dtx v1.3d	
General: Make various commands robust	1012	<code>\load@onefilewithoptions</code> : Initialize <code>\...-h@ok</code> only when loading the package or class (gh/198)	1122
2019-08-27 ltspc.dtx v1.3j		2019-10-22 ltluatex.dtx v1.1j	
General: Make various commands robust	463	General: <code>page_objnum_provider</code> and <code>process_pdf_image_content</code> classified data	62
2019-08-27 lttab.dtx v1.1q		2019-10-25 ltmiscen.dtx v1.1q	
General: Make various commands robust	947	<code>\add@percent@to@temptokena</code> : Allow unbalanced conditionals in #1 (gh/202)	871
Remove several unnecessary <code>\gdef</code> definitions	947	2019-10-26 ltfiles.dtx v1.2d	
2019-08-30 lterror.dtx v1.2q		<code>\iffileonpath</code> : quote on openin	502
<code>\conditionally@traceoff</code> : Macro added	437	<code>\IfFileExists</code> : don't quote name	500
<code>\conditionally@traceon</code> : Macro added	437	<code>\IfFileExists@</code> : quote on openin	500
2019-09-09 ltfsdcl.dtx v3.0s		<code>\set@curr@file</code> : remove quotes	499
<code>\DeclareMathSymbol</code> : Allow definition if the math symbol was a command already robust	726	2019-11-01 ltdirchk.dtx v1.3a	
		<code>\filename@parse</code> : take last . not first	12
		2019-11-02 ltmiscen.dtx v1.1s	
		<code>\@centercr</code> : Make <code>\@centercr</code> robust (gh/203)	878
		2019-11-02 ltspc.dtx v1.3k	
		<code>\@normalcr</code> : Make also <code>\@normalcr</code> robust	467

2019-11-09 ltfiles.dtx v1.2e	2020-01-22 lttextcomp.dtx v1.0b
\set@curr@file: expand and \string	\tc@subst: The overall default is
before removing quotes 499	\textcompsubstdefault not
2019-11-10 ltmiscen.dtx v1.1r	\substdefault 800
\add@percent@to@temptokena: fix to	2020-01-25 fontdef.dtx v3.0f
special comment catcodes	General: Load tlenc.def last (gh/255) 768
(gh/202) 871	2020-01-25 ltoutenc.dtx v2.0m
2019-11-11 ltfiles.dtx v1.2f	General: Load each encoding file only
\iffileonpath: make \@filef@und	once (gh/255) 554
match quoting used on \openin . 502	2020-01-27 ltclass.dtx v1.3g
2019-11-22 ltoutenc.dtx v2.0l	\endfilecontents: Fix typo in error
General: Avoid spurious if fontenc	message 1133
selects LY1 as default encoding	2020-01-28 ltclass.dtx v1.3h
(gh/199) 555	\endfilecontents: Allow spaces in
2019-11-29 ltclass.dtx v1.3e	option string and display only
\@pr@videpackage: Protect package	unknown options not the whole
info text (gh/52) 1109	option list (gh/256) 1131
2019-12-17 fontdef.dtx v3.0e	2020-01-31 ltvers.dtx v1.1e
\mddefault: Set \bfdefault to “b” 771	General: Allow for upcoming format as
\shapedefault: Set \shapedefault	pre-release 0 36
explicitly to “n” 772	2020-02-02 ltluatex.dtx v1.1l
\updefault: Set \updefault to “up” 771	General: Add reverselist callback type 64
2019-12-17 ltfntcmd.dtx v3.4c	glyph_info added 62
\textssc: Macro added 793	page_order_index added 62
2019-12-17 ltfssbas.dtx v3.2e	post_linebreak_filter is
\usefont: Don’t call \fontseries or	reverselist 61
\fontshape 582	create_callback: Provide proper
2019-12-17 ltfssini.dtx v3.1e	fallbacks for user-defined callbacks
General: Provide custom series	without user-provided default
settings a la mweights 736	handler 65
\DeclareEmphSequence: Provide \emph	2020-02-05 ltfssini.dtx v3.1g
sequences 756	\DeclareFontSeriesDefault:
2019-12-18 ltoutenc.dtx v2.0m	Clarified error text 737
General: Don’t fake	Corrected misspelled csname
\textcompwordmark; take default	(gh/264) 737
from T1 instead 523	2020-02-05 lttextcomp.dtx v2.0n
\add@accent: Avoid code that breaks	General: Changed the package default
\accent 516	to info (gh/262) 818
2019-12-21 fontdef.dtx v3.0e	Ensure we are on a new format
General: Distangle alias (gh/184) 777–780	(gh/260) 818
2020-01-05 ltclass.dtx v1.3f	2020-02-07 ltfssini.dtx v3.1h
\endfilecontents: Support more	\symbol: XeTeX-specific version to
write streams in LuaTeX gh/238 1131	avoid bug in maths mode. 760
2020-01-11 ltfssini.dtx v3.1f	2020-02-10 ltfssaxes.dtx v1.0c
\rmfamily: Streamlined	\fontseries: Switch
implementation with hook 751	\if@forced@series added 656
\ttfamily: Streamlined	\fontseriesforce: Switch
implementation with hook 751	\if@forced@series added 656
2020-01-20 ltfssdcl.dtx v3.0t	\if@forced@series: Switch
\set@mathdelimater: fix for gh/251 730	\if@forced@series added 656
2020-01-20 ltoutenc.dtx v2.0n	2020-02-10 ltfssini.dtx v3.1h
General: fix for gh/251 524	\@defaultfamilyhook: Add
	\@defaultfamilyhook to
	\normalfont (gh/269) 752

<code>\reset@font</code> : Add		<code>\update@series@target@value</code> : Drop	
<code>\@defaultfamilyhook</code> to		surplus “m” from <code>\reserved@</code>	
<code>\normalfont</code> (gh/269)	761	(gh/291)	742
2020-02-10 lttextcomp.dtx v1.0c		2020-02-27 ltdefns.dtx v1.5g	
General: Use <code>\@tabacckludge</code> for		<code>\@gobblethree</code> : Macro added	90
tabbing where necessary (gh/271)	804	2020-02-27 ltfssaxes.dtx v1.0d	
2020-02-11 fontdef.dtx v3.0g		<code>\series@maybe@drop@one@m</code> : Drop	
General: Provide value for		“m” in certain values from a fixed	
<code>\@fontenc@load@list</code> (gh/273) .	768	list (gh/293)	659
2020-02-11 ltfssini.dtx v3.1h		<code>\set@target@series</code> : Drop “m” only	
General: Provide default value for		in a specific set of values (gh/293)	659
<code>\@fontenc@load@list</code> (gh/273) .	764	2020-02-27 ltfssbas.dtx v3.2g	
2020-02-11 ltoutenc.dtx v2.0o		<code>\DeclareFontShape@</code> : Only “m” if the	
General: Update		series value is a member of a fixed	
<code>\@fontenc@load@list</code> with option		list and issue warning if doing it	
list (gh/273)	555	(gh/293)	572
2020-02-14 ltpictur.dtx v1.1n		2020-03-02 ltexpl.dtx v1.0a	
<code>\linethickness</code> : Suppress spaces		General: Don’t load expl3 if already in	
following the declaration (gh/274)	979	the format (gh/295)	72
2020-02-18 ltfssini.dtx v3.1i		2020-03-05 ltexpl.dtx v1.1	
<code>\bfseries</code> : Make the <code>\ifx</code> selection		General: Load xparse.ltx if	
outside of <code>\fontseries</code> argument		<code>\NewDocumentCommand</code> is not	
so that it is not done several times	746	defined by expl3.ltx	72
<code>\mdseries</code> : Make the <code>\ifx</code> selection		2020-03-06 ltboxes.dtx v1.3c	
outside of <code>\fontseries</code> argument		<code>\clap</code> : Macro <code>\clap</code> added	946
so that it is not done several times	746	2020-03-07 ltluatex.dtx v1.1m	
<code>\prepare@family@series@update</code> : No		<code>remove_from_callback</code> : Do not call	
series auto-update when forced		callback.register for user-defined	
(gh/277)	740	callbacks	69
Recognize current family if it is not		2020-03-07 ltmath.dtx v1.2e	
a “meta” family and auto-update		<code>\negthickspace</code> : Add <code>amsmath</code>	
series using <code>\bfdefault</code> (gh/277)	741	math/text spacing commands to	
2020-02-18 ltmath.dtx v1.2d		the kernel (gh/303)	895
<code>\mathindent</code> : Make <code>\mathindent</code> a		2020-03-07 ltspace.dtx v1.3l	
skip register to match <code>amsmath</code>		General: Moved <code>\thinspace</code> ,	
(gh/252)	904	<code>\negthinspace</code> and <code>\,</code> to	
<code>equation</code> : Separate formula and eqn		ltmath.dtx (gh/303)	463
number by at least a space in fleqn		2020-03-19 fontdef.dtx v3.0h	
option	905	General: Support legacy use of	
2020-02-20 ltclass.dtx v1.3j		<code>\bfdefault</code> and <code>\mddefault</code>	
<code>\endfilecontents</code> : Fix missing quotes		(gh/306)	771
around file name (gh/284)	1133	2020-03-19 ltfssdcl.dtx v3.0u	
2020-02-24 ltfssbas.dtx v3.2f		<code>\document@select@group</code> : fix for	
<code>\DeclareFontShape@</code> : Drop surplus		(gnats/3357)	708
“m” in series when defining		2020-03-19 ltfssini.dtx v3.1k	
fontshape (gh/289)	572	<code>\DeclareFontSeriesDefault</code> : Support	
2020-02-25 ltfssini.dtx v3.1j		legacy use of <code>\bfdefault</code> and	
<code>\bf@def@ult</code> : Drop surplus “m” from		<code>\mddefault</code> (gh/306)	737
<code>\bfdef@ult</code> and <code>\mddef@ult</code>		<code>\maybe@update@bfseries@defaults</code> :	
(gh/291)	749	Support legacy use of <code>\bfdefault</code>	
<code>\prepare@family@series@update</code> :		and <code>\mddefault</code> (gh/306)	746
Drop surplus “m” from		<code>\mdseries</code> : Support legacy use of	
<code>\target@series@value</code> (gh/291)	741	<code>\bfdefault</code> and <code>\mddefault</code>	
		(gh/306)	746

2020-04-06 ltfssini.dtx v3.1m			
\bf@default: Hook added (gh/306)	749	\@yargarraycr: Support calc syntax	
\maybe@update@bfseries@defaults:		(gh/152)	963
Hook added (gh/306)	746	2020-04-22 ltmiscen.dtx v1.1u	
\maybe@update@mdseries@defaults:		\@sverb: Drop spaces before \verb	
Hook added (gh/306)	747	delimiter (gh/327)	885
2020-04-07 ltclass.dtx v1.3k		2020-04-22 ltoutenc.dtx v2.0p	
\IfFormatAtLeastTF: Macro added;		General: y unicode value in tuenc.def	509
also in rollback (gh/168)	1104	2020-04-29 lttextcomp.dtx v1.0d	
\load@onefile@withoptions: Use		General: Make all capital accents text	
different method to ignore		commands for hyperref (gh/332)	804
unprocessed options (gh/22)	1126	2020-05-02 ltfiles.dtx v1.2g	
\ProcessOptions*: Use different		\@include: Support spaces in	
method to ignore unprocessed		filenames by enclosing the names	
options (gh/22)	1115	of .aux-files in quotes (gh/217)	495
\RequirePackageWithOptions: Use		\includeonly: Get rid of leading and	
different method to ignore		trailing spaces from the filename	
unprocessed options (gh/22)	1118	(gh/217)	493
2020-04-09 ltfloat.dtx v1.2d		Improved support for spaces in	
\@textsubscript: Set non-zero		filenames (gh/217)	493
baseline (gh/249)	1045	Pass the filename to \@include by	
\textsubscript: Set non-zero baseline		value instead of by reference	
(gh/249)	1045	(gh/217)	493
2020-04-13 ltfssdcl.dtx v3.0v		2020-05-05 ltxref.dtx v1.1n	
\process@table: Small update for		\refstepcounter: record the counter	
speed.	713	name in \@currentcounter	845
2020-04-13 ltfssini.dtx v3.1n		2020-05-06 ltspace.dtx v1.3n	
\init@series@setup: Handling		General: Made softthyphen active in	
\seriesdefault changes (gh/315)	745	TU engines	483
\seriesdefault@kernel: Handling		2020-05-09 ltdefns.dtx v1.5j	
\seriesdefault changes (gh/315)	765	\@if@DeclareRobustCommand: Added	
2020-04-21 ltmath.dtx v1.2f		\DeclareCommandCopy (gh/239)	103
\@yeqncr: Support calc syntax		\DeclareCommandCopy: Added	
(gh/152)	902	\DeclareCommandCopy (gh/239)	100
2020-04-21 ltmiscen.dtx v1.1t		2020-05-11 ltdefns.dtx v1.5j	
\@icentercr: Support calc syntax		\@dischyph: Do not overwrite \-	
(gh/152)	879	under LuaTeX	112
2020-04-21 ltpictur.dtx v1.1o		2020-05-15 ltdefns.dtx v1.5g	
\@istackcr: Support calc syntax		\typeout: Allow \par in the argument	
(gh/152)	980	(gh/335)	80
2020-04-21 ltspace.dtx v1.3m		2020-05-19 ltfssaxes.dtx v1.0e	
\@hspace: Support calc syntax		\series@maybe@drop@one@m: Need to	
(gh/152)	481	use \edef (gh/336)	660
\@newline: Support calc syntax		2020-05-19 ltfssini.dtx v3.2a	
(gh/152)	468	\IfFontSeriesContextTF: Macros	
\@vspace@calcify: Support calc		added (gh/335)	753
syntax (gh/152)	468	2020-05-31 ltmiscen.dtx v1.1u	
\@vspacer: Support calc syntax		\centering: Added	
(gh/152)	478	\finalhyphendemerits setting	
\addvspace: Support calc syntax		(gh/247)	879
(gh/152)	474	\raggedleft: Added	
2020-04-21 lttab.dtx v1.1r		\finalhyphendemerits setting	
\@itabcr: Support calc syntax		(gh/247)	880
(gh/152)	953		

<p><code>\raggedright</code>: Added</p> <p><code>\finalhyphendemerits</code> setting (gh/247) 880</p> <p>2020-06-04 <code>ltexpl.dtx</code> v1.2c</p> <p>General: Define a local version of some $\LaTeX$ basic macros to support package loading 72</p> <p>2020-06-04 <code>ltfinal.dtx</code> v2.2a</p> <p>General: Load <code>ltexpl</code> in <code>ltdefns</code> . . 1351</p> <p>2020-06-05 <code>ltclass.dtx</code> v1.3l</p> <p><code>\@currnamestack</code>: Added</p> <p><code>\@expl@pop@filename@@</code> 1102</p> <p>Added <code>\@expl@push@filename@@</code> and <code>\@expl@push@filename@aux@@</code> . 1100</p> <p>2020-06-05 <code>ltfiles.dtx</code> v1.2h</p> <p><code>\document</code>: Added hook to load l3backend code 487</p> <p>2020-06-10 <code>ltluatex.dtx</code> v1.1n</p> <p>General: Define</p> <p><code>\@gobble/\@firstofone</code> even for $\LaTeX$ to allow early loading. 45</p> <p>2020-07-04 <code>ltoutenc.dtx</code> v2.0q</p> <p>General: Implement <code>\remove@tlig</code> in Lua$\TeX$ without font reloading . . 542</p> <p>2020-07-08 <code>ltexpl.dtx</code> v1.2d</p> <p>General: Add a last-minute hook for <code>expl3</code> 73</p> <p>2020-07-08 <code>ltfinal.dtx</code> v2.2b</p> <p>General: Add a last-minute hook for <code>expl3</code> 1338</p> <p>2020-07-27 <code>ltmath.dtx</code> v1.2g</p> <p><code>\cases</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 893</p> <p><code>\matrix</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 893</p> <p><code>\pmatrix</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 893</p> <p>2020-07-27 <code>ltoutenc.dtx</code> v2.0r</p> <p><code>\@use@text@encoding</code>: Don't make the command <code>\long</code> (gh/354) . . 519</p> <p>2020-07-27 <code>ltpage.dtx</code> v1.0m</p> <p><code>\markright</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 1090</p> <p>2020-07-27 <code>ltpictur.dtx</code> v1.1p</p> <p><code>\linethickness</code>: Don't make the command <code>\long</code> (gh/354) 979</p> <p>2020-07-27 <code>ltsect.dtx</code> v1.1e</p> <p><code>\author</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 1012</p> <p><code>\date</code>: Don't make the command</p> <p><code>\long</code> (gh/354) 1012</p> <p>2020-08-01 <code>ltluatex.dtx</code> v1.1p</p> <p>General: new <code>_graf</code> is <code>exclusive</code> 61</p>	<p>2020-08-02 <code>ltluatex.dtx</code> v1.1q</p> <p><code>\newattribute</code>: Move reset to 0 inside conditional 47</p> <p><code>\newluabytecode</code>: Move reset to 0 inside conditional 50</p> <p><code>\newluachunkname</code>: Move reset to 0 inside conditional 50</p> <p><code>\newluafunction</code>: Move reset to 0 inside conditional 49</p> <p><code>\newwhatsit</code>: Move reset to 0 inside conditional 50</p> <p>2020-08-08 <code>ltclass.dtx</code> v1.3m</p> <p><code>\endfilecontents</code>: define <code>\q@curr@file</code> directly as the quotes have already been removed (gh/220) 1132</p> <p>2020-08-10 <code>ltluatex.dtx</code> v1.1r</p> <p>General: Load <code>ltluatex</code> Lua module during format building 51</p> <p>2020-08-15 <code>ltpictur.dtx</code> v1.2a</p> <p><code>\@defaultunitsset</code>: Macro added . . 976</p> <p>2020-08-19 <code>ltdefns.dtx</code> v1.5k</p> <p><code>\@carcube</code>: Made <code>\long</code> for</p> <p><code>\NewCommandCopy</code> 83</p> <p><code>\robust@command@act</code>: Made</p> <p><code>\robust@command@act</code> (was <code>\declare@command@copy</code>) more generic 99</p> <p><code>\ShowCommand</code>: Added <code>\ShowCommand</code> (gh/373) 103</p> <p>2020-08-19 <code>ltexpl.dtx</code> v1.2e</p> <p>General: Add</p> <p><code>\@expl@cs@<thing>@spec@@N</code> for <code>\ShowCommand</code> (gh/373) 76</p> <p>Add <code>\@expl@cs@to@str@@N</code> and <code>\@expl@str@if@eq@@nnTF</code> for <code>\NewCommandCopy</code> (gh/239) 76</p> <p>2020-08-20 <code>ltplain.dtx</code> v2.3d</p> <p><code>\alloc@</code>: Define <code>\alloc@</code> in terms of <code>\e@alloc</code> 20</p> <p>2020-08-21 <code>ltclass.dtx</code> v1.3o</p> <p>General: Integration of new hook management interface 1094</p> <p>2020-08-21 <code>ltdefns.dtx</code> v1.5l</p> <p>General: Integration of new hook management interface 80</p> <p>2020-08-21 <code>ltdefns.dtx</code> v1.5m</p> <p><code>\MakeRobust</code>: Make <code>\MakeRobust</code> produce the same command structure as</p> <p><code>\DeclareRobustCommand</code> 95</p> <p>2020-08-21 <code>ltexpl.dtx</code> v1.2d</p> <p>General: Dropped unused command . . 72</p>
---	---

2020-08-21 ltfiles.dtx v1.2i	2020-09-30 ltffssini.dtx v3.2d
General: Integration of new hook management interface 485	<code>\maybeupdate@bfseries@defaults:</code> <code>\bfdefault@previous</code> not <code>\bfseries@previous</code> (gh/395) .. 746 <code>\mdseries:</code> <code>\mddefault@previous</code> not <code>\mdseries@previous</code> (gh/395) .. 746
2020-08-21 ltfinal.dtx v2.2i	2020-10-01 ltclass.dtx v1.3r
General: Integration of new hook management interface 1332	<code>\providepackage:</code> Allow for package substitution 1109
2020-08-21 ltffssaxes.dtx v1.0g	2020-10-01 ltsect.dtx v1.1e
General: Integration of new hook management interface 669	<code>\addcontentsline:</code> add a fourth argument for better hyperref compatibility 1021
2020-08-21 ltffssini.dtx v3.2b	2020-10-04 ltfiles.dtx v1.2j
<code>\bf@def@ult:</code> Integration of new hook management interface 749	<code>\@include:</code> Quotes around the aux file name removed, they are not needed and upset BibTeX (gh/400) 495
<code>mdseries/defaults:</code> Integration of new hook management interface 751	2020-10-04 lthooks.dtx v1.0d
<code>\maybeupdate@bfseries@defaults:</code> Integration of new hook management interface 746	General: Definition <code>\AddToHookNext</code> was supposed to be for <code>\AddToHook</code> vice versa (gh/401) 325
<code>\maybeupdate@mdseries@defaults:</code> Integration of new hook management interface 747	2020-10-08 ltclass.dtx v1.3s
<code>\reset@font:</code> Integration of new hook management interface 761	<code>\@currnamestack:</code> Added missing 2020/02/02 <code>\IncludeInRelease</code> 1100
<code>\rmfamily:</code> Integration of new hook management interface 751	2020-10-11 ltclass.dtx v1.3t
<code>\ttfamily:</code> Integration of new hook management interface 751	<code>\load@onefilewithoptions:</code> Restore <code>\@currpkg@reqd</code> after finished loading a package file (gh/408). 1124
2020-08-21 ltmiscen.dtx v1.1v	2020-10-18 ltclass.dtx v1.3t
General: Integration of new hook management interface 864	<code>\PassOptionsToClass:</code> Drop path from <code>\input@path</code> (gh/414). . . 1111
2020-08-21 ltoutput.dtx v1.4f	2020-10-23 ltmiscen.dtx v1.1w
<code>\@begindvibox:</code> Integration of new hook management interface . . . 1225	<code>\enddocument:</code> Make <code>enddocument/afteraux</code> one-time 866
2020-08-23 ltxref.dtx v1.1o	2020-10-26 ltmiscen.dtx v1.1x
<code>\refstepcounter:</code> add default definition of <code>\@currentcounter</code> . 845	<code>\@kernel@before@enddocument:</code> <code>\enddocument</code> should always start out in vmode (gh/385) 869
2020-09-06 ltclass.dtx v1.3q	2020-11-09 ltclass.dtx v1.3u
<code>\load@onefilewithoptions:</code> Save <code>\@currpkg@reqd</code> so that we don't lose track of package substitutions. 1124	<code>\pkgcls@rollbackdate@error:</code> Change help text because the package may have existed then — there is just no rollback data (gh/423). 1146
2020-09-06 ltdefs.dtx v1.5n	2020-11-09 ltmath.dtx v1.2h
<code>\char@if@alph:</code> Macro added 111	<code>\negthickspace:</code> <code>\negmedspace</code> and <code>\negthickspace</code> have been only in amsmath, so we need to undefine for rollback (gh/423) 896
2020-09-06 ltexpl.dtx v1.2f	2020-11-20 ltclass.dtx v1.3u
General: Add <code>\@expl@str@map@function@NN</code> and <code>\@expl@char@generate@NN</code> for <code>\string@makeletter</code> (gh/386) 76	<code>\@currpath:</code> Macro added 1099
2020-09-09 ltshipout.dtx v1.0b	<code>\@kernel@currpathstack:</code> Macro added 1103
<code>__shipout_picture_overlay:n:</code> Prevent overfull box warnings (gh/387) 1192	
2020-09-26 ltfinal.dtx v2.2j	
General: Load first aid file if existing 1353	

<code>\load@onefile@withoptions</code> : Copy option list to the requested package.	1125	<code>\fontseries</code> : Distangle series and shape update (gh/444)	656
<code>\PassOptionsToClass</code> : Copy option list to the requested package. . .	1111	<code>\fontshape</code> : Distangle series and shape update (gh/444)	667
<code>\ProvidesPackage</code> : Use string comparison instead of <code>\ifx</code> . . .	1108	<code>\fontshapeforce</code> : Distangle series and shape update (gh/444)	667
2020-11-20 ltcmd.dtx v1.0a		2020-12-04 ltfssini.dtx v3.2f	
General: Initial version derived from <code>xparse.dtx</code>	115	General: Adjust start values for series and shape (gh/444)	764
2020-11-20 ltfilehook.dtx v1.0d		2020-12-10 ltbibl.dtx v1.1s	
<code>\unqu@tefilef@und</code> : Move loading to <code>\@input@file@exists@with@hooks</code> and expand <code>\@filef@und</code> to avoid getting the wrong file name in the case of a substitution.	1169	<code>\nocite</code> : Delay any <code>\nocite</code> in the preamble instead of raising an error	1056
2020-11-23 ltshipout.dtx v1.0d		2020-12-10 ltfssbas.dtx v3.2h	
<code>__shipout_execute_cont::</code> Check for both kernel and user hook (gh/431)	1195	<code>\usefont</code> : Drop “m” if the series value is a member of a fixed list and issue warning if doing it (gh/453) .	582
<code>__shipout_execute_main_cont:Nmn</code> : Check for both kernel and user hook (gh/431)	1198	2020-12-14 ltclass.dtx v1.3v	
2020-11-24 ltexpl.dtx v1.2g		<code>\@currnamestack</code> : Removed <code>\@expl@@hook@curr@name@push@@n</code>	1100
General: Support for roll forward (gh/434)	74, 76	2020-12-18 ltexpl.dtx v1.2h	
2020-11-24 ltfilehook.dtx v1.0d		<code>\@kernel@after@enddocument@afterlastpage</code> : Define kernel <code>\enddocument</code> hooks early	72
General: Support for roll forward (gh/434)	1167	2020-12-22 ltfssaxes.dtx v1.0h	
2020-11-24 lthooks.dtx v1.0f		<code>\delayed@merge@font@series</code> : Distangle series and shape update (gh/444)	658, 659
<code>__hook_end_document_label_check::</code> Support for roll forward (gh/434) .	260	<code>\delayed@merge@font@shape</code> : Distangle series and shape update (gh/444)	668
2020-11-24 ltshipout.dtx v1.0d		2020-12-22 ltfssstrc.dtx v3.0n	
General: Support for roll forward (gh/434)	1210	<code>\selectfont</code> : Execute delayed series and shape updates (gh/444) . . .	675
<code>\AtBeginDvi</code> : Support for roll forward (gh/434)	1209	2021-01-07 ltfilehook.dtx v1.0e	
2020-11-25 ltdefns.dtx v1.5o		General: Added rollback for this case to avoid spurious errors (part of gh/463)	1181
<code>\@carcube</code> : Added missing <code>latexrelease</code> entry	83	<code>\unqu@tefilef@und</code> : Restore <code>\CurrentFile(Path)(Used)</code> after the input (gh/464)	1170
2020-12-02 ltluatex.dtx v1.1s		2021-01-07 lthooks.dtx v1.0h	
General: Fix return value of list callbacks	64	<code>__hook_strip_double_slash:w</code> : Assume hook name has at least three nonempty parts (gh/464) .	272
2020-12-03 ltfssstrc.dtx v3.0m		<code>__hook_tl_set:cn</code> : Manually define some <code>l3tl</code> commands to work around <code>expl3</code> changes	247
<code>\selectfont</code> : Install a hook in <code>\selectfont</code> (gh/444)	676	2021-01-08 ltshipout.dtx v1.0f	
2020-12-04 ltfilehook.dtx v1.0d		<code>__shipout_execute_cont::</code> Added another kernel hook for more flexibility (cf.	
<code>\undeclare@file@substitution</code> : Don’t drop file substitution commands on rollback	1172		
2020-12-04 ltfssaxes.dtx v1.0h			
General: Reorganized the rollback data	601		

https://github.com/pgf-tikz/pgf/issues/1960	2019-10-10 ltfloat.dtx v1.2e	
..... 1195	\@footnotetext: Explicitly run \par at the end of footnote text in preparation for paragraph hooks	1047
2021-01-10 ltshipout.dtx v1.0g		
\@kernel@after@shipout@background:		
Internal hook		
\@kernel@after@shipout@background	\document@select@group: fix for (gh/501)	708
added 1199		
\RawShipout: Macro added 1199	2021-02-16 ltfloat.dtx v1.2f	
2021-01-19 ltshipout.dtx v1.0h	\footref: \footref added 1049	
__shipout_run_firstpage_hook::	2021-02-17 ltoutenc.dtx v2.0t	
Handling of firstpage hook	General: Adjust values for	
altered 1200	\textasteriskcentered To match TS1 definition (gh/502)	547
2021-01-21 ltclass.dtx v1.3w	Special definition for	
\@kernel@currrpathstack: Add empty entry for latexrelease 1103	\textasteriskcentered when missing in TS1 (gh/502)	538
2021-01-21 ltexpl.dtx v1.3a	2021-02-18 ltclass.dtx v1.3x	
General: Move xparse rollback code to ltcmd.dtx 75	\@fileswithoptions: save raw class option list (gh/85)	1120
2021-01-21 ltfinal.dtx v2.2l	\@remove@eq@value: macro added (gh/85)	1112
General: Load glyphtounicode.tex for pdfTeX 1340	\@use@option: value from unused option list (gh/85)	1115
2021-01-22 ltshipout.dtx v1.0i	\OptionNotUsed: value from unused option list (gh/85)	1112
__shipout_finalize_box:: Add pre_shipout_filter Lua callback 1194	\PassOptionsToClass: save raw option lists (gh/85)	1111
2021-01-24 ltexpl.dtx v1.3a	2021-02-19 ltoutenc.dtx v2.0u	
General: Define expl3 hooks conditionally 72	General: Add	
2021-01-31 ltfilehook.dtx v1.0f	\textnonbreakinghyphen,	
\@curr@file@reqd: set \protect to \string gh/481 1173	\textfiguredash and	
2021-02-03 ltfloat.dtx v1.2e	\texthorizontalbar	
\@savemarbox: Explicitly end with \par (gh/489)	(gh/404)	526, 530, 546
..... 1039	2021-02-25 ltfinal.dtx v2.2m	
2021-02-04 ltboxes.dtx v1.4b	General: Improve speed of ToUnicode everyjob loading code 1340	
\color@endbox: Always add the color groups (gh/488)	2021-03-03 ltclass.dtx v1.3y	
..... 929	\endfilecontents: Fix overwrite check for files with UTF-8 (gh/415)	1133
2021-02-08 ltfilehook.dtx v1.0g	2021-03-05 ltclass.dtx v1.3z	
\unqu@tefilef@und: Undo the internal for robust \InputIfFileExists in rollback (gh/494)	\ProcessOptions*: modify so braces to not give errors (gh/513)	1114
..... 1171	2021-03-12 ltfiles.dtx v1.2k	
2021-02-08 ltmiscen.dtx v1.1y	\IfFileExists@: Allow unbalanced conditionals (gh/530)	500
\begin: Undo the internals for robust \begin and \end in rollback (gh/494)	2021-03-18 ltcmd.dtx v1.0b	
..... 874	General: Use \NewModuleRelease.	129
\end\verbvisiblespace: Undo the internals for robust \begin and \end in rollback (gh/494)	2021-03-18 ltfilehook.dtx v1.0h	
..... 877	__filehook_file_pop_assign:nnnn: Define	
2021-02-10 ltboxes.dtx v1.4b	\g__filehook_input_file_seq to avoid losing data when rolling back.	1167
\@mpfootnotetext: Explicitly run \par in support for paragraph tagging 941		

2021-03-18 ltfsaxes.dtx v1.0i	2021-04-20 ltexpl.dtx v1.3c
General: Fix rollforward definition. 657	\@kernel@after@enddocument@afterlastpage: Don't empty kernel hooks on rollback 72
2021-03-18 ltfsini.dtx v3.2g	2021-04-20 ltfilehook.dtx v1.0i
General: Add legacy hook definitions for rollback. 752	\@curr@file@reqd: Make expand to a string (tracks change in l3kernel) 1173
2021-03-18 lthooks.dtx v1.0i	2021-04-26 ltfsbas.dtx v3.2i
_hook_end_document_label_check:: Only add top-level if not already there. 260	\usefont: Unconditionally switch to the requested font face (gh/444) 582
Remove the (empty) "top-level" from \@currnamestack. 260	2021-04-26 ltfsini.dtx v3.2h
General: Use \NewModuleRelease. 245	\reset@font: Unconditionally switch to the requested font face (gh/444) 761
2021-03-18 ltvers.dtx v1.1f	2021-04-26 ltfsstrc.dtx v3.0o
\@check@IncludeInRelease: Add support for usage in \NewModuleRelease 37	\selectfont: Unset the forced series boolean when reaching \selectfont (gh/444) 676
\new@moduledate: Added \NewModuleRelease. 38	2021-04-29 lthooks.dtx v1.0m
2021-03-19 lttextcomp.dtx v1.0e	\ActivateGenericHook: Add \ProvideHook etc. 320
General: Use \NewModuleRelease 799	2021-04-29 ltoutenc.dtx v2.0v
2021-03-26 ltplain.dtx v2.3e	General: Add composites for \ae/\AE/\ae/\AE (gh/552) 551
\@unused: Allocate \@inputcheck and \@unused early so that they are before expl3 allocates more streams (gh/538) 21	2021-05-18 ltclass.dtx v1.4b
2021-03-27 ltclass.dtx v1.4a	\@raw@classoptionslist: Initialize to \relax to match \@classoptionslist 1099
\@currnamestack: Do not completely roll back if expl3 is loaded. 1101, 1102	2021-05-24 ltcmd.dtx v1.0e
2021-04-16 ltvers.dtx v1.1g	General: Use \msg_... instead of _kernel_msg... 129
\new@moduledate: \NewModuleRelease with the same arguments as \IncludeInRelease. 38	2021-05-24 ltcmdhooks.dtx v1.0b
2021-04-17 ltfiles.dtx v1.2m	General: Use \msg_... instead of _kernel_msg... 332
\@kernel@after@begindocument: Move \@kernel@before@begindocument and \@kernel@after@begindocument init earlier so that other modules can write to the hooks 490	2021-05-24 ltfilehook.dtx v1.0k
2021-04-18 ltluatex.dtx v1.1t	General: Use \msg_... instead of _kernel_msg... 1165
General: input_level_string added 62	2021-05-24 lthooks.dtx v1.0n
2021-04-18 ltplain.dtx v2.3f	General: Use \msg_... instead of _kernel_msg... 245
\loggingall: 3 31	2021-05-24 ltpara.dtx v1.0g
Drop pre- ϵ -TeX support 31	General: Use \msg_... instead of _kernel_msg... 448
\tracingnone: 3 32	2021-05-26 ltdefns.dtx v1.5p
Drop pre- ϵ -TeX support 32	\MakeRobust: Normalize error message in \MakeRobust 95
2021-04-19 ltcmd.dtx v1.0d	2021-06-03 ltclass.dtx v1.4c
_cmd_cmd_type_cases:NnnnnnF: Renamed _cmd_cmd_if_xparse:NTF to _kernel_cmd_if_xparse:NTF for cross-module usage 203	\@kernel@currpathstack: Take care of \@kernel@currpathstack when rolling back/forward. 1103
	2021-06-04 ltcmd.dtx v1.0f
	General: Normalize various error messages 206

2021-06-05 ltmiscen.dtx v1.1z	2021-07-20 ltcmdhooks.dtx v1.0c
\@sverb: Normalize error message . 885	__hook_patch_DeclareRobustCommand:Nnn: Use \robust@command@chk@safe before \@if@newcommand. 337
2021-06-06 ltclass.dtx v1.4c	2021-07-22 lthooks.dtx v1.0o
\@loadwithoptions: handle raw options for gh/580 1117	__hook_gremove_code:nn: Do not queue removals (gh/625) 274
\load@onefile@withoptions: Copy raw options for gh/580 1125	__hook_hash_check_aux:w: Do not queue removals (gh/625) 262
\PassOptionsToClass: apply \expandafter to raw options for gh/580 1111	2021-07-23 ltclass.dtx v1.4e
2021-06-09 ltclass.dtx v1.4b	\load@onefile@withoptions: Make class/name/after a one-time hook 1126
\endfilecontents: Use \@latex@note@no@line to display the information 1132, 1133	Make class/name/before a one-time hook 1126
2021-06-09 lterror.dtx v1.2r	Make package/name/after a one-time hook 1126
\@latex@note@no@line: Macros added 432	Make package/name/before a one-time hook 1126
2021-06-09 ltfssbas.dtx v3.2j	2021-07-23 ltfiles.dtx v1.2n
\DeclareFontShape@: Improve information message 572	\@include: Make include/name/after a one-time hook 496
2021-07-08 ltcounts.dtx v1.1m	Make include/name/before a one-time hook 496
\counterwithin: New implementation for \counterwithin 563	Make include/name/end a one-time hook 496
\counterwithout: New implementation for \counterwithout 564	2021-07-27 lthooks.dtx v1.0o
2021-07-11 lterror.dtx v1.2s	\ClearHookNext: Macro added 321
\PackageNoteNoLine: Provide \ClassNote and \PackageNote . 431	\hook_gc_clear_next_code:n: Macro made public 307
2021-07-12 ltclass.dtx v1.4d	2021-07-28 ltsect.dtx v1.1f
\@fileswithoptions: add \unexpanded 1121	\contentsline: Pick up four arguments (gh/633) 1023
2021-07-16 ltplain.dtx v2.3g	2021-07-30 ltcmd.dtx v1.0d
General: Use 2 as default value for \tracinglostchars 22	__cmd_cmd_type_cases:NnnnnnF: Added __cmd_cmd_type_cases:NnnnnnF for \NewCommandCopy and \ShowCommand support 203
2021-07-19 ltclass.dtx v1.4e	2021-07-31 ltoutput.dtx v1.4e
\@classoptionslist: Drop \@onlypreamble 1099	\fl@tracemessage: Enable display when doing \tracefloatvals . 1292
\@ifclasslater: Drop \@onlypreamble 1104	2021-07-31 ltoutput.dtx v1.4g
\@ifclassloaded: Drop \@onlypreamble 1104	\ShowFloat: Macro added 1290
\@ifclasswith: Drop \@onlypreamble 1106	2021-08-02 lthooks.dtx v1.0o
\@ifl@ter: Drop \@onlypreamble 1105	\ActivateGenericHook: Change name 320
\@pkgextension: Drop \@onlypreamble 1100	\DisableGenericHook: Change name 320
\@optionlist: Drop \@onlypreamble 1104	2021-08-07 ltcmd.dtx v1.0g
\@unusedoptionlist: Drop \@onlypreamble 1099	__cmd_add_grabber:N: Replicate argument processors for all embellishments (gh/639) 157
\IfFormatAtLeastTF: Drop \@onlypreamble 1104	

<code>__cmd_add_type_E:w</code> : Replicate argument processors for all embellishments (gh/639) 155	2021-08-27 lthooks.dtx v1.0q General: Internal message name changes 317
2021-08-08 ltfinal.dtx v2.2p <code>\IfPDFManagementActiveTF</code> : Default definition added (gh/640) 1353	2021-08-27 ltpara.dtx v1.0i General: Internal message name changes 456
2021-08-10 ltvers.dtx v1.1h <code>\@check@IncludeInRelease</code> : Add error to aid debugging 37	2021-08-30 ltcmd.dtx v1.0h General: Added support for <code>\NewCommandCopy</code> 162 Added support for <code>\ShowCommand</code> 168
2021-08-11 ltuatex.dtx v1.1u General: Define missing local function 52	2021-09-03 ltoutput.dtx v1.4h <code>\ShowFloat</code> : Renamed, original name never distributed 1290
2021-08-20 lterror.dtx v1.2t <code>\@badend</code> : Improve <code>\@badend</code> error message (gh/587) 435	2021-09-06 ltfinal.dtx v2.2q <code>\@uclclist</code> : Correctly upper and lowercase <code>\ij</code> and <code>\IJ</code> (gh/658) 1350
2021-08-20 lthooks.dtx v1.0p General: Added deprecation warnings for old generic hook commands (gh/638) 323 Documentation updates for generic hook commands (gh/638) 218 Renames of generic hook commands (gh/638) 256 Section on generic hooks added (gh/638) 235	2021-09-06 lthooks.dtx v1.0r <code>__hook_hash_check_aux:w</code> : Use dedicated conditional (gh/606) 262 <code>__hook_if_execute_immediately:n</code> : Macro added (gh/606) 312 <code>__hook_use_once_clear:n</code> : Clean up after <code>\UseOneTimeHook</code> (gh/606) 312 <code>\hook_use_once:nnw</code> : Clean up after <code>\UseOneTimeHook</code> (gh/606) 311
2021-08-25 ltclass.dtx v1.4f General: Standardize generic hook names (gh/648) 1094 <code>\load@onefile@withoptions</code> : Declare non-generic package and class hooks 1127	2021-09-10 ltfssini.dtx v3.2i <code>\bfseries</code> : Do delayed changes to <code>\bfdefault</code> in a separate macro for better reuse (gh/663) 746 <code>\DeclareFontSeriesDefault</code> : Do delayed changes to <code>\bfdefault</code> or <code>\mddefault</code> first (gh/663) 737 <code>\maybe@update@bfseries@defaults</code> : Do delayed changes to <code>\bfdefault</code> in a separate macro for better reuse (gh/663) 746 <code>\maybe@update@mdseries@defaults</code> : Do delayed changes to <code>\mddefault</code> in a separate macro for better reuse (gh/663) 747
2021-08-25 ltcmdhooks.dtx v1.0d <code>__hook_try_put_cmd_hook:w</code> : Simplify generic hook detection 334	<code>\mdseries</code> : Do delayed changes to <code>\mddefault</code> in a separate macro for better reuse (gh/663) 746
2021-08-25 ltfilehook.dtx v1.0l <code>\unqu@tefilef@und</code> : Declare non-generic file hooks 1170	2021-09-12 ltfontcmd.dtx v3.5a <code>\check@nocorr@</code> : use <code>\unexpanded</code> to make <code>#</code> safe 794
2021-08-25 ltfiles.dtx v1.2o <code>\@include</code> : Declare non-generic include hooks 496 Standardize generic hook names (gh/648) 496	2021-09-12 ltoutenc.dtx v2.0w General: Move zero skip between <code>i</code> and <code>j</code> for hyphenation (gh/658) 527
2021-08-25 lthooks.dtx v1.0p <code>__hook_try_declaring_generic_next_hook:nn</code> : Standardize generic hook names (gh/648) 267	2021-09-18 ltpara.dtx v1.0i <code>\para_end::</code> : Use skip rather than kern as guard. 454
2021-08-27 ltcmd.dtx v1.0h <code>\NewDocumentEnvironment</code> : Check for end-of-environment command 212	
2021-08-27 ltfilehook.dtx v1.0l <code>__filehook_file_pop_assign:nnnn</code> : Internal message name changes 1167 <code>__filehook_file_subst_cycle_error:cN</code> : Use <code>\msg_...</code> not <code>__kernel_msg_...</code> 1178	

2021-09-26 ltfsdcl.dtx v3.0x	expansion+redefinition when the macro contains a parameter token (gh/697)	338
\c@localmathalphabets: Counter added for (gh/676)	707	
\document@select@group: Test if we should freeze the version (gh/676)	707	
\freeze@math@version: Macro added for (gh/676)	710	
2021-09-28 ltcmdhooks.dtx v1.0e		
__hook_make_prefixes:w: Make patching of commands a global operation (gh/674)	342	
__hook_patch_retokenize:Nnnn: Make patching of commands a global operation (gh/674)	351	
2021-09-28 lthooks.dtx v1.0s		
__hook_if_usable_use:n: Correct usage of older		
__hook_if_file_hook:wTF (gh/675)	310	
__hook_try_declaring_generic_hook_split:Nnnn: Correct usage of older		
__hook_if_file_hook:wTF (gh/675)	268	
2021-10-14 ltboxes.dtx v1.4c		
\@mpfootnotetext: Explicitly set		
\@currentcounter (gh/687) . . .	941	
2021-10-14 ltfiles.dtx v1.2p		
\@include: Warn about use in preamble	495	
2021-10-14 ltfloat.dtx v1.2g		
\@footnotetext: Explicitly set		
\@currentcounter (gh/687) . .	1047	
2021-10-14 ltmath.dtx v1.2j		
\@eqnset: Explicitly set		
\@currentcounter (gh/687) . . .	900	
eqnarray: Explicitly set		
\@currentcounter (gh/687) . . .	906	
2021-10-15 ltfsdcl.dtx v3.0y		
\DeclareMathVersion: Initialize variable for freezing math version (gh/676)	715	
2021-10-15 ltluatex.dtx v1.1v		
General: provide_charproc_data added	62	
2021-10-16 ltoutenc.dtx v2.0x		
\add@accent: Dont set		
\spacefactor in math mode gh/643	516	
2021-10-19 ltpara.dtx v1.0k		
General: Remove content from		
\tex_everypar:D on rollback . . .	456	
2021-10-20 ltcmdhooks.dtx v1.0f		
__hook_make_prefixes:w: Correct patching by		
2021-11-17 ltluatex.dtx v1.1w		
General: hpack_quality is exclusive	61	
Never pass on true return values for list callbacks	64	
vpack_quality is exclusive	61	
2021-11-30 ltclass.dtx v1.5a		
\@onefilewithoptions@clashchk: Separated out from		
\@onefilewithoptions	1125	
2021-11-30 ltexpl.dtx v1.3d		
\skipeval: Moved over from xfp package (gh/711)	77	
2021-12-02 ltcmd.dtx v1.0i		
__cmd_run_code:: Correct defaults for optional arguments in end-of-environment code (gh/712)	140	
__cmd_start_aux:ccnnnn: Correct defaults for optional arguments in end-of-environment code (gh/712)	140	
2021-12-07 ltexpl.dtx v1.3d		
\skipeval: Added \dimeval and \skipeval (gh/711)	77	
2021-12-11 ltdirchk.dtx v1.3a		
General: Add comment lines into latex.ltx to indicate temp definitions that are later overwritten (gh/725)	2	
2021-12-12 ltoutenc.dtx v2.0y		
General: \DeclareUnicodeAccent now makes the encoding argument implicit (gh/253)	543	
Added \DeclareUnicodeCommand and \DeclareUnicodeSymbol (gh/253)	544	
2021-12-27 ltluatex.dtx v1.1x		
\newprotectedluacmd: Macros added	49	
2021-12-28 ltexpl.dtx v1.3d		
\ExpandArgs: Added document level names for \exp_args:Nc and the like (gh/735)	77	
2021-13-31 ltcmd.dtx v1.0j		
__cmd_show:e: Make \ShowCommand stop for interaction	169, 170	
2022-01-06 ltexpl.dtx v1.3e		
\ExpandArgs: Adjust document level names for \exp_args:Nc and the like (gh/735)	77	
2022-01-06 ltshipout.dtx v1.0l		
\AtBeginShipoutNext: Correctly simulate \AtBeginShipout and		

<code>\AtBeginShipoutNext</code> without extending the syntax	1212	2022-02-28 <code>ltexpl.dtx</code> v1.3f General: Move <code>latexrelease</code> redefinitions from <code>ltxcmd.dtx</code>	75
<code>\AtEndDvi</code> : Correctly simulate <code>\AtEndDvi</code> without extending the syntax	1211	2022-02-28 <code>ltvers.dtx</code> v1.1i <code>\new@moduledate</code> : Detect a missing <code>\IncludeInRelease{0000/00/00}</code>	38
2022-01-15 <code>ltkeys.dtx</code> v1.0b <code>_keys_options_aux:n</code> : Clear option list in end-of-package hook . . .	1153	2022-03-10 <code>ltbibl.dtx</code> v1.1t <code>\cite</code> : Ensure that an empty argument generates a warning (gh/790)	1054
2022-01-25 <code>ltplain.dtx</code> v2.3h <code>\obeyspaces</code> : Provide redirection to support special use cases (gh/367) . .	25	2022-03-10 <code>ltfilehook.dtx</code> v1.0m <code>\@curr@file@reqd</code> : Add <code>\set@curr@file@nosearch</code> for <code>graphicx</code>	1173
2022-02-05 <code>ltkeys.dtx</code> v1.0b .usage: Create properties in <code>ltkeys</code> . . .	1151	2022-03-18 <code>ltclass.dtx</code> v1.5b <code>\load@onefilewithoptions</code> : Switch to <code>\ProcessKeyOptions</code>	1123
2022-02-07 <code>ltkeys.dtx</code> v1.0c .usage: Correct <code>..code</code> property . . .	1151	2022-03-18 <code>ltxcmd.dtx</code> v1.0l <code>_cmd_cmd_type_cases:NnnnnF</code> : Fix <code>_cmd_cmd_type_cases:NnnnnF</code> prematurely expanding macros (gh/795)	203
2022-02-15 <code>ltkeys.dtx</code> v1.0c <code>\DeclareKeys</code> : Expand module argument	1157	2022-03-18 <code>ltkeys.dtx</code> v1.0f <code>_keys_options:n</code> : Simplify to always cover global options . . .	1153
<code>\DeclareUnknownKeyHandler</code> : Added <code>\DeclareUnknownKeysHandler</code>	1157	<code>_keys_options_global:n</code> : Simplify to always cover global options . .	1153
<code>\ProcessKeyOptions</code> : Expand module argument	1157	<code>\ProcessKeyOptions</code> : Remove <code>\ProcessKeyPackageOptions</code> . .	1157
<code>\SetKeys</code> : Expand module argument	1158	2022-04-01 <code>ltfiles.dtx</code> v1.2q <code>\@include</code> : Process some hooks is an include file is bypassed	496
2022-02-16 <code>ltkeys.dtx</code> v1.0d <code>\DeclareKeys</code> : Allow for active characters in module argument	1157	2022-04-01 <code>lthooks.dtx</code> v1.0t <code>\c__hook_generic_include/./end_tl</code> : Support generic <code>include/.../excluded</code> hooks . . .	273
<code>\DeclareUnknownKeyHandler</code> : Allow for active characters in module argument	1157	2022-04-03 <code>ltfinal.dtx</code> v2.2s General: Integration of new mark management interface	1332
Rename <code>\DeclareUnknownKeysHandler</code> to <code>\DeclareUnknownKeyHandler</code>	1157	2022-04-03 <code>ltoutput.dtx</code> v1.1h <code>\@opcol</code> : Interface with new mark mechanism	1241
<code>\ProcessKeyOptions</code> : Allow for active characters in module argument	1157	2022-04-04 <code>ltpage.dtx</code> v1.0n <code>\markright</code> : Interface with new mark mechanism	1090
<code>\SetKeys</code> : Allow for active characters in module argument	1158	2022-04-08 <code>ltmath.dtx</code> v1.2k <code>\openup</code> : Make <code>\protected</code> (gh/123)	894
2022-02-19 <code>ltxcmd.dtx</code> v1.0k <code>\IfBlankTF</code> : Added <code>\IfBlankTF</code> and friends	216	2022-04-12 <code>ltxref.dtx</code> LaTeX2e <code>\pageref</code> : Macro reimplemented with a starred version	842
2022-02-20 <code>ltfinal.dtx</code> v2.2r <code>\@uclclist</code> : Use <code>\@expl@text@uppercase@n</code> , removing local redefinition of <code>\UTF@two@octets@noexpand</code> . . .	1347	<code>\ref</code> : Macro reimplemented with a starred version	842
use <code>\text_lowercase:n</code>	1347		
2022-02-21 <code>ltkeys.dtx</code> v1.0e <code>_keys_options_expand_module:Nn</code> : Faster approach to module expansion	1156		
2022-02-28 <code>ltxcmd.dtx</code> v1.0k General: Move <code>latexrelease</code> redefinitions from <code>ltxcmd.dtx</code> . . .	216		

2022-04-12 ltxref.dtx v1.1p		<code>__keys_options_package:n</code> : Better handling of option removal . . .	1155
General: Add starred variants for the ref commands	840	2022-06-19 ltkeys.dtx v1.0h	
<code>\Ref</code> : Macro reimplemented with a starred version	847	<code>__keys_options_class:n</code> : Further work on handling of option removal	1154
2022-04-21 ltfinal.dtx v2.2t		<code>__keys_options_package:n</code> : Further work on handling of option removal	1155
<code>\@uclclist</code> : Support <code>\noexpand</code> in argument of		<code>__keys_options_package:nnn</code> : New function	1155
<code>\@expl@text@uppercase@cn</code> . . .	1347	2022-06-20 ltclass.dtx v1.5c	
2022-05-06 ltmarks.dtx v1.0c		<code>\load@onefilewithoptions</code> : Pass raw options to <code>\ProcessKeyOptions</code> . . .	1123
<code>__mark_new_class:nn</code> : Wrong command made <code>\@onlypreamble</code> . . .	1068	2022-06-20 ltkeys.dtx v1.0h	
2022-05-08 ltmath.dtx v1.2l		<code>__keys_options_class:n</code> : Use raw options data	1154
General: Use consistent math styles under LuaTeX	903	<code>__keys_options_class:nnn</code> : New function	1154
2022-05-08 ltshipout.dtx v1.0m		<code>__keys_options_global:n</code> : Use raw options data	1153
<code>\@kernel@after@enddocument@afterlastpage</code> :		<code>__keys_options_local::</code> Use raw options data	1155
Handle case where shipout/lastpage is run too early (gh/813)	1207	<code>__keys_options_package:n</code> : Use raw options data	1155
<code>__shipout_execute_main_cont:Nnnn</code> :		2022-06-22 ltkeys.dtx v1.0i	
Handle case where shipout/lastpage is run too early (gh/813)	1198	<code>.usage</code> : Add <code>.notif</code> property	1151
2022-05-13 lthooks.dtx v1.0u		2022-06-30 ltfinal.dtx v2.2u	
<code>__hook_use_once_clear:n</code> : Check if prop exists to avoid l3debug error . . .	312	<code>\@uclclist</code> : Add	
2022-05-17 lthooks.dtx v1.0u		<code>\AddToNoCaseChangeList</code>	1349
<code>__hook_initialize_hook_code:n</code> :		2022-06-30 ltfinal.dtx v2.2v	
Refuse sorting one-time hooks (gh/818).	289	<code>\@uclclist</code> : Just use	
2022-05-17 ltmeta.dtx v1.0b		<code>\text_lowercase:n</code> without	
General: Default definition for targets added	460	<code>\protectd@edf</code> gh/881x	1347
2022-05-27 ltfiles.dtx v1.2r		2022-07-04 ltfinal.dtx v2.2w	
<code>\listfiles</code> : Try saved version string, if ver@.. is <code>\relax</code> (gh/825) . . .	507	<code>\@uclclist</code> : Introduced <code>\CaseSwitch</code> , <code>\DeclareCaseChangeEquivalent</code> and <code>\MakeTitlecase</code> to support hooking into case changing gh/889	1347
2022-05-27 ltoutenc.dtx v2.0z		2022-07-04 ltfsbas.dtx v3.2k	
General: Save the version string (gh/825)	555	<code>\frozen@everydisplay</code> : Ignore spaces if necessary (gh/886)	586
2022-06-01 ltmarks.dtx v1.0d		<code>\frozen@everymath</code> : Ignore spaces if necessary (gh/886)	586
<code>__mark_prepare_and_extract:nn</code> :		2022-07-04 ltssdcl.dtx v3.0z	
Extend the logic for detecting the marks in the box (gh/836)	1071	<code>\freeze@math@version</code> : Ignore spaces if necessary (gh/886)	711
General: Marks are kernel errors . .	1081	2022-07-05 ltkeys.dtx v1.0i	
2022-06-02 ltfinal.dtx v2.2u		<code>__keys_options_aux:n</code> : Support <code>\CurrentOption</code>	1153
<code>\@uclclist</code> : Add <code>\NoCaseChange</code> . .	1349	<code>__keys_options_class:nnn</code> : Correct naming of raw class options storage	1154
2022-06-15 lthooks.dtx v1.0v			
<code>__hook_activate_generic:n</code> : Ensure that a newly activated generic hook gets its execution code set . .	257		
2022-06-16 ltkeys.dtx v1.0g			
<code>__keys_options_class:n</code> : Better handling of option removal . . .	1154		

2022-07-23 ltkeys.dtx v1.0j	<code>\freeze@math@version</code> : New logic for freezing math versions (gh/921) . 710
<code>__keys_options_aux:n</code> : Output ‘public’ package name in messages 1153	2022-09-20 ltfssdcl.dtx v3.1b
<code>__keys_options_loaded:nn</code> : Output ‘public’ package name in messages 1158	<code>\DeclareSymbolFont</code> : Drop any surplus ‘m’ in series argument (gh/918) 716
2022-08-07 ltfssbas.dtx v1.0g	<code>\SetSymbolFont</code> : Drop any surplus ‘m’ in series argument (gh/918) 718
<code>\DeclareEncodingSubset</code> : Make global declaration (gh/905) 577	2022-10-03 ltluatex.dtx v1.2a
2022-08-10 ltcmd.dtx v1.1a	General: Add rules for callback ordering 58
<code>__cmd_arg_to_keyvalue:nn</code> : New internal arg-to-keyval processor . 199	<code>add_to_callback</code> : Add rules for callback ordering 67
<code>__cmd_grab_D_long_obey_spaces_no_strip:w:declare_callback_rule</code> : Add support for skipping brace stripping 178	<code>remove_from_callback</code> : Add rules for callback ordering 69
<code>__cmd_grab_R_aux:NNnN</code> : Track changes in D-type implementation 185	2022-10-10 ltclass.dtx v1.5d
<code>__cmd_grab_b_end:Nw</code> : Track changes in D-type implementation 174	<code>\@unprocessedoptions</code> : Use <code>\protected@edef</code> 1130
General: New switch 132	<code>\load@onefilewithoptions</code> : Use <code>\protected@edef</code> 1123
2022-08-13 ltluatex.dtx v1.1y	<code>\PassOptionsToClass</code> : Use <code>\protected@xdef</code> 1111
General: shared_callbacks added . . 62	<code>\ProcessOptions*</code> : Use <code>\protected@edef</code> 1113
<code>add_to_callback</code> : Adapted code for shared_callbacks 67	2022-10-20 ltclass.dtx v1.5e
<code>remove_from_callback</code> : Adapted code for shared_callbacks 69	<code>\load@onefilewithoptions</code> : Define key option handler in ltkeys . . 1123
<code>mlist_to_hlist</code> : Use shared_callback system for pre/post/mlist_to_hlist 71	2022-10-20 ltkeys.dtx v1.0l
2022-08-18 ltfilehook.dtx v1.0n	<code>__keys_options_aux:n</code> : Define key option handler in ltkeys 1153
<code>\@disable@packageload@do</code> : Inhibit checking the loaded version when package is load-disabled, and write to the .log (gh/888) 1179	<code>__keys_options_loaded:nn</code> : Correct an argument for a message . . . 1157
2022-08-20 ltoutput.dtx v1.4j	2022-10-22 ltclass.dtx v1.5e
<code>\stockheight</code> : Register added . . . 1226	<code>\@fileswithoptions</code> : Use <code>\protected@xdef</code> 1120
<code>\stockwidth</code> : Register added 1226	<code>\@use@option</code> : Use <code>\detokenize</code> . . 1115
2022-08-21 ltkeys.dtx v1.0k	<code>\ProcessOptions*</code> : Use <code>\detokenize</code> 1114
<code>__keys_options_loaded:nn</code> : Correct error message 1158	2022-10-22 ltdefns.dtx v1.5r
2022-09-03 ltmath.dtx v1.2m	<code>\@expandtwoargs</code> : Use <code>\protected@edef</code> 91
<code>\smash</code> : Guard against reboxing in fractions (gh/517) 893	2022-10-22 ltkeys.dtx v1.0l
2022-09-07 ltboxes.dtx v1.4d	<code>__keys_options_class:nnn</code> : Correct handling of unused option list . 1154
<code>\@iiiminipage</code> : Check for nested minipages and warn (gh/168) . . 940	2022-10-26 ltfinal.dtx v2.2x
<code>\if@in@minipage@env</code> : Check for nested minipages and warn (gh/168) 939	<code>\@uclclist</code> : Auto-detect babel locale 1347
2022-09-17 ltfssdcl.dtx v3.1a	Introduce optional argument for case-changing commands 1347
<code>\DeclareMathVersion</code> : New logic for freezing math versions (gh/921) . 715	Make case changing commands language-aware 1347

2022-11-06 ltmiscen.dtx v1.2a		<code>__cmd_start_-</code>	
<code>\enddocument</code> : Repeat release info at		<code>expandable:nNNNNn</code>	164
the end (gh/944)	866	<code>__cmd_show:e</code> : Distinguish	
2022-11-06 ltvers.dtx v1.1j		(non-expandable) document	
General: Repeat release info at the		commands starting with <code>__cmd_-</code>	
end (gh/944)	36	<code>start_expandable:nNNNNn</code>	168
2022-11-08 ltshipout.dtx v1.0n		2023-03-22 ltspac.dtx v1.3p	
<code>__shipout_execute_main_cont:Nnnn</code> :		<code>\samepage</code> : Add <code>\predisplaypenalty</code>	
Add shipout hook (gh/920) . . .	1198	setting (gh/1022)	466
<code>shipout/lastpage</code> : Add shipout hook		2023-03-28 ltclass.dtx v1.5g	
(gh/920)	1199	<code>\IfFormatAtLeastTF</code> : Added	
2022-11-16 ltclass.dtx v1.5f		<code>\IfFileAtLeastTF</code> (gh/1015) . .	1104
<code>\endfilecontents</code> : Do not show		2023-03-28 ltfinal.dtx v2.2z	
"current dir" in message		<code>\@uclclist</code> : Use groups for gh/1021	1348
(gh/917)	1133	2023-04-01 ltffsbas.dtx v3.2l	
Introduce key 'nowarn' on		<code>\math@version</code> : Reset frozen	
filecontents (gh/958)	1131–1133	<code>mathversion</code> gh/1028	584
2022-11-24 ltdefs.dtx v1.5s		2023-04-02 ltfilehook.dtx v1.0o	
<code>\DeclareEnvironmentCopy</code> : Add		<code>\@set@curr@file@aux</code> : Make	
<code>\NewEnvironmentCopy</code> ,		<code>\@set@curr@file@aux \long</code>	
<code>\RenewEnvironmentCopy</code> , and		gh/942	1175
<code>\DeclareEnvironmentCopy</code>		2023-04-06 ltcmdhooks.dtx v1.0g	
(gh/963)	102	<code>__hook_double_hashes_space:w</code> :	
<code>\ShowEnvironment</code> : Added		Add case for <code>\c__hook_hashes_tl</code>	
<code>\ShowEnvironment</code>	107	(hook-args)	344
2022-11-28 ltspac.dtx v1.3o		<code>__hook_make_prefixes:w</code> : Rename to	
<code>\@hspace</code> : Support calc syntax		<code>\c__hook_hashes_tl</code> (hook-args) .	340
correctly (gh/967)	481	<code>\c__hook_hashes_tl</code> : Rename to	
<code>\@vspace@calcify</code> : Support calc		<code>\c__hook_hashes_tl</code> and add	
syntax without a group (gh/967)	468	<code>\c__hook_hash_tl</code> (hook-args) .	333
2022-11-29 ltcmd.dtx v1.1b		2023-04-06 lthooks.dtx v1.1a	
<code>__cmd_show:e</code> : Add		<code>__hook_activate_generic:n</code> :	
<code>\@showenvironmentlisthook</code> . .	171	Changes to add hook arguments	
2022-11-30 ltcmd.dtx v1.1b		(hook-args).	256
<code>__cmd_show:e</code> : Don't stop for the		<code>__hook_braced_real_loop:w</code> : Macro	
<code>\begin</code> part of an		added (hook-args).	282
environment	169, 170	<code>__hook_chk_args_allowed:nn</code> :	
2022-11-30 ltfinal.dtx v2.2y		Macro added (hook-args).	266
<code>\@uclclist</code> : Set <code>\oe/\OE</code> equal to act		<code>__hook_clear_next:n</code> : Changes to	
as a marker for <code>babel</code>	1348	add hook arguments (hook-args). .	307
2023-01-05 ltfiles.dtx v1.2s		<code>__hook_code_gset_aux:nnn</code> : Macro	
<code>\document</code> : <code>\do</code> now with default		added (hook-args).	277
definition in the kernel (gh/975)	489	<code>__hook_code_gset_auxi:eeen</code> :	
2023-01-19 ltluatex.dtx v1.2b		Macro added (hook-args).	276
General: Remove unused local variable		<code>__hook_cs_end:w</code> : Macro added	
<code>tex_setattribute</code>	52	(hook-args).	281
2023-01-30 ltpara.dtx v1.0l		<code>__hook_cs_if_empty:c</code> : Macro added	
<code>\g__para_standard_everypar_tl</code> :		(hook-args).	281
Backout <code>\parskip</code> at top of		<code>__hook_gput_next_do:nn</code> : Changes	
minipage (gh/989)	449	to add hook arguments	
2023-03-12 ltcmd.dtx v1.1c		(hook-args).	306
<code>__cmd_copy_expandable:NnNNNNnnn</code> :		<code>__hook_gremove_code:nn</code> : Changes	
Distinguish (non-expandable)		to add hook arguments	
document commands starting with		(hook-args).	274

<code>__hook_hash_check_aux:w</code> : Add	
<code>\hook_gput_code_with_args:nnn</code>	
(hook-args).	262
Changes to add hook arguments	
(hook-args).	262, 263
Macro added (hook-args).	264
<code>__hook_if_execute_immediately:n</code> :	
Changes to add hook arguments	
(hook-args).	312
<code>__hook_init_structure:n</code> : Changes	
to add hook arguments	
(hook-args).	253
<code>__hook_initialize_all::</code> Changes	
to add hook arguments	
(hook-args).	287
<code>__hook_initialize_hook_code:n</code> :	
Changes to add hook arguments	
(hook-args).	288
<code>__hook_initialize_single:ccn</code> :	
Changes to add hook arguments	
(hook-args).	291
<code>__hook_log:nN</code> : Changes to add hook	
arguments (hook-args)	299
<code>__hook_log_next_code:n</code> : Changes	
to add hook arguments	
(hook-args).	303
<code>__hook_make_usable:nn</code> : Changes to	
add hook arguments (hook-args).	251
<code>__hook_new:nn</code> : Add	
<code>\hook_new_with_args:nn</code>	
(hook-args).	250
Update hook code after declaring.	251
<code>__hook_new_reversed:nn</code> : Add	
<code>\hook_new_reversed_with_-</code>	
<code>args:nn</code>	
(hook-args).	253
<code>__hook_normalize_cs_args:nn</code> :	
Macro added (hook-args).	278
<code>__hook_parameter:n</code> : Macro added	
(hook-args).	283
<code>__hook_post_initialization_defs::</code>	
Macro added (hook-args).	309
<code>__hook_set_normalize_fn:nn</code> :	
Macro added (hook-args).	279
<code>__hook_tl_set:cn</code> : Clean-up unused	
variants (hook-args).	247
<code>__hook_try_declaring_generic_hook:wn</code> :	
Changes to add hook arguments	
(hook-args).	268
<code>__hook_use_i_delimit_by_s_mark:nw</code> :	
Use standard naming scheme	
(hook-args).	247
<code>__hook_use_initialized:nw</code> : Add	
<code>\hook_use:nw</code> (hook-args).	309
<code>__hook_use_once:nn</code> : Changes to	
add hook arguments (hook-args).	311
<code>__hook_use_once_clear:n</code> : Changes	
to add hook arguments	
(hook-args).	312
General: Add dedicated rollback code	
to revert data structures	
(hook-args).	326
Messages 'too-many-args',	
'without-args' and 'one-time-args'	
added (hook-args).	317
<code>\AddToHookNextWithArguments</code> : Add	
<code>\AddToHookNextWithArguments</code>	
(hook-args).	321
<code>\AddToHookWithArguments</code> : Add	
<code>\AddToHookWithArguments</code>	
(hook-args).	321
<code>\c__hook_??_parameter_tl</code> : Token	
list added (hook-args).	283
<code>\c__hook_nine_parameters_tl</code> : Add	
auxiliary token lists (hook-args).	247
<code>\g__hook_replacing_stack_seq</code> :	
Macro added (hook-args).	316
<code>\hook_gput_next_code:nn</code> : Add	
<code>\hook_gput_next_code_with_-</code>	
<code>args:nn</code>	
(hook-args).	305
<code>\hook_if_empty:n</code> : Changes to add	
hook arguments (hook-args).	313
<code>\hook_new_pair_with_args:nnn</code> : Add	
<code>\hook_new_pair_with_args:nnn</code>	
(hook-args).	254
<code>\hook_use_once:nw</code> : Add	
<code>\hook_use_once:nw</code> (hook-args).	311
<code>\NewMirroredHookPairWithArguments</code> :	
Add <code>\NewHookWithArguments</code>	
(hook-args).	320
<code>\UseOneTimeHookWithArguments</code> : Add	
<code>\UseHookWithArguments</code>	
(hook-args).	322
2023-04-11 ltfinal.dtx v2.3a	
<code>\@uclclist</code> : Use new generic	
mechanism to detect locale	1347
2023-04-13 ltcmd.dtx v1.1d	
<code>__cmd_grab_v_group_end::</code> Set	
<code>\tex_endlinechar:D</code> earlier	
(gh/876).	186
2023-04-14 ltclass.dtx v1.5h	
<code>\load@onefilewithoptions</code> : Define	
<code>\load@onefilewithoptions</code> when	
in latexrelease (gh/992)	1122
2023-04-15 ltplain.dtx v2.3i	
<code>\extrafloats</code> : Protect box 255 in	
lualatex gh/1041	19

unwind numexpr and ifnum nesting	19	<code>__hook_try_declaring_generic_next_hook:nn</code> :	
2023-04-16 ltfinal.dtx v2.3a		Changes to allow support	
<code>\BCPdata</code> : Command added	1347	arguments in cmd hooks	
2023-04-16 lthooks.dtx v1.1b		(cmd-args).	267
<code>__hook_include_legacy_code_chunk:n</code> :		<code>\c__hook_parameter_cmd/.after_tl:</code>	
<code>__hook_replacing_args_false:</code>		Token lists added (cmd-args).	274
in <code>__hook_include_legacy_code_chunk:n</code>	255	2023-05-26 ltcmd.dtx v1.1e	
2023-04-19 ltfinal.dtx v2.3b		<code>__cmd_defaults_error:w</code> : Use	
<code>\@uclclist</code> : Add		simpler variant <code>\cs_generate_from_arg_count:NNno</code>	141
<code>\DeclareLowercaseMapping</code> ,		2023-05-30 ltfinal.dtx v2.3c	
<code>\DeclareTitlecaseMapping</code> and		<code>\@uclclist</code> : Fix a typo in	
<code>\DeclareUppercaseMapping</code>	1349	implementation of	
2023-04-19 lthooks.dtx v1.1c		<code>\DeclareLowercaseMapping</code> , etc.	1349
<code>__hook_gput_next_do:nn</code> : Initialize		2023-06-06 lthooks.dtx v1.1e	
hook structure when adding 'next'		<code>__hook_code_gset_auxi:eeen</code> :	
code (gh/1052).	306	Short-circuit when the hook is	
<code>\hook_if_empty:n</code> : Simpler and faster		declared without args (gh1078).	276
version (gh/1052).	313	2023-06-14 ltmiscen.dtx v1.2b	
2023-05-12 ltxref.dtx v1.1q		<code>\@vobeytabs</code> : Macro added	881
<code>\label</code> : (UFI)added a hook with		<code>\verbatim*</code> : Support visible tabs in	
argument	844	<code>\verb*</code>	883
(UFI)extended to store five		2023-06-14 ltspace.dtx v1.3q	
arguments	844	<code>\@xobeytab</code> : Macro added	481
2023-05-13 ltmath.dtx v1.2n		2023-06-15 ltmiscen.dtx v1.2b	
<code>\leqno</code> : add <code>\ignorespaces</code> gh/1059	903	<code>\@sverb</code> : Support visible tabs	885
2023-05-15 ltfiles.dtx v1.2t		<code>\@setupverbvisibleSPACE</code> : Support	
<code>\IfFileExists@</code> : Use expl3 file		visible tabs	884
existence test	500	<code>\@vobeyspaces</code> : Support tabs	881
<code>\IfFileExists@@</code> : Macro added	500	2023-06-15 ltmiscen.dtx v1.2q	
2023-05-21 ltcmdhooks.dtx v1.0h		<code>\@setupverbvisibletab</code> : Provide	
<code>__hook_guess_arg_count:nw</code> : Macro		visible tab in <code>\verb*</code>	885
added (cmd-args)	349	2023-06-16 ltfiles.dtx v1.2u	
<code>__hook_make_prefixes:w</code> : Changes		<code>\IfFileExists@@</code> : Support piped	
to allow support arguments in cmd		input	500
hooks (cmd-args)	341	2023-07-02 ltluatex.dtx v1.2c	
<code>__hook_patch_retokenize:Nnnn</code> :		<code>new_luafunction</code> : Ensure existing	
Changes to allow support		table entries not overwritten	
arguments in cmd hooks		gh/1100	57
(cmd-args)	349	2023-08-03 ltluatex.dtx v1.2c	
2023-05-21 lthooks.dtx v1.1d		<code>mlist_to_hlist</code> : Fix callback type of	
<code>__hook_if_cmd_hook:w</code> : Changes to		post <code>mlist_to_hlist_callback</code>	71
allow support arguments in cmd		2023-08-19 ltcmd.dtx v1.2a	
hooks (cmd-args).	315	General: Removed commands that	
<code>__hook_make_usable:nn</code> : Changes to		should have remained only in	
allow support arguments in cmd		<code>xparse.dtx</code>	115
hooks (cmd-args).	252	2023-09-01 ltmiscen.dtx v1.2c	
<code>__hook_new:nn</code> : Changes to allow		<code>\@vobeytabs</code> : Provide global definition	
support arguments in cmd hooks		for active tab	881
(cmd-args).	250	2023-09-06 ltmiscen.dtx v1.2b	
<code>__hook_try_declaring_generic_hook:wn</code> :		<code>\enddocument</code> : Test changes of values	
Changes to allow support		in <code>\new@label@record</code>	866
arguments in cmd hooks			
(cmd-args).	269		

2023-09-20 ltproperties.dtx v1.0c	2024-01-13 ltkeys.dtx v1.0m
\RecordProperties: use	__keys_options_class:n: Trim
\protected@edef for safer	spaces off key names 1154
handling of active chars. 857	__keys_options_package:n: Trim
\SetProperty: use \protected@edef 856	spaces off key names 1155
2023-10-13 ltexpl.dtx v1.3g	2024-01-17 ltproperties.dtx v1.0d
\IfExplAtLeastTF: Provide a test for	__property_record_value_aux:e:
expl3 date (gh/1004) 78	Use \protected@write 856
2023-10-26 ltboxes.dtx v1.4e	2024-01-24 lthooks.dtx v1.1h
\@iframebox: Guard against unknown	__hook_if_usable_use:n: Correct
alignment gh/1072 933	usage of older
\@imakebox: Guard against unknown	__hook_if_file_hook:wTF
alignment gh/1072 928	(gh/1243) 310
\@parboxto: Guard against unknown	__hook_try_declaring_generic_hook_split:nNNnn:
alignment gh/1072 936	Correct usage of older
2023-10-26 ltspace.dtx v1.3r	__hook_if_file_hook:wTF
\nobreakspace: Protected definition	(gh/1243) 268
for tilde 480	2024-01-27 lttextcomp.dtx v1.1a
2023-11-07 ltcounts.dtx v1.1n	General: Added check file for encoding
\@definecounter: Do not change	subset 830
\the... if already defined	Adjusted/corrected TS1
(gh/823) 559	sub-encoding declarations for
2023-11-07 ltoutenc.dtx v2.1a	various families (gh/1257) 809
General: Add more explanation to	2024-01-29 ltmmarks.dtx v1.0e
error message (gh/1102) 554	__mark_extract_and_handle_marks:nn:
2023-11-07 ltplain.dtx v2.3j	Macro added 1070
\tracingnone: Set	\mark_update_structure_from_material:nn:
\tracinglostchars to 2 in	Macro renamed 1073
\tracingnone (gh/549) 32	\ShowMarksAt: Macro added 1083
2023-11-15 ltfiles.dtx v1.2v	2024-01-30 ltclass.dtx v1.5i
\if@listfiles@hashes: Extend file	\@fileswithoptions: Test group
list information 506	level 1119
\if@listfiles@sizes: Extend file list	2024-03-18 ltthm.dtx v1.0g
information 506	\@endtheorem: Insert link target in the
\listfiles: Extend file list	label (UFI) 1010
information 507	\@thm: Use \@kernel@refstepcounter
2023-11-16 ltpara.dtx v1.0m	to avoid an unwanted target
\g__para_standard_everypar_tl:	(UFI) 1010
Correct error message: hook left	2024-03-21 ltcmd.dtx v1.2d
horizontal not vertical mode	__cmd_grab_v_aux_put:N: Collect
(gh/1182) 449	\endlinechar as \obeyedline .. 189
2023-12-01 ltcmd.dtx v1.2b	2024-03-22 ltclass.dtx v1.5j
__cmd_cmd_type_cases:NnnnnnF:	\load@onefilewithoptions: Apply
Extend for optimized commands 203	one-step expansion to raw option
__cmd_start_optimized::: Optimize	list, to be consistent with change
cmd creation for all-m arguments 135	for gh/580 (gh/1298). 1123
2023-12-22 ltcmd.dtx v1.2c	2024-04-10 ltclass.dtx v1.5k
General: Generalize message	\IfClassLoadedWithOptionsTF:
invalid-bang (gh/1198) 206	Provide T and F conditionals not
2024-01-03 lthooks.dtx v1.1g	just TF (gh/1262) 1107
__hook_log:nN: Fix expansion of	\IfFileLoadedF: Provide
__hook__print_args:nn	\IfFileLoadedTF and variants
argument (gh/1221) 299	(gh/1222) 1107

2024-04-17 ltcmd.dtx v1.2e	<code>__cmd_cmd_type_cases:NnnnnnF</code> : Use <code>__kernel_cs_parameter_spec:N</code> instead of <code>\cs_argument_-</code> <code>spec:N/\cs_parameter_spec:N</code> 203	<code>insertmark</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359) 1079
2024-04-17 ltcmdhooks.dtx v1.0j	General: Use <code>__kernel_cs_parameter_spec:N</code> instead of <code>\cs_argument_-</code> <code>spec:N/\cs_parameter_spec:N</code> 332	2024-05-31 ltmarks.dtx v1.0g <code>\c__mark_empty_tl</code> : Initialize all marks with an id, use 0 when a new class is made (gh/1359) 1070
2024-04-17 ltdefs.dtx v1.5t	General: Rename <code>\@expl@cs@argument@spec@N</code> to <code>\@expl@cs@parameter@spec@N</code> (gh/1014) 80	<code>\mark_use_last:nn</code> : Remove the id when returning the mark value (gh/1359) 1080
2024-04-17 ltexpl.dtx v1.3h	General: Add a kernel-level copy of <code>\cs_parameter_spec:N</code> 76	2024-06-04 ltclass.dtx v1.5l <code>\@cls@pkg</code> : Allow for extensions other than "cls" and "pkg" 1129
	Rename <code>\@expl@cs@argument@spec@N</code> to <code>\@expl@cs@parameter@spec@N</code> (gh/1014) 76	<code>\@onefilewithoptions</code> : New argument <code>\pkgcls@ext</code> (gh/870) 1140
	Update name of <code>expl3</code> function 76	<code>\pkgcls@parse@date@arg</code> : New argument <code>\pkgcls@ext</code> (gh/870) 1143
2024-04-17 ltproperties.dtx v1.0e		2024-06-04 ltluatex.dtx v1.2d <code>provides_module</code> : Add <code>v</code> to version string if required (gh/1364) 53
	<code>\IfLabelExistsF</code> : Renamed <code>\IfLabelExistTF</code> to <code>\IfLabelExistsTF</code> (gh/1262) 860	2024-06-10 ltboxes.dtx v1.4g <code>\@finalstrut</code> : Always use a <code>\vrule</code> strut after all, but back up by a baseline if already in vertical mode. Otherwise empty table p-cells will not get the correct width (bug seen first with <code>colortbl</code>) 945
	<code>\IfPropertyExistsF</code> : Renamed <code>\IfPropertyExistTF</code> to <code>\IfPropertyExistsTF</code> (gh/1262) 860	2024-06-10 lttagging.dtx v1.0b <code>\tbl_crchr:n</code> : Always issue a <code>\crchr</code> even if we are at the start of a row to avoid problems with tabulary and similar code 1328
2024-04-18 ltboxes.dtx v1.4f	<code>\@finalstrut</code> : Use a <code>\hrule</code> strut not a <code>\vrule</code> if already in vertical mode (bug seen first with <code>footmisc/14</code>) 945	2024-06-17 ltfsbas.dtx v3.2m <code>\showhyphens</code> : set <code>\tracinglostchars</code> to 0 in <code>\showhyphens</code> 598
2024-04-22 lttextcomp.dtx v2.1a	General: Drop default option info (gh/1333) 819	2024-06-19 ltkeys.dtx v1.0n <code>__keys_options_class:n</code> : Refactor function 1154
2024-04-24 lttextcomp.dtx v2.1b	General: Load the 2018 version when rolling back prior to 2018-08-11 (gh/1333) 818	<code>__keys_options_class:nn</code> : New function 1154
2024-05-30 ltmarks.dtx v1.0f	<code>__mark_update_dblcol_structures::</code> Correct logic for first mark in page region if first column contains no marks (gh/1359) 1086	<code>__keys_options_class:nnn</code> : Refactor function 1154
	<code>__mark_value:nn</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359) 1079	Track options used by classes 1154
	<code>\mark_insert:nn</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359) 1078	<code>__keys_options_package:nn</code> : New functions 1155
		<code>__keys_options_package:nnn</code> : Skip options given to packages 1155
		<code>__keys_scope:N</code> : New function 1151
		<code>__keys_scope:n</code> : New function 1151
		<code>\l_keys_class_only_clist</code> : New variable 1152
		<code>\l_keys_forced_global_clist</code> : New variable 1152

.pass-to-packages: New key property	1151	and empty hook (gh/1423)	259
2024-06-20 ltkeys.dtx v1.0o		_hook_parse_dot_label:nN: Distinguish between empty label and empty hook (gh/1423)	258
_keys_scope:N: Ensure only key name is stored	1151	_hook_parse_label_default:nN: Distinguish between empty label and empty hook (gh/1423)	258
2024-06-23 ltboxes.dtx v1.4h		General: New message “empty-hook” (gh/1423).	318
\color@endgroup: Adjust for new @endpe handling	930	2024-08-16 ltluatex.dtx v1.2e	
2024-06-23 ltlists.dtx v1.0u		provides_module: Support missing version string (gh/1443)	53
\@doendpe: Set \if@endpe to false before calling \par (needed for tagging)	918	2024-09-03 fontdef.dtx v3.0k	
\propagate@doendpe: Set \if@endpe globally and also set up migration to the outside	919	General: Load also t1cmss.fd and t1cmtt.fd	769
2024-06-23 ltmiscen.dtx v1.2d		2024-09-04 ltlength.dtx v1.1e	
\begin: Separate \begin and \end definitions for individual rollback	873	\@settodim: (UF) Suspend tagging .	569
\end: Separate \begin and \end definitions for individual rollback	874	2024-09-05 ltproperties.dtx v1.0f	
\end\verbvisiblespace: Adjust for new @endpe handling	876	_property_record_value_aux:e: Remove \if@files test to be in line with \label, tagging-project issue 696	856
2024-06-23 ltpara.dtx v1.0n		2024-09-10 ltspace.dtx v1.3s	
\g__para_standard_everypar_tl: Append \everypar toks to \g__para_standard_everypar_tl, rollback 2023/06/01 (gh/1386) . .	450	\addpenalty: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460)	475
2024-06-24 fontdef.dtx v3.0j		\addvspace: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460)	474
General: load ts1 cmr fd file in Unicode engines	770	2024-09-11 lterror.dtx v1.2u	
2024-07-06 ltcmd.dtx v1.2f		\@LRmoderr: Error message added .	436
_cmd_declare_env:ennn: Use space-trimmed envname directly (gh/1399)	137	2024-09-13 lttagging.dtx v1.0h	
\NewDocumentEnvironment: Trim spaces from envname first (gh/1399)	212	float/hmode/end: Sockets for floats added	1320
2024-07-10 ltmiscen.dtx v1.2e		2024-09-20 ltcounts.dtx v1.1o	
General: Drop code chunks before adding them to avoid duplication in rollback (gh/1407)	868	\@addtoreset: add the parent theHfoo if a counter is reset	561
\enddocument: Drop code chunks before adding them to avoid duplication in rollback (gh/1407)	866	\@definecounter: define theHfoo (used for internal links)	559
2024-07-13 lttagging.dtx v1.0c		2024-09-20 lttagging.dtx v1.0h	
tbl/leaders/end: Sockets for \cline leaders added (tagging/134) . .	1319	General: moved \@kernel@refstepcounter into ltxref	1328
2024-08-03 ltclass.dtx v1.5m		2024-09-20 ltxref.dtx v1.1r	
\pkgcls@use@this@release: Add selected release to the file list (gh/1413)	1145	\refstepcounter: add sockets	845
2024-08-09 lthooks.dtx v1.1i		provide a kernel copy kernel@refstepcounter	845
_hook_normalize_hook_args_aux:Nn: Distinguish between empty label		set also currentHref	845
		2024-09-21 ltmeta.dtx v1.0c	
		General: Added tagging support . . .	460

2024-09-25 ltproperties.dtx v1.0g	Support for macron-i for T1 (gh/1453)	533
\IfLabelExistsF: Fixed definitions of \IfLabelExistsT and \IfLabelExistsF	2024-11-09 ltmarks.dtx v1.1a __mark_class_status:nnn: Add a third argument	1082
2024-10-10 ltagging.dtx v1.0i	__mark_status:nn: Add a second argument	1083
default: Restore also paratagging (tagging/723)	\mark_clear_structure:n:	1077
2024-10-12 ltmiscen.dtx v1.2f	\mark_copy_structure:nn: Macro renamed	1077
\begin: Make \begin engine-protected	\mark_use_last:nn:	1080
2024-10-21 lthooks.dtx v1.1j	2024-11-12 ltfinal.dtx v2.3d \@uclclist: Add option to titlecase first or all words	1347
\IfHookEmptyF: Define \IfHookEmptyT, \IfHookEmptyF	Use updated titlecase-first function in expl3	1347
2024-10-21 ltmarks.dtx v1.0h	2024-11-16 ltpage.dtx v1.0o \markright: Drop legacy mark support	1090
\IfMarksEqualF: Define \IfMarksEqualT, \IfMarksEqualF	2024-11-19 fontdef.dtx v3.0l General: Preload the TS1 .fd file	770
2024-10-21 ltproperties.dtx v1.0h	Preload the TS1 ts1cmr.fd file in all engines	769
\IfPropertyRecordedF: Define \IfPropertyRecordedT, \IfPropertyRecordedF	2024-11-21 ltagging.dtx v1.0l \UseExpandableTaggingSocket: Define \tag_if_active:TF conditionals here (github/1558)	1311
2024-10-21 ltagging.dtx v1.0k	2024-11-26 lthooks.dtx v1.1l __hook_hash_check_aux:w: Adjust debugging message (gh/1459)	263
\UseExpandableTaggingSocket: Added expandable variants	__hook_initialize_all::: Adjust debugging message (gh/1459)	287
Changed behavior of two argument tagging sockets when disabled.	__hook_initialize_hook_code:n: Adjust debugging message (gh/1459)	288
2024-10-26 ltcounts.dtx v1.1p	__hook_initialize_single:ccn: Adjust debugging message (gh/1459)	292
\@addtoreset: Fully expand counter name in theHfoo commands (gh/1508)	\hook_gput_next_code:nn: Add debugging message (gh/1459)	305
\@definecounter: Fully expand counter name in theHfoo commands (gh/1508)	2024-11-26 ltoutenc.dtx v2.1c \@inmathwarn: Declare command for logging (gh/1242)	513
2024-10-27 ltsockets.dtx v0.9b	\@strip@args: Alter error and info message	517
\socket_set_plug:nnn: Make plug definition non-protected	\UndeclareTextCommand: Log text command change (gh/1242)	520, 521
\socket_use_expandable:n: Added \socket_use_expandable:n	Use \ifcurname to avoid generating a curname	520
2024-10-29 lthooks.dtx v1.1k	2024-11-27 ltfsdcl.dtx v3.1d \freeze@math@version: Reset math version if necessary (gh/1101 and gh/1028)	710
__hook_list_if_rule_exists:nnnF: Skip mapping over undeclared \g_hook_{hook}_code_prop (gh/1513)		
__hook_log:nN: Skip mapping over undeclared \g_hook_{hook}_code_prop (gh/1513)		
2024-11-02 ltdirchk.dtx v1.3b		
General: Add tab char to \dospecials		4
2024-11-02 ltplain.dtx v1.3k		
General: Add tab char to \dospecials		14
2024-11-06 ltoutenc.dtx v2.1b		
General: Support for macron-i for OT1		527

Reset top-level alphabet definitions only for the current math version (gh/1101 and gh/1028)	711	2025-01-02 ltclass.dtx v1.5n	
2024-11-30 ltfssbas.dtx v3.2o		<code>\AtEndDocument</code> : Do not make <code>\AtBeginDocument</code> preamble only (gh/1604)	1129
<code>\@math@level</code> : New approach to <code>localmathalphabets</code> (gh/1101)	585	2025-01-03 lthooks.dtx v1.1l	
<code>\frozen@everydisplay</code> : New approach to <code>localmathalphabets</code> (gh/1101)	586	General: Correct example to use <code>env/quote/begin</code> instead of <code>/before</code> (gh/1599)	221
<code>\frozen@everymath</code> : New approach to <code>localmathalphabets</code> (gh/1101)	586	2025-01-10 ltmarks.dtx v1.1c	
<code>\math@version</code> : New approach to <code>localmathalphabets</code> (gh/1101)	584, 585	<code>__mark_get_from_splitmarks::</code> Do not expand mark content while debugging	1076
2024-12-03 ltmarks.dtx v1.1b		2025-01-15 ltkeys.dtx v1.0p	
<code>\c__mark_empty_tl</code> : Fix inconsistent local/global assignment (gh/1574)	1069	<code>__keys_options_aux:n</code> : Only process global options on first pass . . .	1153
2024-12-11 ltfssaxes.dtx v1.0k		2025-01-21 ltcmd.dtx v1.2g	
General: Add <code>ssc</code> shape change rules (gh/1581)	662	<code>__cmd_grab_v_aux_catcodes::</code> Store spaces and tabs as active chars . .	189
Rationalize <code>sw</code> shape change rules (gh/1581)	662	<code>__cmd_grab_v_aux_put:N</code> : Simplify catcode handling	189
Reorganized the rollback data . . .	660	2025-01-21 ltoutput.dtx v1.4l	
2024-12-13 ltfssaxes.dtx 1.0k		<code>\@writeseup</code> : Support extended syntax for <code>\label</code> , <code>\index</code> , and <code>\glossary</code> (gh/311)	1257
General: Add numerous <code>\DeclareFontSeriesChangeRule</code> entries to support the full range of weights (from <code>ul</code> to <code>ub</code>) and widths (from <code>uc</code> to <code>ux</code>) (gh/1583)	603	2025-01-21 ltsect.dtx v1.1g	
Minor modifications to a few <code>\DeclareFontSeriesChangeRule</code> entries (gh/1583)	603	<code>\@gobble@with@sphack@som</code> : Support extended syntax for <code>\label</code> , <code>\index</code> and <code>\glossary</code> (gh/311)	1022
2024-12-21 ltoutenc.dtx v2.1c		<code>\addtocontents</code> : Support extended syntax for <code>\label</code> , <code>\index</code> , and <code>\glossary</code> (gh/311)	1022
General: Correct base letter (gh/1587)	533	2025-01-22 ltcounts.dtx v1.2a	
2024-12-22 ltcmdhooks.dtx v1.0k		<code>\@ifbothcounters</code> : Guard against star alias.	563
<code>__hook_try_put_cmd_hook:w</code> : Avoid defining command while adding a cmd hook (gh/1591)	334	<code>\c@*:</code> <code>\c@*</code> added (gh/1632)	559
2024-12-27 ltsockets.dtx v0.9c		2025-01-22 ltxref.dtx v1.1t	
<code>\IfSocketPlugAssignedF</code> : Conditionals for sockets, plugs, and assignments (gh/1577)	367	<code>\ltx@star@counter</code> : Macro added . .	845
<code>\socket_if_exist:n</code> : Conditionals for sockets, plugs, and assignments (gh/1577)	363	<code>\refstepcounter</code> : Guard * from causing infinite loop	845
<code>\socket_if_plug_assigned:nn</code> : Conditionals for sockets, plugs, and assignments (gh/1577)	365	2025-01-23 ltcmd.dtx v1.3a	
<code>\socket_if_plug_exist:nn</code> : Conditionals for sockets, plugs, and assignments (gh/1577)	364	<code>__cmd_add_grabber:N</code> : Extend to support c-type grabbing	157
		<code>__cmd_add_type_b:w</code> : Extend to cover c-type grabbing	154
		<code>__cmd_add_type_b_or_c:N</code> : New function	154
		<code>__cmd_add_type_c:w</code> : New function	154
		<code>__cmd_grab_D_aux:NNNNNN</code> : Extend to support c-type grabbing	179
		<code>__cmd_grab_D_long_obey_spaces_no_strip:w</code> : Auto-generate D-grabbers	178
		<code>__cmd_grab_D_verb_safe:NN</code> : New macro	179

<code>__cmd_grab_c:w</code> : New c-type grabbing function	174	<code>\@if@footnotes@TF</code> : Macro added	1248
<code>__cmd_normalize_type_b:w</code> : Extend to cover c-type grabbing	149	<code>\@make@normalcolbox</code> : Macro added	1242
<code>__cmd_normalize_type_b_or_c:nn</code> : New function	149	<code>\@make@specialcolbox</code> : Macro added	1242
<code>__cmd_normalize_type_c:w</code> : New function	149	<code>\@makecol</code> : Use sockets and hooks	1241
<code>__cmd_prepare_signature:n</code> : Extend to cover c-type grabbing	152	<code>\@outputbox@append</code> : Macro added	1246
General: Generalize message <code>arg-after-body</code>	206	<code>\@outputbox@appendfootnotes</code> : Macro added	1246
New message <code>chars-dropped-first-line</code>	206	<code>\@outputbox@attachbottomfloats</code> : Macro added	1247
New message <code>chars-dropped-last-line</code>	206	<code>\@outputbox@attachfloats</code> : Macro added	1247
New variable	130, 133	<code>\@outputbox@attachtopfloats</code> : Macro added	1247
2025-01-23 ltoutput.dtx v1.4m		<code>\@outputbox@depth</code> : Macro added	1242
<code>\@writsetup</code> : Make <code>\label</code> , <code>\index</code> and <code>\glossary</code> truly invisible when typesetting (gh/1638)	1257	<code>\@outputbox@removebskip</code> : Finally fix bottom skip bug (from 2.09)	1244
2025-01-23 ltsect.dtx v1.1h		Macro added	1244
<code>\@gobble@with@sphack@som</code> : Make <code>\label</code> , <code>\index</code> and <code>\glossary</code> truly invisible when typesetting (gh/1638)	1022	<code>\@writsetup</code> : Hooks and sockets added	1254
2025-01-26 ltlists.dtx v1.0v		General: Use sockets and hooks in OR	1248
<code>\@propagate@doendpe</code> : Only migrate <code>\@doendpe</code> out of simple and semi-simple groups (gh1641)	919	<code>build/column/after</code> : Hook added	1251
2025-01-27 lttagging.dtx 1.0m		<code>build/column/outputbox</code> : Socket added	1248
General: add sockets for math phantom commands	1322	<code>build/page/after</code> : Hook added	1251
2025-01-29 ltoutput.dtx v1.4n		<code>build/page/reset</code> : Hook added	1251
<code>\@addtocurcol</code> : Save current value of <code>\@reqcolroom</code> (gh/1645)	1269	2025-02-14 lttagging.dtx v1.0m	
<code>\@addtonextcol</code> : Save current value of <code>\@reqcolroom</code> (gh/1645)	1279	General: Tagging support for output routines added	1320
<code>\@flcheckspace</code> : Reset current value of <code>\@reqcolroom</code> (gh/1645)	1298	<code>build/column/footins</code> : Tagging socket added	1320
2025-02-01 ltcmd.dtx v1.3b		<code>build/column/outputbox</code> : Tagging socket added	1320
<code>\ProcessList</code> : Use <code>\tl_map_tokens:nn</code> instead of <code>\tl_map_function:nN</code>	216	<code>build/page/footer</code> : Tagging socket added	1320
2025-02-11 ltoutenc.dtx v2.1c		<code>build/page/header</code> : Tagging socket added	1320
<code>\@inmathwarn</code> : Log text command/symbol redeclarations (gh/1242)	513	2025-02-19 lttagging.dtx v1.0n	
2025-02-14 ltoutput.dtx v1.4m		General: Moved math sockets into lttagging	1321
<code>\@combinefloats</code> : Macro got a new name	1260	2025-02-20 ltfssbas.dtx v3.2p	
<code>\@if@bottomfloats@TF</code> : Macro added	1248	<code>\CheckEncodingSubset</code> : If necessary, load the <code>.fd</code> before checking the encoding subset (gh/1669)	578
<code>\@if@flushbottom@TF</code> : Macro added	1248	2025-02-21 lttagging.dtx v1.0n	
		<code>marginpar/end</code> : move marginpar sockets	1318
		2025-02-22 ltcounts.dtx v1.2b	
		<code>\counterwithin</code> : <code>\counterwithin</code> needs to expand its arguments (gh/1675)	563

2025-02-25 ltoutput.dtx v1.4o	(gh/1687)	22
\@writesetup: Reset various catcodes to their default values in case a pagebreak occurs while a special catcode scheme is in force (gh/600)		1255
2025-02-26 lttagging.dtx v1.0o		
\AssignTaggingSocketPlug: Macro added		1311
\NewTaggingSocket: Initialize tagging sockets with 1 or 2 arguments		1311
2025-03-01 ltcounts.dtx v1.2b		
\counterwithout: \counterwithout needs to expand its arguments (gh/1675)		564
2025-03-01 ltexpl.dtx v1.3i		
\IfExplAtLeastTF: Also provide T and F variants of \IfExplAtLeastTF (gh/1522)		78
2025-03-01 ltfinal.dtx v2.3e		
\IfPDFManagementActiveTF: Provide T F variants of \IfPDFManagementActiveTF (gh/1522)		1353
2025-03-01 ltfssini.dtx v3.2k		
\IfFontSeriesContextTF: Provide T F variants of \IfFontSeriesContextTF (gh/1522)		754
2025-03-01 ltmeta.dtx v1.0d		
General: Provide T F variants of \IfDocumentMetadataTF (gh/1522)		458
2025-03-05 ltexpl.dtx v1.3j		
\@kernel@after@begindocument: Add \@kernel@before@enddocument@afterlastpage		72
2025-03-05 ltfssbas.dtx v3.2q		
\showhyphens: set \tracinglostchars to 0 in \showhyphens for all engines		598
2025-03-05 ltmiscen.dtx v1.2g		
\enddocument: Added \@kernel@before@enddocument@afterlastpage		866
2025-03-05 ltoutput.dtx v1.4p		
\clearpage: A L ^A T _E X 2 _ε name for \write in case it gets changed after the last page		1228
Make sure this is a normal write in all circumstances		1228
2025-03-05 ltplain.dtx v2.3l		
General: Use 4 as default value for \tracinglostchars in LuaT _E X (gh/1687)		22
\loggingall: Remove \tracinglostchars setting (gh/1691)		31
\tracingnone: Remove \tracinglostchars setting (gh/1691)		32
2025-03-05 ltshipout.dtx v1.0o		
__shipout_force_immediate_writes:: Macro added		1210
2025-03-05 ltshipout.dtx v1.4p		
\@kernel@before@enddocument@afterlastpage: Make all \writes immediate after the last page (gh/1689)		1206
2025-03-08 ltmath.dtx v1.2o		
\@eqnset: \dollar@begin and \dollar@end introduced for tagging		900, 901
\]: \dollar@begin and \dollar@end introduced for tagging		897
\c@equation: \dollar@begin and \dollar@end introduced for tagging		899
\dollar@end: \dollar@begin and \dollar@end introduced for tagging		897
eqnarray: \dollar@begin and \dollar@end introduced for tagging		906
2025-03-20 ltoutput.dtx v1.4q		
\@outputbox@appendfootnotes: Add experimental socket for baseline attachment of footnotes – subject to change (footmisc/19)		1247
\@outputbox@removebskip: Remove a ‘fil’ bottom skip even if we are in ragged bottom typesetting (footmisc/19)		1244
\newpage: Remove skip added in 2.4a (footmisc/19)		1229
\build/column/baselineattach: Add experimental socket for baseline attachment of footnotes – subject to change (footmisc/19)		1251
2025-03-25 ltoutput.dtx v1.4r		
\@addtobot: Don’t set \maxdepth to zero any more (footmisc/19)		1267
\@backup@outputbox@depth: Macro added (footmisc/19)		1245
\@outputbox@reinsertbskip: Back out the depth if we have no		

footnotes and no bottom floats (footmisc/19)	1245	2025-04-25 ltfiles.dtx v1.2w <code>\protected@write</code> : Move <code>\write</code> outside the group /gh1717	492
on: (footmisc/19)	1251	2025-04-27 ltshipout.dtx v1.0p <code>\@kernel@after@enddocument@afterlastpage</code> : Increment page counter gh/1717	1207
footnotes-floats-legacy: Add support for attaching footnotes with a skip starting at the baseline of the last text line (footmisc/19)	1250	<code>__shipout_force_immediate_writes::</code> Decrement page counter gh/1717	1210
2025-03-29 ltoutput.dtx v1.4s <code>\@outputbox@attachbottomfloats</code> : Add socket (footmisc/19)	1247	2025-04-28 ltmiscen.dtx v1.2h <code>\enddocument</code> : Increment page counter gh/1717	866
footnotes-floats-legacy: Add socket (footmisc/19)	1250	2025-05-10 ltclass.dtx v1.5m <code>\IfClassLoadedWithOptionsTF</code> : avoid errors for par in code argument of T versions (gh/1733)	1107
2025-03-40 ltexpl.dtx v1.3k <code>\expandableinput</code> : Added document level name for <code>\file_input_raw:n</code> (gh/514)	78	2025-05-29 ltmath.dtx v1.2p <code>\leqno</code> : Ensure saved version is primitive gh/1747	903
2025-04-12 ltcmd.dtx v1.3c <code>__cmd_add_arg_spec_mandatory:nn</code> : Track total mandatory args	151	2025-06-03 ltdefs.dtx v1.5u <code>\provide@command</code> : Strip trailing spaces from arg (gh/1708)	89
<code>__cmd_add_grabber:N</code> : Refine handling of optional arguments for c-type grabbing	157	<code>\renew@command</code> : Strip trailing spaces from arg (gh/1708)	87
<code>__cmd_grab_c_loop:w</code> : Correct behavior on dropping tokens	175	2025-06-04 ltfinal.dtx v2.3f <code>\@uclclist</code> : Add default for titlecasing	1347
<code>__cmd_normalize_arg_spec:n</code> : Track total mandatory args	144	2025-06-05 ltexpl.dtx v1.3l <code>\IfExplAtLeastTF</code> : Provide T and F variants of <code>\IfExplAtLeastTF</code> with a correct implementation (gh/1752)	78
General: New variable	130	2025-06-07 ltoutput.dtx v1.4y <code>\@writsetup</code> : Reset catcode of CR (gh/1766)	1255
2025-04-13 ltoutput.dtx v1.4x <code>\newpage</code> : reinserted par token (github/1712)	1229	2025-06-08 ltoutput.dtx v1.4y <code>\@writsetup</code> : Reset catcode of TAB (gh/1766)	1255
2025-04-14 ltoutput.dtx v2.3m General: Always use extended free list	1224	2025-06-19 ltmarks.dtx v1.1d <code>__mark_new_class:nn</code> : Also initialize saved-column region (SX chat)	1069
2025-04-14 ltplain.dtx v2.3m <code>\e@alloc</code> : Always use extended allocation	18	2025-06-23 ltfssaxes.dtx 1.0l General: Improve the order of the <code>\DeclareFontSeriesChangeRule</code> entries (gh/1727)	626
<code>\e@alloc@chardef</code> : Always use extended allocation	18	Offer <code>m</code> as alternative result series for some <code>\DeclareFontSeriesChangeRule</code> entries (gh/1727)	627
<code>\e@alloc@top</code> : Drop a redundant ε -TeX test	18	2025-06-25 ltkeys.dtx v1.0q <code>__keys_options_aux:n</code> : Revise for multiple family support	1153
<code>\e@mathgroup@top</code> : Always use extended allocation	18	<code>__keys_options_local::</code> : Revise for multiple family support	1155
<code>\extrafloats</code> : Always use extended allocation	18		
<code>\newinsert</code> : Always use extended allocation	20		
Drop redundant ε -TeX test	20		
<code>\newlanguage</code> : Always use extended allocation	17		
2025-04-21 ltcmd.dtx v1.3d <code>__cmd_grab_c_auxviii::</code> Correct a logic flaw	176		

<code>__keys_options_local:n</code> : New function	1156	statements are affected (gh/1845)	1256
<code>__keys_options_local:nnn</code> : New function	1156	2025-08-23 lttagging.dtx v1.0r	
<code>\l_keys_local_clist</code> : New variable	1152	General: MathML intent macros added	1310
2025-06-26 ltdefns.dtx v1.5v		2025-09-15 ltfsdcl.dtx v3.1e	
<code>\declare@robustcommand</code> : Allow for active chars	92	<code>\process@table</code> : Run <code>\set@current@meta@family</code> and the <code>normalfont</code> hook	713
<code>\declare@robustcommand@auxi</code> : New macro	92	2025-09-15 ltfsini.dtx v3.2l	
<code>\declare@robustcommand@auxii</code> : New macro	92	<code>\@currentmetafamily</code> : Macro added	740
<code>\declare@robustcommand@auxiii</code> : New macro	92	<code>\prepare@family@series@update</code> : Always set <code>\@currentmetafamily</code>	740
2025-06-27 ltcmd.dtx v1.3e		<code>\reset@font</code> : Drop old 2020 hook <code>\@defaultfamilyhook</code>	761
<code>__cmd_show:e</code> : Support <code>\showstream</code> (gh/1062)	170	Set current meta family	761
2025-06-27 lthooks.dtx v1.1m		<code>\set@current@meta@family</code> : Macro added	761
<code>\hook_log:n</code> : Support <code>\showstream</code> (gh/1062)	298	<code>\update@series@target@value</code> : Rename <code>\target@meta@family@value</code> to <code>\@currentmetafamily</code> and set it always	741
2025-06-29 ltfsstrc.dtx v3.0q		2025-09-23 ltcmd.dtx v1.3f	
<code>\getanddefine@fonts</code> : Introduced <code>\transform@scriptfont</code> for script fonts.	684	<code>__cmd_add_arg_spec_mandatory:nn</code> : Correct logic for recording position of mandatory args	151
2025-07-01 ltfinal.dtx v2.3g		General: Rename variable	130
<code>\@uclclist</code> : Add <code>\DeclareLowercaseExclusions</code> , <code>\DeclareTitlecaseExclusions</code> and <code>\DeclareUppercaseExclusions</code>	1349	2025-09-23 ltfinal.dtx v2.3h	
2025-07-18 ltoutenc.dtx v2.1d		General: Setting <code>\partokencontext</code>	1350
General: Add forgotten <code>\DeclareFontSubstitution</code> (gh/1709)	526, 528	2025-09-24 lthooks.dtx v1.1n	
2025-08-01 ltproperties.dtx v1.0j		<code>__hook_hash_check:nTF</code> : New function	262
<code>__property_gset:nnnn</code> : correct name of shipout boolean	855	<code>__hook_hash_check_aux:w</code> : New function	262
2025-08-03 lttagging.dtx v1.0s		Only double hashes when needed	264
General: Add sockets for block code	1317	2025-09-24 lttab.dtx v1.1t	
2025-08-11 ltmarks.dtx v1.1e		<code>\@array</code> : Check <code>\currentgrouptype</code> in <code>\par</code> (gh1864)	961
<code>__mark_vbox_set_split_to_maxdimen:NN</code> : Use <code>\ignoreprimitiverror</code> if available (gh/1750)	1072	2025-10-01 ltdefns.dtx v1.5w	
2025-08-21 lttagging.dtx v1.0t		<code>\@star@or@long</code> : Handling protection status differently (gh/571)	83
<code>default</code> : Use symbolic name instead of <code>text-unit</code>	1314	<code>\@yargd@f</code> : Ensure that commands without arguments are not long (gh/571)	86
2025-08-22 ltoutput.dtx v1.4z		Handling protection status differently (gh/571)	86
<code>\@writesetup</code> : Don't use <code>\nfss@catcodes</code> + updates, but spell everything out explicitly	1255	<code>\declare@robustcommand@auxiii</code> : Handling protection status differently (gh/571)	93
Move setting of <code>\protect</code> after all hooks so that only the <code>\write</code>		<code>\pr@tectedrel@x</code> : Handling protection status differently (gh/571)	84

2025-10-02 ltcmd.dtx v1.3g	2025-11-10 ltcmdhooks.dtx v1.0m
<code>__cmd_arg_spec_opt:N</code> : New function 205	<code>__hook_double_hashes_space:w</code> : Support <code>\alignmark</code> 344
<code>__cmd_cmd_type_cases:NnnnnnF</code> : Remove protected status 203	2025-11-10 ltxref.dtx v1.1v
<code>__kernel_cmd_if_xparse:NTF</code> : Remove protected status 203	<code>\label:</code> (UFI)Disable commands while writing to the aux, gh1841 844
<code>\cmd_arg_spec:c</code> : New function . . . 205	2025-11-20 ltproperties.dtx v1.0l
2025-10-12 ltcmdhooks.dtx v1.0l	<code>\property_item:ee</code> : added
<code>__hook_retokenize_patch:Nnn</code> : Reset the catcode of space when patching 346	<code>\property_item:nn</code> , gh1194 858
2025-10-20 ltagging.dtx v1.0u	2025-11-22 ltagging.dtx v1.0v
General: Tagging sockets for headings added 1317	General: Add sockets for minipage and parbox 1317
2025-11-04 ltcmd.dtx v1.3h	2025-11-27 ltfinal.dtx v2.3i
<code>__cmd_add_default_E:nn</code> : Update for <code>\NoValue</code> adjustment 158	<code>\@addtofilelist</code> : Support commas in file names (gh/1920) 1352
<code>__cmd_add_expandable_type_E:w</code> : Update for <code>\NoValue</code> adjustment 160	<code>\@addtofilelist@aux</code> : New macro . . . 1352
<code>__cmd_add_type_v:w</code> : Update for <code>\NoValue</code> adjustment 157	2025-12-17 ltspace.dtx v1.3t
<code>__cmd_expandable_grab_D:NNwNNn</code> : Update for <code>\NoValue</code> adjustment 191	<code>\esphack</code> : Prevent generating an extra space when chained (gh/1910) 471
<code>__cmd_expandable_grab_D_alt:NNwNNn</code> : Update for <code>\NoValue</code> adjustment 193	2025-12-23 ltluatex.dtx v1.2f
<code>__cmd_expandable_grab_R_alt_aux:NNwNNn</code> : Update for <code>\NoValue</code> adjustment 195	<code>\newluachunkname</code> : Use <code>\csstring</code> instead of <code>\string</code> 50
<code>__cmd_expandable_grab_R_aux:NNwNNn</code> : Update for <code>\NoValue</code> adjustment 194	2025-12-31 ltcounts.dtx v1.2c
<code>__cmd_grab_D_aux:NNnNNN</code> : Update for <code>\NoValue</code> adjustment 179	<code>\@addtoreset</code> : Support alias counters 561
<code>__cmd_grab_E_long_obey_spaces:w</code> : Update for <code>\NoValue</code> adjustment 182	<code>\@definecounter</code> : Add support for alias counter 559
<code>__cmd_grab_R_aux:NNnN</code> : Update for <code>\NoValue</code> adjustment 185	<code>\@removefromreset</code> : Support for alias counter 562
<code>__cmd_grab_v_aux_abort:n</code> : Update for <code>\NoValue</code> adjustment 189	General: Added <code>\newcounteralias</code> . . . 556
<code>__cmd_normalize_type_s:w</code> : Update for <code>\NoValue</code> adjustment 145	<code>\newcounteralias</code> : New command
<code>__cmd_show_processor:Nw</code> : Update for <code>\NoValue</code> adjustment 172	<code>\newcounteralias</code> 558
<code>__cmd_split_argument_aux:nnnn</code> : Update for <code>\NoValue</code> adjustment 197	2025-12-31 ltthm.dtx v1.0h
2025-11-05 ltcmd.dtx v1.3h	<code>\@othm</code> : Use new alias counter (UFI) 1009
<code>__cmd_if_novalued_aux:w</code> : Update for <code>\NoValue</code> adjustment 215	2026-01-02 ltfssr.dtx v3.0r
General: Update for <code>\NoValue</code> adjustment 215	<code>\DeclareMathScriptfontMapping</code> : Mappings defined by
<code>\NoValue</code> : New command 215	<code>\DeclareMathScriptfontMapping</code> are set globally. 685
<code>\tl_if_novalued:n</code> : Update for <code>\NoValue</code> adjustment 215	2026-01-15 lttab.dtx v1.1u
	General: Hook declarations for array and <code>longtable</code> added 972
	2026-01-20 ltpictur.dtx v1.2c
	<code>\pictur@</code> : adding no-op optional argument for gh/1172 976
	2026-02-02 ltcmd.dtx v1.3i
	<code>__cmd_arg_to_keyvalue_braces:nnn</code> : Treat empty input as empty keyval list 199
	2026-02-15 ltfloat.dtx v1.2h
	<code>\@textsubscript</code> : Drop use of fake math 1045
	<code>\@textsuperscript</code> : Drop use of fake math 1044

<code>\textsubscript@offset</code> : Command added	1045	2026-03-12 lthooks.dtx v1.1o	General: Switch from x- to e-type expansion in expl3 code	245
<code>\textsubscript@space</code> : Command added	1045	2026-03-12 ltkeys.dtx v1.0r	General: Switch from x- to e-type expansion in expl3 code	1151
<code>\textsuperscript@offset</code> : Command added	1044	2026-03-12 ltmarks.dtx v1.1f	General: Switch from x- to e-type expansion in expl3 code	1068
<code>\textsuperscript@space</code> : Command added	1044	2026-03-12 ltpara.dtx v1.0o	General: Switch from x- to e-type expansion in expl3 code	448
2026-02-24 ltboxes.dtx v1.4i		2026-03-12 ltproperties.dtx v1.0m	General: Switch from x- to e-type expansion in expl3 code	855
<code>\@parboxto</code> : Use text mode vertical centering	936	2026-03-12 ltshipout.dtx v1.0q	General: Switch from x- to e-type expansion in expl3 code	1193
<code>\vcenter@text</code> : New macro	935	2026-03-12 ltsockets.dtx v0.9d	General: Switch from x- to e-type expansion in expl3 code	361
<code>\vcenter@text@auxi</code> : New macro	935	2026-03-13 ltcmd.dtx v1.3k	<code>__cmd_defaults_error:w</code> : Tidy up variants	141
<code>\vcenter@text@auxii</code> : New macro	935	2026-03-13 ltexpl.dtx v1.3m	General: Add check for sufficiently-recent expl3	75
<code>\vcenter@text@axis</code> : New macro	935	2026-03-13 lthooks.dtx v1.1o	<code>\str_count:e</code> : Tidy up variants	247
2026-02-24 lttab.dtx v1.1v		2026-03-13 ltkeys.dtx v1.0r	General: Tidy up variants	1152
<code>\@tabarray</code> : Move <code>\m@th</code> to <code>\array</code>	960	2026-04-08 lttagging.dtx v1.0w	<code>struct/end</code> : Added generic sockets for structure	1313
<code>\@tabular</code> : Remove math mode switch	960	2026-04-12 ltsockets.dtx v0.9e	<code>__socket_debug_gset::</code> Use <code>\on@line</code> in debugging messages	362
General: Use text mode vertical centering	960	2026-04-13 ltboxes.dtx v1.4j	<code>\color@endgroup</code> : Do not reset <code>@endpe</code> in <code>\color@endgroup</code> when ending an <code>\hbox</code>	930
<code>\array</code> : Move <code>\m@th</code> from <code>\@tabarray</code>	959	2026-04-16 ltfinal.dtx v2.3l	General: Revise glyphtounicode loading to reduce tokens in <code>\everyjob</code>	1340
<code>\endtabular</code> : Remove math mode switch	959	2026-04-22 ltmeta.dtx v1.0e	General: Add hook to change target name and use it in <code>\NextLinkTarget</code>	460
2026-02-25 ltcmd.dtx v1.3j		2026-04-22 lttagging.dtx v1.0x	<code>inline/end</code> : added inline/begin, inline/end socket	1313
General: Add documentation as part of <code>cmdguide</code> work	115		<code>struct/end</code> : renamed struct-begin socket	1313
2026-02-25 ltfinal.dtx v2.3j				
General: Define <code>\pdfglyphtounicode</code> for lua _{TeX}	1340			
Load and enable glyphtounicode.tex for lua _{TeX} in PDF mode	1340			
Provide noop definitions for the other engines	1340			
2026-03-10 lthooks.dtx v1.1o				
<code>__hook_curr_name_push_aux:n</code> : Support dot notation (gh/1162)	260			
2026-03-12 ltcmd.dtx v1.3k				
General: Switch from x- to e-type expansion in expl3 code	129			
2026-03-12 ltcmdhooks.dtx v1.0n				
General: Switch from x- to e-type expansion in expl3 code	332			
2026-03-12 ltfilehook.dtx v1.0p				
General: Switch from x- to e-type expansion in expl3 code	1165			
2026-03-12 ltfinal.dtx v2.3k				
General: Switch from x- to e-type expansion in expl3 code	1332			
2026-03-12 ltfsdcl.dtx v3.1f				
General: Switch from x- to e-type expansion in expl3 code	702			

2026-04-26 ltcmd.dtx v1.3l	(gh/2132)	362
<code>__cmd_add_arg_spec_mandatory:nn:</code>	2026-07-28 ltcmd.dtx v1.3n	
Update mandatory spec function	<code>__cmd_prepare_signature_verb_chk:n:</code>	
to use two arguments (gh/1204) .	Test last character as a string . .	152
<code>__cmd_normalize_type_s:w:</code> Update	2026-07-29 ltboxes.dtx v1.4k	
<code>\@@_normalize_type_r:w</code> to use	<code>\underline:</code> Avoid fake math in	
<code>\@@_normalize_type_R_aux:w</code>	<code>\underline</code>	943
(gh/1204)	<code>\underline@text:</code> New macro	943
<code>__cmd_normalize_type_v:w:</code> Update	<code>\underline@text@kern:</code> New macro	943
<code>\@@_normalize_type_R:w</code> to use a	<code>\underline@text@thickness:</code> New	
new auxiliary function (gh/1204)	macro	943
2026-04-27 ltsockets.dtx v0.9f	<code>\underline@text@vgap:</code> New macro	943
<code>\socket_get_plug:nN:</code> A command to	2026-07-31 ltoutenc.dtx v2.2a	
retrieved the assigned plug	General: HiTeX support	509
(gh/1851)	2026-08-20 ltagging.dtx v1.1a	
2026-05-01 ltproperties.dtx v1.0n	<code>default:</code> Avoid use of internal tagpdf	
<code>\property_item:een:</code> added	variables	1314
<code>\property_item:nnn,</code> gh2018 . . .	2026-09-13 ltclass.dtx v1.5o	
2026-05-06 ltmeta.dtx v1.0f	<code>\@currnamestack:</code> Put	
General: Add <code>\SaveLastSkip</code> and	<code>\pkgcls@outerdate</code> on the stack	
<code>\RestoreLastSkip</code> to better mimic	(gh/828)	1100
hyperref's spacing behavior,	Retrieve the value for	
gh/2070	<code>\pkgcls@innerdate</code> (gh/828) . .	1102
Use <code>\latelua</code> to mimic a whatsit to	<code>\pkgcls@outerdate:</code> Set	
avoid spurious warning, gh/2070	<code>\pkgcls@outerdate</code> (gh/828) . .	1140
2026-05-13 ltfloat.dtx v1.2i	<code>\pkgcls@parse@date@arg:</code> Set	
<code>\textsubscript@offset:</code> Correct	<code>\pkgcls@outerdate</code> (gh/828) . .	1142
value of offset for large x-height	2026-09-13 ltfilehook.dtx v1.0q	
2026-05-16 ltagging.dtx v1.0z	<code>\g__filehook_nesting_level_int:</code>	
<code>kernel_L(para/textblock/end):</code> Add	File stack now has 4 items per	
unskip, tagging issue 803	file.	1181
2026-05-27 ltmeta.dtx 1.0g	2026-09-13 ltoutput.dtx v1.5a	
General: Add commands related to	<code>\@writesetup:</code> Resetting <code>\catcode</code> of	
document properties container . .	comma added (gh/2056)	1255
2026-06-01 ltcmd.dtx v1.3m	2026-09-15 ltmeta.dtx v1.0h	
General: Switch to US spelling for	<code>\NextLinkTarget:</code> Do not set the	
macro names	spacefactor in math mode, gh2188	461
<code>\NoValue:</code> Allow for overwriting expl3	2026-09-16 ltluatex.dtx v1.2g	
version	<code>new_luacommand:</code> Function added . . .	58
2026-06-01 ltexpl.dtx v1.3n	2026-09-24 ltvers.dtx v1.1l	
General: Switch to US spelling for	<code>\development@branch@name:</code> Use two	
macro names	digits for month and day	35
2026-06-01 ltfinal.dtx v2.3m	2026-09-27 ltfinal.dtx v2.3n	
General: Switch to US spelling for	General: Support strong emphasis	
macro names	(tagging/1620)	1352
2026-06-01 lthooks.dtx v1.1p	2026-09-27 ltfntcmd.dtx v3.5b	
General: Switch to US spellings for	<code>\strongemph:</code> Support strong	
macro names	emphasis (tagging/1620)	793
2026-06-02 ltkeys.dtx v1.0s	2026-09-27 ltfssini.dtx v3.2m	
<code>\DeclareUnknownKeyHandler:</code> Correct	General: Support strong emphasis	
doubling of hashes	(tagging/1620)	758
2026-06-24 ltsockets.dtx v0.9g	2026-10-01 ltfloat.dtx v1.2j	
<code>__socket_debug_gset::</code> Do not	<code>\@textsubscript:</code> Move	
expand <code>\on@line</code> prematurely	<code>\check@mathfonts</code> inside group	1045

\@textsuperscript: Move		v1.0j lttemplates.dtx 2026-04-21	
\check@mathfonts inside group	1044	_template_debug_off:: Show debugging on the terminal	383
2026-10-12 ltshipout.dtx v1.0r		v1.0k lttemplates.dtx 2026-06-24	
_shipout_picture_overlay:n: Add support for links with page breaks (gh/2210)	1192	_template_parse_vars_elt_auxiv:nnw: New function to support use of name	395
2026-10-28 ltuatex.dtx v1.5h		_template_parse_vars_elt_auxv:enn: New function to support use of name	395
General: Rename callback descriptions to identifiers	58	v1.0l lttemplates.dtx 2026-08-14	
add_to_callback: Add warning for duplicate callback identifiers	67	_template_parse_keys_elt:nn: Revise value setting	390
2026-10-29 ltpara.dtx v1.0p		_template_parse_values_elt_auxi:n: Revise value setting	400
_para_handle_indent:: Make \OmitIndent call \noindent when used in vertical mode (tagging/1674)	451	General: New constant	381
v1.0b lttemplates.dtx 2024-02-15		v1.0m lttemplates.dtx 2026-09-29	
\IfInstanceExistsTF: New macros	419	_template_show_values:nn: Error if instance is unknown	412
v1.0c lttemplates.dtx 2024-04-17		v1.0n lttemplates.dtx 2026-10-07	
\IfInstanceExistsTF: Use plural names	419	_template_date_compare:nNnTF: New function	380
v1.0d lttemplates.dtx 2024-10-07		_template_date_compare_aux:nNn: New function	380
_template_assign_variable:: Correct passing of \KeyValue contents	409	_template_date_compare_auxi:nnnNnnn: New function	380
v1.0d lttemplates.dtx 2025-01-08		_template_date_compare_auxiii:w: New function	380
\SetTemplateKeys: Test for empty key/val list to speed up processing	419	_template_declare_template_equiv:nnnn: New function	392
v1.0e lttemplates.dtx 2025-01-20		_template_execute_if_keyname_free:VT: New function	385
_template_declare_template_code:nnnnn: Speed up test for \AssignTemplateKeys	394	_template_parse_equiv_elt:n: New function	392
v1.0f lttemplates.dtx 2025-07-08		_template_parse_equiv_elt:nnnn: New function	392
_template_parse_values_elt_auxi:n: Allow for value expansion	400	_template_parse_equiv_elt_aux:nnn: New function	392
_template_parse_values_elt_auxi:w: New macro	400	_template_parse_values_add:c: New function	400
_template_parse_values_set:c: New macro	400	_template_parse_values_add:e: New function	400
_template_parse_values_set:e: New macro	400	_template_parse_values_elt_auxi:n: Rename function	400
General: New variable	382	_template_parse_values_elt_auxiii:Nn: New function	400
v1.0g lttemplates.dtx 2025-11-21		_template_parse_values_elt_auxiv:w: New function	400
_template_instance_value:nnn: New function	406	_template_parse_values_set:c: Rename function	400
_template_show_values:nn: Extend message	412	_template_parse_values_set:e: Rename function	400
\InstanceValue: New command	419	_template_parse_vars_elt_auxiii:enn: Split out each keytype path	395
v1.0h lttemplates.dtx 2025-12-24			
_template_assign_function:: Create protected functions	408		
v1.0i lttemplates.dtx 2026-03-1			
General: Add debugging support	383		

<code>__template_parse_vars_instance:nnn:</code> New function 397 <code>__template_tmp:n: New function .</code> 383 General: New message ... 413, 414, 416 <code>\DeclareTemplateInterfaceEquivalents:</code> New function 418 v1.0n lttemplates.dtx 2026-10-08 <code>__template_declare_template_keys:nnnn:</code> Reset all template data in interface declaration 388 <code>__template_parse_equiv_auxi:nnnn:</code> New function 392 <code>__template_parse_equiv_auxii:n:</code> New function 392 <code>__template_parse_equiv_auxiii:nnnn:</code> New function 392 <code>__template_parse_values_set:c:</code> Record value order 400 <code>__template_parse_values_set:e:</code> Record value order 400 <code>__template_recover_vars:nn:</code> Extend to cover messages 387 <code>__template_store_key_implementation:nnnn:</code> Omit error for equivalent key types 395 Reset all template data in interface declaration 395 <code>__template_store_vars:nn:</code> Extend to cover messages 386 General: New constant 381 New message 415 New variable 382 v1.0n lttemplates.dtx 2026-10-09 <code>__template_keys_set_known_aux:nnn:</code> New function 411 <code>__template_parse_known_values:nnn:</code> New function 399	<code>__template_parse_values:NNnnn:</code> New function 399 <code>__template_parse_values:nnn:</code> Support selectable behavior for unknown keys 399 <code>__template_parse_values_elt:NNn:</code> Permit missing value for boolean keys 400 <code>__template_parse_vars_elt_auxv:enn:</code> Remove l3keys creation 395 General: New variable 382, 383 <code>\SetTemplateKeys:</code> Switch to template-native approach to key setting 419 v1.0n lttemplates.dtx 2026-10-13 <code>\SetCurrentKnownTemplateKeys:</code> New command 420 v1.0n lttemplates.dtx 2026-10-14 <code>__template_convert_to_assignments_aux:nV:</code> Correct error message 405 <code>__template_store_key_implementation:nnn:</code> Retain variable information 395 v1.0n lttemplates.dtx 2026-10-15 <code>__template_implement_choice_elt_aux:nnn:</code> Remove l3keys creation 398 <code>__template_implement_choices:nn:</code> Remove l3keys creation 397 <code>__template_parse_values_elt_auxi:w:</code> Rename function 400 <code>__template_parse_values_meta:nn:</code> New function 400 <code>__template_parse_values_meta_elt:nnnn:</code> New function 400 v1.0o lttemplates.dtx 2026-10-29 <code>__template_parse_values_elt_auxiv:w:</code> Retain + when storing unknown keys 400
---	--

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
$\backslash!$	02:306, 02:308, 57:567
$\backslash!$	<u>38:214</u>
$\backslash"$ 04:520, 04:521, 21:246, 21:399, 21:440, 21:478, 21:491, 21:602, 21:634, 21:661, 21:669, 21:675, 21:679, 21:685, 21:689, 21:695, 21:701, 21:708, 21:709, 21:715, 21:719, 21:770, 21:816, 21:1272, 21:1290, 21:1296, 21:1300, 21:1306, 21:1310, 21:1316, 21:1322, 21:1329, 21:1330, 21:1336, 21:1340, 21:1342, 21:1449, 24:513, 33:135, 33:162, 54:873, 57:568	
$\backslash\#$	01:46, 01:59, 02:6, 02:14, 02:406, 06:893, 09:480, 24:500, 54:858, 57:551
$\backslash\$$	01:58, 02:4, 02:13, 06:892, 21:327, 21:465, 21:472, 21:588, 21:828, 21:835, 33:595, 33:602, 54:855, 57:552
$\backslash\$$	<u>1376</u>
$\backslash\%$	01:59, 01:89, 01:91, 01:111, 02:14, 02:404, 06:893, 21:514, 21:516, 24:502, 33:604, 33:724, 37:212, 50:1609, 50:1610, 54:876, 57:553
$\backslash\&$	01:58, 02:5, 02:13, 02:405, 06:892, 50:474, 54:856, 57:554
$\backslash'$	02:426, 21:247, 21:400, 21:442, 21:476, 21:488, 21:604, 21:614, 21:620, 21:622, 21:625, 21:627, 21:635, 21:641, 21:647, 21:649, 21:652, 21:654, 21:662, 21:666, 21:673, 21:677, 21:682, 21:687, 21:690, 21:692, 21:699, 21:704, 21:705, 21:712, 21:717, 21:720, 21:771, 21:818, 21:837, 21:839, 21:840, 21:841, 21:844, 21:846, 21:847, 21:848, 21:850, 21:851, 21:1266, 21:1287, 21:1294, 21:1298, 21:1303, 21:1308, 21:1311, 21:1313, 21:1320, 21:1325, 21:1326, 21:1333, 21:1338, 21:1341, 21:1349, 21:1350, 21:1397, 21:1398, 21:1403, 21:1404, 21:1415, 21:1416, 21:1421, 21:1422, 21:1450, 21:1451, 21:1465, 21:1466, 21:1467, 21:1468, 21:1481, 21:1482, 24:512, 29:932, 30:264, 33:125, 37:740, 38:254, 40:450, 40:471, 41:72, 54:835, 54:872, 57:569
$\backslash($	07:2621, 38:389
$\backslash($	<u>38:271</u>
$\backslash)$	02:426, 07:2643, 38:390
$\backslash)$	<u>38:271</u>
$\backslash*$	22:186, 24:505, 50:1290, 50:1421, 50:1535, 50:1611, 54:865
$\backslash*$	<u>38:251</u>
$\backslash+$	41:72
$\backslash+$	<u>1374</u>
$\backslash,$	02:307, 02:309, 30:517, 37:740, 38:7, 38:8, 38:40, 38:167, 38:169, 38:172, 38:197, 38:220, 38:221, 38:237, 54:874
$\backslash,$	<u>1415</u>
$\backslash-$	02:253, 06:24, 18:485, 18:576, 21:437, 21:438, 21:597, 21:812, 21:813, 24:507, 37:740, 40:449, 40:470, 41:72, 54:867, 57:224, 57:264
$\backslash-$	<u>06:927</u> , <u>112</u>
$\backslash.$	02:306, 02:308, 20:45, 20:115, 20:173, 21:248, 21:401, 21:473, 21:474, 21:497, 21:610, 21:611, 21:637, 21:638, 21:664, 21:772, 21:842, 21:849, 21:1271, 21:1353, 21:1354, 21:1363, 21:1364, 21:1373, 21:1374, 21:1391, 21:1452, 21:1453, 21:1489, 21:1490, 24:506, 33:137, 33:163, 54:866
$\backslash.$	<u>1403</u>
$\backslash\ldots/$ after (hook)	<u>1169</u>
$\backslash\ldots$ default	602
$\backslash$ code	51:4
$\backslash$ if	51:4
$\backslash$ ifnot	51:4
$\backslash$ pass-to-packages	51:22
$\backslash$ store	51:4
$\backslash$ usage	51:4
$\backslash/$	01:81, 06:25, 09:218, 09:293, 24:454, 24:508, 50:473, 54:868
$\backslash/$	<u>1370</u>
$\backslash$ after (hook)	1160, 1160, 1161, 1169
$\backslash$ before (hook)	1160, 1160, 1169
$\backslash$ end (hook)	1169
$\backslash:$	<u>38:214</u> , 38:224, 38:225, 38:226, 38:242, <u>38:252</u> , 38:252, 06:887, 06:888, 02:307, 02:309
$\backslash;$	02:307, 02:309, 30:511, 38:198
$\backslash;$	<u>38:214</u> , 895
$\backslash<$	21:598, 21:763, 24:503, 37:740, 41:71, 41:109, 54:863

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

$\backslash =$	21:249, 21:402, 21:479, 21:496, 21:737, 21:773, 21:1269, 21:1343, 21:1344, 21:1359, 21:1360, 21:1382, 21:1383, 21:1384, 21:1409, 21:1410, 21:1435, 21:1436, 21:1469, 21:1470, 21:1471, 21:1472, 21:1487, 21:1488, 21:1495, 21:1496, 29:932, 33:143, 40:450, 40:471, 41:71	$\backslash @ifstar$	81
$\backslash >$	21:595, 21:764, 24:504, 37:740, 38:226, 38:241, 38:242, 38:252, 41:71, 54:864	$\backslash @ifundefined$	81
$\backslash ?$	02:306, 02:308, 57:569	$\backslash @missingfileerror$	1098
$@@$ commands:		$\backslash @namedef$	81
$\backslash @@_normalize_type_R:w$	1442	$\backslash @nameuse$	81
$\backslash @@_normalize_type_R_aux:w$	1442	$\backslash @outputbox@append$	1233
$\backslash @@_normalize_type_r:w$	1442	$\backslash @outputbox@appendfootnotes$	1233
$\backslash @par$	438	$\backslash @outputbox@attachbottomfloats$..	1233
$\backslash @TeXversion$	6, 1	$\backslash @outputbox@attachfloats$	1233
$@botlist$ commands:		$\backslash @outputbox@attachtopfloats$	1233
$\backslash @botlist:$		$\backslash @outputbox@reinsertbskip$	1233
.. 54:1380, 54:1475, 54:1624, 54:1778		$\backslash @restorepar$	438
$@botnum$ commands:		$\backslash @setpar$	438
$\backslash @botnum:$	54:1354	$@topnum$ commands:	
$\backslash @car$	82	$\backslash @topnum:$	54:1400
$\backslash @cdr$	82	$\backslash [$	24:509, 29:44, 29:55, 29:77, 29:88, 38:391, 54:869, 57:570
$@citeb$ commands:		$\backslash [$	38:306, 38:555, 1405
$\backslash @citeb:$	47:36, 47:65, 47:82	$\backslash \backslash$	01:58, 01:231, 01:232, 01:233, 01:234, 01:237, 01:244, 01:245, 01:246, 01:247, 01:250, 01:257, 01:258, 01:259, 01:260, 01:263, 01:270, 01:276, 01:277, 01:281, 01:283, 01:284, 01:288, 01:293, 01:294, 01:297, 01:303, 02:13, 02:252, 04:271, 04:425, 06:272, 06:380, 06:541, 06:664, 06:682, 06:892, 07:472, 07:532, 07:563, 07:713, 07:725, 07:756, 07:1158, 07:1164, 07:2834, 07:2862, 07:2888, 07:2895, 07:2924, 07:3021, 07:3049, 07:3060, 07:3086, 07:3092, 07:3099, 08:1799, 08:1810, 08:2628, 08:2642, 08:2643, 08:2644, 08:2645, 08:2646, 08:2651, 08:2654, 08:2655, 08:2656, 08:2666, 08:2667, 08:2681, 08:2702, 08:2706, 08:2707, 08:2709, 08:2715, 08:2726, 08:2727, 08:2732, 08:2737, 08:2742, 08:2758, 08:2923, 08:2924, 08:2925, 09:477, 09:615, 11:1319, 11:1332, 11:1338, 11:1339, 11:1340, 11:1341, 11:1354, 11:1360, 11:1361, 11:1373, 11:1382, 11:1388, 11:1389, 11:1396, 11:1403, 11:1410, 11:1417, 11:1424, 11:1425, 11:1434, 11:1458, 11:1459, 11:1477, 11:1478, 11:1479, 11:1502, 11:1503, 11:1504, 11:1505, 11:1506, 11:1507, 11:1508, 11:1509, 11:1510, 11:1511, 11:1512, 14:279, 16:148, 16:155, 16:162, 16:163, 16:164, 16:165, 18:578, 20:773, 20:792, 21:583, 24:497, 30:262, 36:271, 37:453, 37:458, 37:463, 37:473, 37:477, 37:481, 37:522,
$@colht$ commands:			
$\backslash @colht:$	54:532, 54:545		
$@colnum$ commands:			
$\backslash @colnum:$	54:1540, 54:1685, 54:1852		
$\backslash @cons$	82		
$\backslash @currdir$	5, 1		
$@currname$ commands:			
$\backslash @currname:$	20:780		
$\backslash @dblarg$	81		
$@dbldeferlist$ commands:			
$\backslash @dbldeferlist:$	54:2332, 54:2374		
$@dbltopnum$ commands:			
$\backslash @dbltopnum:$	54:2231, 54:2358		
$@deferlist$ commands:			
$\backslash @deferlist:$	54:1466, 54:1563, 54:1615, 54:1708, 54:1768, 54:1876, 54:1921, 54:1950, 54:2002, 54:2031, 54:2088, 54:2120, 54:2206, 54:2246		
$\backslash @if@bottomfloats@TF$	1233		
$\backslash @if@footnotes@TF$	1233		
$\backslash @ifclasslater$	1097		
$\backslash @ifclassloaded$	1097		
$\backslash @ifclasswith$	1097		
$\backslash @ifdefinable$	81		
$\backslash @ifnextchar$	81		
$\backslash @ifpackagelater$	1097		
$\backslash @ifpackageloaded$	1097		
$\backslash @ifpackagewith$	1097		

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

38:432, 38:461, 38:616, 40:482, 40:689, 40:691, 41:73, 41:174, 41:181, 41:208, 41:215, 41:246, 41:269, 42:150, 54:852, 57:555, 463	37:204, 37:213, 37:507, 37:508, 37:662, 50:1291, 50:1292, 50:1293, 50:1374, 50:1377, 50:1380, 50:1422, 50:1423, 50:1424, 50:1506, 50:1509, 50:1512, 50:1536, 50:1537, 50:1538, 50:1596, 50:1599, 50:1602, 54:857, 54:859, 54:862, 57:255, 57:256, 57:257, 57:258, 57:259, 57:260, 57:261, 57:262, 57:263, 57:556, 57:562, 57:563, 57:564, 57:565, 57:599, 57:600, 57:601, 57:602, 57:603, 57:604, 57:605, 57:606, 57:607
<code>\%</code> 18:53, 1379	<code>_</code> 21:334, 30:265, 38:269, 38:270, 02:8, 02:14, 06:893, 54:860, 57:557, 01:59, 1383
<code>\{</code> 01:6, 01:10, 01:58, 02:2, 02:13, 07:635, 07:2163, 09:478, 14:22, 21:328, 21:585, 24:498, 30:260, 37:521, 38:59, 38:167, 54:853, 57:558	<code>_hook\textvisiblespace_meta_hook}</code> 248
<code>\}</code> 01:11, 01:58, 02:3, 02:13, 09:479, 14:21, 21:329, 21:586, 24:499, 30:261, 37:521, 38:59, 54:854, 57:559	<code>_hook_next\textvisiblespace_meta_hook}</code> 248
<code>\%</code> 1229	<code>_hook_toplevel\textvisiblespace_meta_hook}</code> 248
<code>\<addto-cmd></code> 219	<code>\%</code> 21:252, 21:404, 21:441, 21:475, 21:487, 21:603, 21:665, 21:672, 21:676, 21:681, 21:686, 21:691, 21:698, 21:702, 21:703, 21:711, 21:716, 21:775, 21:817, 21:1265, 21:1286, 21:1293, 21:1297, 21:1302, 21:1307, 21:1312, 21:1319, 21:1323, 21:1324, 21:1332, 21:1337, 24:511, 29:932, 33:139, 37:740, 40:450, 40:471, 41:72, 54:871, 57:572
<code>\<cmd>_</code> 163	<code>\ </code> 21:584, 22:336, 22:347, 30:584, 30:585, 57:573
<code>\<cmd>_(arg_(num))</code> 165	<code>\~</code> 01:59, 02:10, 02:14, 06:893, 14:20, 18:491, 21:259, 21:307, 21:405, 21:490, 21:582, 21:668, 21:680, 21:684, 21:694, 21:710, 21:714, 21:776, 21:1268, 21:1285, 21:1289, 21:1301, 21:1305, 21:1315, 21:1331, 21:1335, 21:1379, 21:1380, 21:1381, 21:1433, 21:1434, 33:151, 33:171, 37:655, 37:672, 37:686, 37:701, 37:744, 54:875, 57:560
<code>\<cmd>_</code> 163	A
<code>\<cmd>_code</code> 163	<code>\A</code> 57:252, 57:575, 57:596
<code>\<cmd>_defaults</code> 163	<code>\a</code> 21:243, 41:1, 57:243, 57:576, 57:587, 1386
<code>\<filename></code> 1175	<code>A/foo (hook)</code> 227, 227
<code>\<function></code> 191	<code>\AA</code> ... 21:260, 21:449, 21:549, 02:318, 1400
<code>_</code> .. 01:58, 01:75, 02:13, 02:256, 02:312, 02:341, 02:342, 02:359, 06:892, 07:1836, 07:2201, 08:2707, 11:1361, 11:1389, 11:1397, 11:1404, 11:1425, 11:1459, 14:19, 14:20, 14:21, 14:22, 14:25, 18:490, 24:494, 24:745, 24:782, 24:807, 30:263, 37:505, 37:506, 37:515, 37:516, 37:661, 37:677, 37:691, 43:67, 43:69, 43:74, 43:76, 47:37, 50:467, 54:861, 57:550	<code>\aa</code> ... 21:265, 21:443, 21:559, 02:318, 1398
<code>\]</code> ... 02:426, 24:510, 38:392, 54:870, 57:571	<code>\abovedisplayshortskip</code> .. 38:624, 02:288
<code>\]</code> 38:306, 38:579	<code>\abovedisplayskip</code> 38:617, 38:619, 38:621, 38:622, 38:623, 38:624, 02:287, 1384
<code>\^</code> .. 01:47, 01:56, 01:59, 01:103, 01:314, 02:7, 02:9, 02:11, 02:14, 02:255, 02:312, 02:313, 02:332, 02:333, 02:354, 02:355, 06:20, 06:893, 07:6, 07:7, 07:1615, 07:1618, 07:1622, 07:1658, 07:1707, 07:1746, 07:1810, 07:1834, 07:2102, 07:2202, 07:2489, 07:2494, 07:3330, 18:578, 18:580, 18:582, 21:251, 21:306, 21:403, 21:477, 21:489, 21:581, 21:667, 21:674, 21:678, 21:683, 21:688, 21:693, 21:700, 21:706, 21:707, 21:713, 21:718, 21:774, 21:1267, 21:1284, 21:1288, 21:1295, 21:1299, 21:1304, 21:1309, 21:1314, 21:1321, 21:1327, 21:1328, 21:1334, 21:1339, 21:1351, 21:1352, 21:1369, 21:1370, 21:1377, 21:1378, 21:1392, 21:1393, 21:1394, 21:1423, 21:1424, 21:1445, 21:1446, 21:1447, 21:1448, 24:495, 24:496, 24:501, 33:133, 33:161,	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \abspage 854
- abspage 36:255, 854
- \accent 21:91,
21:414, 21:444, 21:502, 21:787, 1401
- \ActivateGenericHook
08:2782, 08:2784, 08:2789, 08:2791, 236
- \active 01:103, 37:505, 37:506,
37:507, 37:508, 37:515, 37:516,
37:654, 37:661, 37:662, 37:671,
37:677, 37:685, 37:691, 37:700,
37:742, 38:254, 38:269, 01:314,
02:10, 02:11, 50:1291, 50:1292,
50:1293, 50:1374, 50:1377, 50:1380,
50:1422, 50:1423, 50:1424, 50:1506,
50:1509, 50:1512, 50:1536, 50:1537,
50:1538, 50:1596, 50:1599, 50:1602,
54:835, 01:48, 02:332, 02:333,
02:341, 02:342, 02:354, 02:355, 02:359
- \acute 30:527
- add commands:
- add_to_callback 04:841
- \add_to_callback 44
- \addcontentsline
..... 44:70, 44:80, 44:159, 45:16, 870
- \AddEverypageHook 1191
- \Adding 1359
- \adpenalty 18:307,
39:124, 39:201, 39:206, 44:50,
54:343, 54:1505, 54:1654, 54:1815, 476
- \AddThispageHook 1191
- \addtocontents
.. 44:164, 44:171, 44:177, 44:188, 870
- \addtocounter 556
- \addtocounter 22:6, 22:35, 1371
- \AddToDocumentProperties 17:27, 458
- \AddToHook 08:2794, 08:2944,
26:153, 37:44, 37:45, 37:46, 37:109,
37:110, 37:111, 37:398, 37:399,
37:400, 37:401, 47:72, 50:1171,
50:1172, 50:1173, 52:561, 52:563,
52:573, 52:574, 52:575, 52:576,
53:251, 53:547, 53:570, 55:201, 220
- \AddToHookNext 08:2805, 08:2942,
17:177, 52:562, 53:9, 53:571, 1191
- \AddToHookNextWithArguments 08:2805, 222
- \AddToHookWithArguments ... 08:2794, 221
- \addtolength 569
- \addtolength 23:16, 38:619, 38:621
- \AddToNoCaseChangeList 57:620, 1426
- \addtoversion 27:20, 27:139
- \advspace 18:256, 37:431, 39:124, 39:202,
39:203, 39:207, 39:255, 44:50, 476
- \adjdemerits 02:206
- \AE 21:261, 21:419, 21:550, 21:792,
21:1155, 21:1465, 21:1469, 57:719
- \ae 21:266, 21:422, 21:560, 21:796,
21:1161, 21:1467, 21:1471, 57:719
- after (hook) 241
- \afterassignment 21:232, 21:240,
24:391, 06:360, 06:366, 38:199,
06:409, 40:336, 02:376, 02:379, 96
- \AfterEndEnvironment 37:394, 241
- \aftergroup 24:92, 24:410, 24:428,
24:433, 26:203, 26:269, 28:115,
28:122, 28:130, 28:476, 32:65,
37:658, 37:675, 37:689, 37:704,
38:538, 38:539, 39:149, 39:154,
40:192, 40:339, 54:884, 54:885,
54:959, 54:960, 54:1017, 54:1018, 1380
- \AfterLastShipout 52:573
- \afterpreamble 489
- \aleph 30:319
- \alignmark 1440
- \allocationnumber 04:52,
04:53, 04:54, 04:91, 04:212, 16:81,
41:4, 41:9, 02:37, 53:95, 02:53,
02:65, 02:67, 57:58, 57:59, 57:60,
02:97, 02:98, 02:99, 02:118, 02:119,
02:157, 02:163, 02:169, 02:170, 451
- \allowbreak 38:40,
06:969, 06:970, 06:989, 06:991, 02:383
- \Alph 556
- \Alph 22:307, 1189
- \alph 556
- \alph 22:306
- \alpha 30:279
- \amalg 30:390
- \AmSfont 657
- \and 1012
- \and 44:14, 44:27
- \angle 30:348
- \approx 30:433
- \arabic 556
- \arabic 22:207, 22:222, 22:260,
22:272, 22:303, 43:60, 53:397, 1189
- \arccos 38:13
- \arcsin 38:10
- \arctan 38:16
- \arg 38:26
- array (env.) 41:168
- \array 41:168, 1441
- \arraycolsep ... 38:435, 38:436, 38:464,
38:465, 38:629, 38:630, 41:335, 41:415
- \arrayrulewidth
... 41:401, 41:415, 41:423, 41:424,
41:436, 41:440, 41:443, 41:453, 41:455

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\arraystretch</code>	<code>\author</code>
41:238, 41:239, 41:261, 41:262, 41:419	1012
<code>\Arrowvert</code>	<code>\author</code>
30:580	44:8, 44:24, 44:32
<code>\arrowvert</code>	
30:578	
<code>\asciispace</code>	B
37:578, 37:580, 37:583,	<code>\b</code>
37:597, 37:608, 37:609, 37:622, 37:623	21:253,
<code>\AssignSocketPlug</code> ...	21:410, 21:498, 21:783, 21:1276, 1386
10:192, 10:218,	<code>babel/⟨language⟩/afterextras</code> (hook) 236
35:138, 54:721, 54:728, 55:24, 355	<code>\backslash</code>
<code>\AssignStructureRole</code>	30:262, 30:601
1310	<code>\bar</code>
<code>\AssignStructureRole</code> 55:110, 55:111, 55:304	30:531
<code>\AssignTaggingSocketPlug</code>	<code>\baselineskip</code>
1309	26:187,
<code>\AssignTaggingSocketPlug</code>	26:188, 26:189, 26:191, 26:192,
55:20, 55:61, 55:71, 55:82,	30:521, 38:171, 38:172, 38:191,
55:83, 55:91, 55:99, 55:105, 55:106,	38:197, 38:201, 40:459, 40:478,
55:116, 55:143, 55:144, 55:168,	40:712, 41:250, 41:273, 42:147,
55:169, 55:203, 55:204, 55:212, 1309	42:325, 42:384, 53:283, 53:334,
<code>\AssignTemplateKeys</code>	54:247, 54:278, 54:907, 54:929,
11:538, 11:543, 11:1619, 372	54:977, 54:992, 54:1036, 54:1051,
<code>\ast</code>	02:284, 02:311, 02:374, 02:410, 1358
30:243, 30:406	<code>\baselinestretch</code> 24:382, 26:121, 26:122,
<code>\asymp</code>	26:167, 26:168, 26:185, 26:246, 1357
30:460	<code>\baselinstretch</code>
<code>\AtBeginDocument</code>	1377
20:126,	<code>\batchmode</code>
20:181, 25:3257, 47:54, 50:1162, 668	20:710, 20:741,
<code>\atbegindocumenthook</code>	20:742, 27:106, 05:94, 05:100,
489	05:110, 05:144, 57:749, 57:770, 505
<code>\AtBeginDvi</code> ...	<code>\BCPdata</code>
53:465, 53:501, 54:62, 1187	57:609, 57:636, 57:640
<code>\AtBeginEnvironment</code>	<code>\BeforeBeginEnvironment</code>
37:394, 241	37:394, 241
<code>\AtBeginShipout</code>	<code>\BeforeClearDocument</code>
53:506, 53:570, 1191	52:577
<code>\AtBeginShipoutAddToBox</code> ...	<code>\begin</code>
53:575, 1190	09:82,
<code>\AtBeginShipoutAddToBoxForeground</code> ..	14:246, 14:248, 20:364, 20:416,
53:575, 1190	26:7, 30:4, 31:4, 37:244, 37:245,
<code>\AtBeginShipoutBox</code>	37:315, 37:337, 37:362, 37:378,
53:568, 1190	37:386, 38:559, 38:571, 44:14, 44:17,
<code>\AtBeginShipoutDiscard</code>	06:801, 50:779, 51:283, 53:447,
53:574, 1191	56:3, 07:1465, 07:1582, 07:2876, 241
<code>\AtBeginShipoutFirst</code> 53:509, 53:572, 1191	<code>begindocument</code> (hook) 327, 354, 354, 489,
<code>\AtBeginShipoutInit</code>	489, 1204, 1208, 239, 239, 239, 242, 250
53:569, 1191	<code>begindocument/before</code> (hook)
<code>\AtBeginShipoutNext</code> . 53:507, 53:570, 1425	489, 242
<code>\AtBeginShipoutOriginalShipout</code>	<code>begindocument/end</code> (hook)
53:583, 1190	242
<code>\AtBeginShipoutUpperLeft</code> ...	<code>\begingroup</code>
53:575, 1190	874
<code>\AtBeginShipoutUpperLeftForeground</code> .	<code>\belowdisplayshortskip</code> ..
53:575, 1190	38:623, 02:290
<code>\AtEndAfterFileList</code>	<code>\belowdisplayskip</code>
52:575	38:622, 02:289
<code>\AtEndDocument</code> 37:146, 50:1162, 52:579, 242	<code>\beta</code>
<code>\AtEndDvi</code>	30:280
53:516, 53:542, 1187	<code>\bezier</code>
<code>\AtEndEnvironment</code>	974
37:394, 241	<code>\bezier</code>
<code>\AtEndOfClass</code>	42:692,
38:554, 50:1162, 1162	42:693, 42:815, 42:816, 42:831, 42:833
<code>\AtEndOfPackage</code>	<code>\bf</code>
50:641, 50:660,	1365
50:758, 50:769, 50:1162, 51:88, 1162	<code>\bfdefault</code> 29:16, 29:340, 29:346, 29:347,
<code>\AtEndPreamble</code>	29:348, 29:349, 29:389, 29:390,
242	29:391, 29:392, 29:401, 29:437,
<code>\AtNextShipout</code>	29:461, 29:480, 29:521, 29:555,
1191	30:104, 30:115, 30:117, 30:125, 1423
<code>\atopwithdelims</code>	<code>bfseries</code> (hook)
38:57, 38:58, 38:59	244, 244
<code>\attribute</code>	<code>\bfseries</code>
04:79, 40	29:14, 29:15, 29:332,
<code>\attributedef</code>	29:333, 29:381, 29:386, 29:387,
04:79, 04:222	
<code>\attributezero</code>	
04:222	
<code>\AtVeryEndDocument</code>	
52:574	
<code>\AtVeryVeryEnd</code>	
52:576	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 29:428, 29:431, 29:432, 29:457,
29:459, 29:460, 29:553, 29:554,
29:706, 29:707, 32:19, 35:17, 35:34,
35:51, 43:67, 43:69, 43:74, 43:76,
47:40, 53:448, 57:786, 57:787, 740
- bfseries 29:499
- bfseries/defaults (hook) 244, 244
- bfseries/defaults 29:499
- \bgroup 02:325, 1358
- \bibtex 47:7, 47:9, 47:10, 1391
- \bibdata 47:45, 47:49
- \bibitem 47:3
- \bibliography 1053
- \bibliography 47:47
- \bibliographystyle 1053
- \bibliographystyle 47:52
- \bibstyle 47:45, 47:57
- \Big 30:634, 30:637,
30:646, 30:648, 38:44, 38:45, 38:46
- \big 30:635, 30:647, 38:41
- \bigbreak 06:971, 06:992, 02:390
- \bigcap 30:356
- \bigcirc 30:403
- \bigcup 30:357
- \Bigg ... 30:641, 30:650, 38:50, 38:51, 38:52
- \bigg ... 30:639, 30:649, 38:47, 38:48, 38:49
- \Biggl 38:50
- \biggl 38:47
- \Biggm 38:51
- \biggm 38:48
- \Biggr 38:52
- \biggr 38:49
- \Bigl 38:44
- \bigl 38:41
- \Bigm 38:45
- \bigm 38:42
- \bigodot 30:364
- \bigoplus 30:363
- \bigotimes 30:362
- \Bigr 38:46
- \bigr 38:43
- \bigskip 18:469, 02:395
- \bigskipamount
..... 18:471, 18:472, 45:391, 02:394
- \bigscup 30:367
- \bigtriangledown 30:372, 30:373
- \bigtriangleup 30:371, 30:374
- \biguplus 30:355
- \bigvee 30:353
- \bigwedge 30:354
- \binoppenalty 02:193
- block/list/label (tag socket) 55:218
- block/recipe (tag socket) 55:219
- \bmod 38:35
- \boldmath 19:14, 29:724
- bool commands:
- \bool_gset_false:N .. 10:16, 36:31,
48:348, 53:76, 53:142, 53:151, 08:15
- \bool_gset_true:N ... 10:11, 36:33,
48:343, 53:71, 53:181, 53:391, 08:10
- \bool_if:NTF 08:1945, 08:1954,
08:2045, 08:2054, 09:178, 09:182,
09:194, 09:253, 09:257, 09:269,
09:413, 10:22, 10:24, 11:317, 11:323,
11:465, 11:967, 11:1083, 11:1132,
11:1149, 11:1171, 36:71, 48:354,
51:24, 51:246, 53:82, 53:149, 53:417,
07:125, 07:134, 07:146, 07:147,
07:161, 07:173, 07:177, 07:186,
07:188, 07:198, 07:207, 07:208,
07:211, 07:216, 07:218, 07:322,
55:172, 55:180, 55:192, 55:195,
07:467, 07:476, 07:478, 07:529,
07:560, 07:575, 07:624, 07:665,
07:710, 07:720, 07:753, 07:761,
07:770, 07:780, 07:785, 07:812,
07:832, 07:914, 07:960, 07:961,
07:1031, 07:1044, 07:1061, 07:1081,
07:1759, 07:2203, 07:2214, 07:2464,
07:2796, 07:2837, 08:21, 08:334, 214
- \bool_if_exist:NTF 36:27
- \bool_lazy_all:nTF 07:966
- \bool_lazy_and:nnTF
..... 08:1947, 08:2047, 08:2497,
08:2965, 53:127, 53:173, 53:428,
07:749, 07:766, 07:962, 07:2555, 08:269
- \bool_lazy_and_p:nn .. 08:2500, 07:116
- \bool_lazy_any:nTF
..... 11:1034, 07:112, 07:487
- \bool_lazy_or:nnTF 08:1898, 11:567,
11:1253, 07:77, 07:1425, 07:1434,
07:2619, 07:2641, 07:2826, 08:908
- \bool_new:N 10:6, 11:78, 11:79,
36:28, 48:339, 51:57, 53:67, 07:19,
53:249, 53:264, 07:21, 07:23, 07:34,
07:35, 07:37, 07:38, 07:40, 07:43,
07:46, 07:47, 07:48, 07:49, 08:6, 08:24
- \bool_set_false:N
..... 08:1937, 08:2037, 11:368,
11:965, 11:1050, 51:72, 07:61, 07:87,
07:236, 07:258, 07:303, 55:104,
55:112, 07:453, 07:454, 07:455,
07:456, 07:457, 07:458, 07:479,
07:537, 07:567, 07:592, 07:606,
07:628, 07:629, 07:630, 07:649,
07:716, 07:759, 07:773, 07:774,
07:799, 07:800, 07:801, 07:803,
07:807, 07:809, 07:823, 07:979,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- 07:980, 07:981, 07:982, 07:1601, 07:2089
`\bool_set_true:N` . 08:1934, 08:2034,
 11:339, 11:381, 11:781, 11:1059,
 51:69, 07:66, 07:237, 07:259, 07:292,
 55:102, 55:114, 07:452, 07:477,
 07:544, 07:551, 07:552, 07:566,
 07:762, 07:763, 07:819, 07:839,
 07:840, 07:846, 07:847, 07:853,
 07:854, 07:861, 07:862, 07:863,
 07:998, 07:1016, 07:1606, 07:2094
`\bool_while_do:nn` . . . 08:1614, 08:1699
`\c_false_bool`
 . . 09:157, 09:167, 09:403, 07:2465,
 07:2768, 07:3101, 07:3269, 08:391,
 08:397, 08:405, 08:430, 08:478, 214
`\l_tmpa_bool` 373
`\c_true_bool`
 09:162, 07:2466, 07:2770, 07:3102,
 07:3267, 08:398, 08:406, 08:408, 337
 bool internal commands:
`\g__mark_debug_bool`
 48:339, 48:343, 48:348, 48:354
`\BooleanFalse` 07:2082, 07:2458, 07:3101, 121
`\BooleanTrue` 07:2081, 07:2457, 07:3101, 121
`\bordermatrix` 38:185
`\bot` 30:332
`\botfigrule` 54:1093, 54:3026
`\botmark` . . . 49:119, 54:1002, 54:1061, 1058
`\bottomfraction` 45:275, 54:2995
`\bowtie` 30:493
`\Box` 29:882
`\box` 443
 box commands:
`\box_dp:N` 53:255
`\box_gclear:N` 16:91
`\box_gset_to_last:N` 16:16, 16:49, 16:119
`\box_ht:N` 53:254, 53:363, 1196
`\box_if_empty:NTF` 48:88, 53:139, 53:158
`\box_if_horizontal:NTF` 53:295, 53:347
`\box_if_vertical:NTF`
 . 48:83, 48:418, 48:450, 53:269, 53:320
`\box_move_up:nn` 53:309, 53:363
`\box_new:N` 16:84, 48:63,
 48:64, 53:84, 53:86, 53:248, 53:265
`\box_set_dp:Nn` 53:37, 53:57, 53:282,
 53:291, 53:308, 53:333, 53:344, 53:362
`\box_set_eq:NN` . 53:210, 53:240, 53:245
`\box_set_eq_drop:NN` 53:154
`\box_set_ht:Nn` 53:36, 53:56, 53:281,
 53:290, 53:307, 53:332, 53:343, 53:361
`\box_set_to_last:N` 48:80
`\box_set_wd:Nn`
 53:280, 53:306, 53:331, 53:360
`\box_use:N` 53:38, 53:58,
 53:187, 53:286, 53:311, 53:339, 53:364
`\box_use_drop:N` 16:86
`\box_wd:N` 53:256, 53:357, 53:365
 box internal commands:
`\l__mark_box` . . 48:63, 48:76, 48:80,
 48:83, 48:84, 48:87, 48:88, 48:98, 1072
`\l__mark_ii_box` . . 48:63, 48:87, 48:97
`\boxmaxdepth`
 . . . 42:485, 42:513, 42:543, 42:621,
 42:638, 54:603, 54:742, 54:1075, 02:270
`\brace` 38:59
`\bracerd`
 30:564, 30:568, 30:569, 30:571, 30:573
`\bracelu` 30:566, 30:570, 30:572
`\bracerd` 30:565, 30:570, 30:572
`\braceru` 30:567, 30:569, 30:573
`\bracevert` 30:619
`\brack` 38:58
`\break` 18:116,
 06:972, 06:993, 02:383, 02:388, 1391
`\breve` 30:532
`\brokenpenalty` 24:765, 02:198
 build/column/after (hook) . . 54:731, 1232
 build/column/baselineattach (socket) .
 54:723,
 1233, 1234, 1246, 1247, 1250, 1250
 build/column/before (hook) . 54:731, 1232
 build/column/footins (socket) 1234
 build/column/footins (tag socket) . . 55:269
 build/column/footnotes (socket)
 54:722, 1234
 build/column/outputbox (socket)
 54:663, 1233, 1234, 1246
 build/column/outputbox (tag socket) 55:268
 build/page/after (hook) . . . 54:729, 1232
 build/page/before (hook) . . . 54:729, 1232
 build/page/footer (socket) 1234
 build/page/footer (tag socket) . . . 55:266
 build/page/header (socket) 1234
 build/page/header (tag socket) . . . 55:266
 build/page/reset (hook) 54:730, 1232
`\bulldrel` 30:480, 38:162
`\bullet` 30:392

C

`\c` 11:534, 21:254, 21:355, 21:357,
 21:359, 21:361, 21:363, 21:365,
 21:367, 21:369, 21:371, 21:392,
 21:394, 21:413, 21:481, 21:501,
 21:629, 21:631, 21:656, 21:658,
 21:671, 21:697, 21:724, 21:727,
 21:728, 21:729, 21:730, 21:731,
 21:732, 21:733, 21:734, 21:735,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- 21:786, 21:1278, 21:1292, 21:1318,
 21:1375, 21:1376, 21:1395, 21:1396,
 21:1399, 21:1400, 21:1405, 21:1406,
 21:1417, 21:1418, 21:1425, 21:1426,
 21:1429, 21:1430, 33:131, 33:160, 1407
 \cal 29:933
 call commands:
 call_callback 04:806
 \call_callback 44
 callback commands:
 callback_identifiers 04:997
 callback.register 04:694
 \callback_descriptions 44
 \callback_identifiers 44
 \cap 30:383
 \capitalacute 21:863, 33:124,
 33:125, 33:157, 33:654, 33:904, 33:1270
 \capitalbreve 21:870, 33:126,
 33:127, 33:158, 33:655, 33:911, 33:1277
 \capitalcaron 21:869, 33:128,
 33:129, 33:159, 33:656, 33:910, 33:1276
 \capitalcedilla 21:856, 33:130,
 33:131, 33:160, 33:657, 33:901, 33:1280
 \capitalcircumflex ... 21:864, 33:132,
 33:133, 33:161, 33:658, 33:905, 33:1271
 \capitaldieresis 21:866, 33:134,
 33:135, 33:162, 33:659, 33:907, 33:1273
 \capitaldotaccent ... 21:872, 33:136,
 33:137, 33:163, 33:660, 33:913, 33:1279
 \capitalgrave 21:862, 33:138,
 33:139, 33:164, 33:661, 33:903, 33:1269
 \capitalhungarumlaut . 21:867, 33:140,
 33:141, 33:165, 33:662, 33:908, 33:1274
 \capitalmacron 21:871, 33:142,
 33:143, 33:166, 33:663, 33:912, 33:1278
 \capitalnewtie
 ... 21:876, 33:154, 33:155, 33:167,
 33:664, 33:978, 33:979, 33:1285, 803
 \capitalogonek 21:859, 33:144,
 33:145, 33:168, 33:665, 33:902, 33:1281
 \capitalring 21:868, 33:146,
 33:147, 33:169, 33:666, 33:909, 33:1275
 \capitaltie 21:874, 33:148, 33:149,
 33:170, 33:667, 33:974, 33:975, 33:1283
 \capitaltilde 21:865, 33:150,
 33:151, 33:171, 33:668, 33:906, 33:1272
 \caption 45:4, 121
 caption/begin (tag socket) 55:262
 caption/end (tag socket) 55:262
 caption/label/begin (tag socket) .. 55:264
 caption/label/end (tag socket) ... 55:264
 \cases 38:166, 38:167, 38:177, 38:179
 \CaseSwitch 57:620, 1426
 \catcode 1442
 \catcodetable 04:89, 04:109, 40
 \catcodetable@atletter 41
 \catcodetable@initex 41
 \catcodetable@latex 41
 \catcodetable@string 41
 \catcoding 1361
 \cdot 30:405
 \cdotp 30:513, 30:519
 \cdots 30:519
 center (env.) 37:446
 \center 37:446
 \centering 37:446, 37:451,
 37:452, 37:470, 37:472, 37:487, 37:489
 \centerline 40:725
 \changes 29:794, 35:201, 1401
 \chapter 557
 \chaptermark 1064
 \char ... 21:412, 21:415, 21:451, 21:454,
 21:465, 21:472, 21:500, 21:504,
 21:509, 21:512, 21:514, 21:516,
 21:757, 21:785, 21:788, 21:821,
 21:828, 21:835, 21:858, 21:861,
 21:890, 21:920, 21:1030, 21:1069,
 21:1193, 21:1195, 21:1197, 21:1244,
 29:737, 29:744, 33:595, 33:602,
 33:604, 33:724, 37:578, 37:745,
 38:251, 42:243, 42:293, 42:307,
 42:315, 42:318, 42:459, 42:573,
 42:578, 42:586, 42:590, 42:626,
 42:627, 42:629, 42:642, 42:643,
 42:646, 42:673, 06:934, 06:949, 1394
 char commands:
 \char_generate:nn
 05:159, 07:1911, 07:2223
 \char_set_catcode_active:N . 07:2489
 \char_set_catcode_active:n
 07:2201, 07:2202, 07:2666
 \char_set_catcode_escape:N .. 09:477
 \char_set_catcode_group_begin:N .
 09:478
 \char_set_catcode_group_end:N 09:479
 \char_set_catcode_other:N
 07:1616, 07:1622,
 07:1658, 07:1707, 07:1746, 07:1810,
 07:1832, 07:1834, 07:2199, 07:2212
 \char_set_catcode_other:n
 07:1618, 07:2204, 07:2215
 \char_set_catcode_parameter:N 09:480
 \char_set_catcode_parameter:n ...
 07:2205, 07:2216
 \char_set_catcode_space:n
 09:415, 07:1836
 \char_set_lccode:nn . 07:2494, 07:2676
 \char_value_catcode:n 09:408

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`\chardef` 04:22, 04:26, 04:38, 04:47, 04:48,
 04:89, 12:2, 04:157, 04:223, 20:52,
 20:122, 21:36, 24:15, 41:4, 41:9,
 02:10, 02:16, 02:17, 02:18, 02:19,
 02:20, 50:1297, 50:1428, 50:1545,
 02:54, 02:60, 02:62, 02:69, 02:75,
 02:78, 02:80, 02:86, 02:87, 02:90,
 02:92, 57:42, 57:44, 57:48, 57:67,
 57:171, 57:172, 57:173, 57:174,
 57:175, 57:176, 57:177, 02:117,
 02:143, 02:145, 02:169, 01:48, 01:54,
 01:55, 02:404, 02:405, 02:406, 1374
`\charsubdef` 57:402
`\charzero` 04:223
`\check` 30:533
`\CheckCommand` 06:228, 1378
`\CheckEncodingSubset`
 24:204, 33:16, 33:73, 33:74, 33:75,
 33:120, 33:122, 33:316, 33:585,
 33:794, 33:843, 33:899, 33:900,
 33:968, 33:1085, 33:1088, 33:1102, 799
`\chi` 30:299
`\choose` 38:57
`\circ` 30:402
`\circle` 42:461, 42:615, 42:817, 42:834, 1361
`\citation` 47:11, 47:39, 47:67, 47:84
`\cite` 1053
`\cite` 47:12, 1054
`\clap` 40:729, 1415
`class` (hook) 236
`class/` (hook) 1162, 1169
`class/.../after` 1161
`class/.../before` 1161
`class/<name>/after` (hook) 1161
`class/<name>/before` (hook) 1161
`class/after` (hook) 1161
`class/after` 1161
`class/before` (hook) 1161
`class/before` 1161
`\ClassError` 14:84
`\ClassInfo` 14:84
`\ClassNote` 14:136, 1422
`\ClassNoteNoLine` 14:136
`\ClassWarning` 14:84
`\ClassWarningNoLine` 14:84
`\cleaders` 30:559, 30:562, 02:424
`\cleardoublepage` 54:117
`\ClearHookNext` 08:2816, 222
`\ClearHookRule` 08:2853, 08:2958, 226
`\clearpage` 20:316, 20:343,
 20:347, 20:378, 20:401, 20:404,
 20:425, 20:443, 37:17, 37:85, 37:149,
 37:242, 54:103, 54:117, 54:122,
 54:205, 54:412, 54:415, 54:419,
 54:460, 54:466, 54:2847, 54:2864, 1163
`\cline` 41:444, 972
clist commands:
`\clist_clear:N` .. 51:65, 51:166, 51:186
`\clist_gclear:N` 08:1613, 08:1698
`\clist_gput_left:Nn` . 08:1552, 08:1517
`\clist_gput_right:Nn` 08:1556, 08:1519
`\clist_if_empty:NTF`
 08:1964, 08:2065, 11:689, 11:718, 51:75
`\clist_if_exist:NTF` 08:115
`\clist_if_in:NnTF` . 11:357, 11:725,
 51:51, 51:52, 51:128, 51:151, 51:153
`\clist_map_function:nN` 36:59
`\clist_map_inline:Nn` 11:691, 51:77,
 51:109, 51:140, 51:175, 51:189, 51:234
`\clist_map_inline:nn`
 11:903, 51:63, 51:269,
 57:705, 57:710, 57:715, 08:939, 08:948
`\clist_map_variable:NNn` 51:70
`\clist_map_variable:nNn` 11:909
`\clist_new:N` 11:92, 51:53,
 51:54, 51:55, 51:56, 08:117, 08:134
`\clist_put_right:Nn` 11:784, 11:927,
 11:939, 51:33, 51:122, 51:129,
 51:136, 51:161, 51:177, 51:185, 51:201
`\clist_remove_all:Nn`
 . 11:727, 51:35, 51:135, 51:162, 51:202
`\clist_set:Nn` 11:680
`\clist_set_eq:NN` 51:171
`\clist_use:Nn` 08:1966, 08:2067
`\l_tmpa_clist` 373
`\clubpenalty`
 20:8, 20:25, 20:95, 20:152, 24:763,
 39:128, 39:225, 39:227, 44:100,
 44:106, 44:130, 44:135, 02:195, 450
`\clubsuit` 30:342
`cmd` (hook)
 315, 330, 331, 230, 236, 236, 269, 274
cmd commands:
`\cmd_arg_spec:N`
 07:2803, 07:2803, 07:2821
cmd internal commands:
`\__cmd_add_arg:n` 07:1757,
 07:1826, 07:1858, 07:1863, 07:1864,
 07:1898, 07:1986, 07:1993, 07:2068,
 07:2081, 07:2082, 07:2156, 07:2229,
 07:2236, 07:2280, 07:2280, 07:2285, 173
`\__cmd_add_arg_spec:n` ... 07:590,
 07:604, 07:672, 07:747, 07:747, 07:793
`\__cmd_add_arg_spec_mandatory:n` .
 07:636, 07:656, 07:747, 07:776
`\__cmd_add_arg_spec_mandatory:nn`
 07:650, 07:747, 07:777, 07:778

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`\__cmd_add_default:` 07:875,
 07:913, 07:930, 07:992, 07:995,
 07:1002, 07:1012, 07:1079, 07:1088, 153
`\__cmd_add_default:n`
 07:882, 07:922, 07:938,
 07:992, 07:992, 07:1009, 07:1023, 153
`\__cmd_add_default_E:nn`
 07:890, 07:992, 07:1007, 07:1057
`\__cmd_add_expandable_grabber:nn`
 07:1036, 07:1049, 07:1060,
 07:1080, 07:1090, 07:1097, 07:1097
`\__cmd_add_expandable_type_+:w` 07:1014
`\__cmd_add_expandable_type_D:w` 07:1019, 07:1019
`\__cmd_add_expandable_type_D_-`
`aux:NN` . . . 07:1019, 07:1025, 07:1042
`\__cmd_add_expandable_type_D_-`
`aux:NNN` . . 07:1019, 07:1026, 07:1029
`\__cmd_add_expandable_type_D_-`
`aux:NNNn` 07:1019, 07:1020, 07:1021, 07:1085
`\__cmd_add_expandable_type_E:w` 07:1055, 07:1055
`\__cmd_add_expandable_type_E_-`
`aux:n` 07:1055, 07:1059, 07:1071
`\__cmd_add_expandable_type_m:w` 07:1077, 07:1077
`\__cmd_add_expandable_type_R:w` 07:1084, 07:1084
`\__cmd_add_expandable_type_t:w` 07:1086, 07:1086
`\__cmd_add_grabber:N` 07:876,
 07:883, 07:896, 07:915, 07:923,
 07:931, 07:939, 07:953, 07:953, 155
`\__cmd_add_type_!:w` 07:843
`\__cmd_add_type_+:w` 07:836
`\__cmd_add_type_=:w` 07:858
`\__cmd_add_type_>:w` 07:850
`\__cmd_add_type_b:w` . . . 07:868, 07:868
`\__cmd_add_type_b_or_c:N`
 07:868, 07:869, 07:871, 07:872
`\__cmd_add_type_c:w` . . . 07:868, 07:870
`\__cmd_add_type_D:w` . . . 07:879, 07:879
`\__cmd_add_type_E:w` . . . 07:887, 07:887
`\__cmd_add_type_m:w` . . . 07:911, 07:911
`\__cmd_add_type_R:w` . . . 07:919, 07:919
`\__cmd_add_type_t:w` . . . 07:927, 07:927
`\__cmd_add_type_v:w` . . . 07:935, 07:935
`\__cmd_all_m_check:n`
 07:100, 07:100, 07:120
`\__cmd_all_m_check_aux:n`
 07:100, 07:101, 07:102
`\__cmd_allowed_token_check:N`
 07:588,
 07:600, 07:621, 07:646, 07:689, 07:689
`\__cmd_arg_spec_opt:N`
 07:2803, 07:2812, 07:2822
`\l__cmd_arg_spec_tl`
 07:12, 07:94, 07:451,
 07:535, 07:568, 07:622, 07:764, 147
`\__cmd_arg_to_keyvalue:nn`
 07:865, 07:2548, 07:2548
`\__cmd_arg_to_keyvalue_auxi:nnn`
 07:2548, 07:2562, 07:2565
`\__cmd_arg_to_keyvalue_auxii:Nnnn`
 07:2548, 07:2568, 07:2571
`\__cmd_arg_to_keyvalue_auxiii:nnn`
 07:2548, 07:2574, 07:2577
`\__cmd_arg_to_keyvalue_auxiv:Nnnn`
 07:2548, 07:2580, 07:2583
`\__cmd_arg_to_keyvalue_auxv:nn`
 07:2548, 07:2569,
 07:2575, 07:2581, 07:2587, 07:2589
`\__cmd_arg_to_keyvalue_braces:nnn`
 07:2548, 07:2550, 07:2553
`\__cmd_arg_to_keyvalue_loop:w`
 07:2548, 07:2591, 07:2594,
 07:2606, 07:2608, 07:2623, 07:2644
`\__cmd_arg_to_keyvalue_loop_-`
`group:n` . . 07:2548, 07:2600, 07:2605
`\__cmd_arg_to_keyvalue_loop_N_-`
`type:N` . . . 07:2548, 07:2597, 07:2609
`\__cmd_arg_to_keyvalue_loop_-`
`space:w` . . 07:2548, 07:2601, 07:2607
`\__cmd_arg_to_keyvalue_math:w`
 07:2548, 07:2622,
 07:2626, 07:2645, 07:2648, 07:2650
`\__cmd_arg_to_keyvalue_math_-`
`group:n` . . 07:2548, 07:2632, 07:2647
`\__cmd_arg_to_keyvalue_math_N_-`
`type:N` . . . 07:2548, 07:2629, 07:2637
`\__cmd_arg_to_keyvalue_math_-`
`space:w` . . 07:2548, 07:2633, 07:2649
`\__cmd_arg_to_keyvalue_set_-`
`default:nn`
 07:2548, 07:2612, 07:2640, 07:2651, 199
`\__cmd_arg_to_keyvalue_set_-`
`keyvalue:nn`
 07:2548, 07:2616, 07:2653, 199
`\l__cmd_args_i_tl` . . . 07:14, 07:331,
 07:337, 07:342, 07:343, 07:345, 130
`\l__cmd_args_ii_tl`
 07:15, 07:341, 07:343,
 07:345, 07:382, 07:387, 07:394, 130
`\__cmd_args_process:`
 07:321, 07:380, 07:380, 154

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

```

\__cmd_args_process_aux:n .....
..... 07:380, 07:393, 07:397
\__cmd_args_process_loop:nn ....
..... 07:380, 07:386, 07:389
\l_cmd_args_tl ... 07:13, 07:284,
07:309, 07:326, 07:331, 07:337,
07:356, 07:384, 07:387, 07:400,
07:401, 07:1584, 07:1585, 07:1590,
07:1593, 07:1964, 07:1998, 07:2282, 132
\__cmd_bad_arg_spec:wn ... 07:507,
07:512, 07:521, 07:528, 07:574,
07:587, 07:597, 07:598, 07:603,
07:614, 07:620, 07:641, 07:741, 07:741
\__cmd_bad_def:wn .....
... 07:465, 07:473, 07:502, 07:533,
07:564, 07:579, 07:669, 07:678,
07:686, 07:705, 07:714, 07:726,
07:741, 07:746, 07:757, 07:789, 162
\__cmd_bool_reverse:N .....
..... 07:2462, 07:2462, 07:3316
\__cmd_break_point:n .....
..... 07:97, 07:99, 07:99, 07:741,
07:746, 07:1147, 07:1154, 07:1365
\__cmd_cant_copy:nwn .....
..... 07:1144, 07:1154,
07:1154, 07:1270, 07:1333, 07:1363
\__cmd_check_definable:nNTF ....
..... 07:2663, 07:2663,
07:3105, 07:3118, 07:3131, 07:3136,
07:3224, 07:3237, 07:3250, 07:3258
\__cmd_check_definable_aux:nN ...
..... 07:2663, 07:2664, 07:2667, 202
\__cmd_check_end:n .....
..... 07:1329, 07:1331, 07:1337
\__cmd_check_end:Nn ... 07:1316,
07:1317, 07:1329, 07:1329, 07:1481
\__cmd_check_end:w .....
..... 07:1329, 07:1339, 07:1342
\__cmd_chk_if_free_cs:N .....
..... 07:3321, 07:3322
\__cmd_cmd_if_xparse:NTF ..... 1421
\__cmd_cmd_if_xparse_aux:N . 07:2730
\__cmd_cmd_type_cases:NnnnnnTF ..
..... 07:1138,
07:1357, 07:2730, 07:2730, 07:2753, 168
\__cmd_cmd_type_cases:NnnnnnTF . 1422
\__cmd_copy:NN .....
..... 07:1131, 07:1133, 07:1133,
07:1168, 07:1183, 07:1229, 07:1347, 163
\__cmd_copy_command:nnNN .....
..... 07:1139, 07:1169, 07:1169, 163
\__cmd_copy_command:NnNNnnnn ...
..... 07:1169, 07:1174, 07:1176, 163
\__cmd_copy_environment:nnNN ...
..... 07:1142, 07:1303, 07:1303
\__cmd_copy_environment:Nnnnnnn ...
..... 07:1303, 07:1309, 07:1312
\__cmd_copy_environment_end:nnNN
..... 07:1143, 07:1314, 07:1314
\__cmd_copy_environment_end_ -
aux:nnNN . 07:1314, 07:1318, 07:1321
\__cmd_copy_expandable:nnN .....
..... 07:1248, 07:1253, 07:1256,
07:1283, 07:1293, 07:1299, 07:1301
\__cmd_copy_expandable:nnNN .....
..... 07:1140, 07:1182, 07:1186,
07:1188, 07:1201, 07:1203, 07:1214
\__cmd_copy_expandable:NnNNNNnnnn
07:1182, 07:1198, 07:1211, 07:1231, 163
\__cmd_copy_expandable_signature:NnNNNNnnnn
07:1194, 07:1209, 07:1248, 07:1248, 163
\__cmd_copy_grabber_(type):w ... 165
\__cmd_copy_grabber_D:w .....
.. 07:1273, 07:1273, 07:1286, 07:1287
\__cmd_copy_grabber_D_alt:w ....
..... 07:1273, 07:1285, 07:1288
\__cmd_copy_grabber_E:w .....
..... 07:1273, 07:1289, 07:1295
\__cmd_copy_grabber_E_long:w ...
..... 07:1273, 07:1295
\__cmd_copy_grabber_m:w .....
..... 07:1273, 07:1301, 07:1302
\__cmd_copy_grabber_m_long:w ...
..... 07:1273, 07:1302
\__cmd_copy_grabber_R:w .....
..... 07:1273, 07:1287
\__cmd_copy_grabber_R_alt:w ....
..... 07:1273, 07:1288
\__cmd_copy_grabber_t:w .....
..... 07:1273, 07:1296
\__cmd_copy_optimized:nnNN .....
..... 07:1141, 07:1217, 07:1217
\__cmd_copy_parse_grabber:w ....
..... 07:1248, 07:1260, 07:1264, 165
\l_cmd_current_arg_int .....
.... 07:16, 07:28, 07:150, 07:158,
07:187, 07:194, 07:274, 07:355,
07:359, 07:363, 07:364, 07:448,
07:461, 07:484, 07:536, 07:582,
07:607, 07:792, 07:797, 07:798,
07:971, 07:997, 07:1004, 07:1251,
07:1259, 07:1277, 07:1281, 07:1282,
07:1292, 07:1490, 07:1556, 07:1565, 147
\__cmd_declare_cmd:Nnn .....
..... 07:59, 07:59,
07:3113, 07:3121, 07:3132, 07:3137

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

```

\__cmd_declare_cmd_aux:Nnn .....
..... 07:59, 07:62, 07:67, 07:69
\__cmd_declare_cmd_code:Nnn ....
..... 07:95, 07:110, 07:110
\__cmd_declare_cmd_code_aux:Nnn .
..... 07:110, 07:127, 07:154
\__cmd_declare_cmd_code_expandable:Nnn
..... 07:110, 07:126, 07:182
\__cmd_declare_cmd_internal:Nnnn
..... 07:59, 07:88, 07:90, 07:267, 138
\__cmd_declare_cmd_optimized:Nnn
..... 07:110, 07:129, 07:132
\__cmd_declare_env:nnnn .....
..... 07:228, 07:228,
07:230, 07:240, 07:242, 07:244,
07:3155, 07:3170, 07:3176, 07:3186,
07:3201, 07:3207, 07:3211, 07:3213, 212
\__cmd_declare_env_internal:nnnn
..... 07:228, 07:238, 07:260, 07:265
\__cmd_declare_expandable_-
cmd:Nnn ..... 07:59, 07:64,
07:3232, 07:3240, 07:3253, 07:3259
\__cmd_defaults: 07:320, 07:328, 07:328
\__cmd_defaults_aux: .....
07:328, 07:332, 07:333, 07:334, 07:339
\l__cmd_defaults_bool .....
..... 07:19, 07:173,
07:188, 07:211, 07:803, 07:998, 130
\__cmd_defaults_def: .....
..... 07:328, 07:330, 07:352
\__cmd_defaults_def:nn .....
..... 07:328, 07:357, 07:361
\__cmd_defaults_def:nnn .....
..... 07:328, 07:364, 07:366, 158
\__cmd_defaults_error:w .....
..... 07:328, 07:335, 07:347
\l__cmd_defaults_tl ..... 07:20,
07:174, 07:195, 07:312, 07:320,
07:356, 07:804, 07:999, 07:1005, 130
\__cmd_delimiter_check:nnn .....
..... 07:635, 07:648, 07:730, 07:730
\__cmd_end_expandable:NNw .....
..... 07:407, 07:409, 07:409
\__cmd_end_expandable_aux:nNNNN .
..... 07:409, 07:412, 07:413
\__cmd_end_expandable_aux:w ....
..... 07:409, 07:410, 07:411
\__cmd_end_expandable_defaults:nnnNNn
..... 07:409, 07:416,
07:425, 07:434, 07:444, 07:445, 142
\__cmd_end_expandable_defaults:nnw
..... 07:409, 07:433, 07:437
\__cmd_end_expandable_defaults:nw
..... 07:409, 07:440, 07:441, 07:443
\l__cmd_environment_bool .....
..... 07:21, 07:87, 07:114, 07:161,
07:237, 07:259, 07:292, 07:303,
07:322, 07:478, 07:665, 07:2837, 130
\__cmd_environment_or_command:..
... 07:350, 07:464, 07:496, 07:500,
07:578, 07:668, 07:676, 07:685,
07:700, 07:744, 07:783, 07:2066,
07:2226, 07:2233, 07:2835, 07:2835
\l__cmd_environment_str .....
..... 07:22, 07:164, 07:232,
07:246, 07:247, 07:248, 07:249,
07:252, 07:256, 07:261, 07:291,
07:294, 07:295, 07:323, 07:2838, 131
\l__cmd_expandable_aux_name_tl..
..... 07:24, 07:25,
07:1034, 07:1038, 07:1047, 07:1051, 131
\l__cmd_expandable_bool .....
07:23, 07:61, 07:66, 07:134, 07:146,
07:186, 07:198, 07:236, 07:258,
07:467, 07:476, 07:529, 07:560,
07:710, 07:720, 07:753, 07:812, 131
\__cmd_expandable_grab_D:nnNNwNN
..... 07:2286, 07:2306, 07:2312
\__cmd_expandable_grab_D:NNNwNNn
..... 07:2286, 07:2287, 07:2288
\__cmd_expandable_grab_D:NNNwNNnnn
..... 07:2286,
07:2299, 07:2304, 07:2330, 07:2419, 192
\__cmd_expandable_grab_D:Nw ....
..... 07:2286, 07:2308, 07:2311
\__cmd_expandable_grab_D:w .....
..... 07:2286, 07:2286
\__cmd_expandable_grab_D_-
alt:NNwn . 07:2347, 07:2352, 07:2442
\__cmd_expandable_grab_D_-
alt:NNwNNn 07:2334, 07:2335, 07:2336
\__cmd_expandable_grab_D_alt:Nwn
..... 07:2334
\__cmd_expandable_grab_D_alt:w ..
..... 07:2334, 07:2334
\__cmd_expandable_grab_E:w .....
..... 07:2365, 07:2365
\__cmd_expandable_grab_E_aux:w ..
..... 07:2365,
07:2366, 07:2368, 07:2369, 07:2397
\__cmd_expandable_grab_E_end:nnw
..... 07:2365, 07:2381, 07:2398
\__cmd_expandable_grab_E_-
find:nnw . 07:2365, 07:2395, 07:2396
\__cmd_expandable_grab_E_find:w .
..... 07:2365, 07:2386, 07:2394
\__cmd_expandable_grab_E_long:w .
..... 07:2365, 07:2367

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

```

\__cmd_expandable_grab_E_-
  loop:nnnNNw .....
  .. 07:2365, 07:2373, 07:2377, 07:2388
\__cmd_expandable_grab_E_-
  test:nnw . 07:2365, 07:2370, 07:2371
\__cmd_expandable_grab_m:w .....
  ..... 07:2400, 07:2400, 166
\__cmd_expandable_grab_m_aux:wNn
  .. 07:2400, 07:2401, 07:2403, 07:2404
\__cmd_expandable_grab_m_long:w .
  ..... 07:2400, 07:2402
\__cmd_expandable_grab_R:w .....
  ..... 07:2406, 07:2406
\__cmd_expandable_grab_R_alt:w ..
  ..... 07:2429, 07:2429
\__cmd_expandable_grab_R_alt_-
  aux:NNwNNn 07:2429, 07:2430, 07:2431
\__cmd_expandable_grab_R_-
  aux:NNNwNNn 07:2406, 07:2407, 07:2408
\__cmd_expandable_grab_t:w .....
  ..... 07:2452, 07:2452
\__cmd_expandable_grab_t_-
  aux:NNwn . 07:2452, 07:2453, 07:2454
\l_cmd_final_verb_bool .....
  ... 07:49, 07:809, 07:819, 07:968, 133
\__cmd_flush_m_args: .... 07:812,
  07:838, 07:845, 07:852, 07:860,
  07:874, 07:881, 07:889, 07:921,
  07:929, 07:937, 07:942, 07:942, 156
\l_cmd_fn_code_tl .....
  ..... 07:32, 07:311, 07:326, 131
\l_cmd_fn_tl .....
  ... 07:31, 07:123, 07:310, 07:1586,
  07:1852, 07:1876, 07:1882, 07:1892,
  07:1902, 07:1911, 07:1916, 07:1920,
  07:1951, 07:1977, 07:1985, 07:1987,
  07:1992, 07:1994, 07:2005, 07:2006,
  07:2011, 07:2012, 07:2017, 07:2018,
  07:2023, 07:2024, 07:2029, 07:2030,
  07:2035, 07:2036, 07:2041, 07:2042,
  07:2047, 07:2048, 07:2078, 07:2084, 134
\l_cmd_function_tl .....
  .... 07:27, 07:33, 07:92, 07:124,
  07:140, 07:143, 07:157, 07:168,
  07:169, 07:185, 07:193, 07:203,
  07:206, 07:210, 07:212, 07:220,
  07:226, 07:472, 07:532, 07:563,
  07:713, 07:725, 07:756, 07:2841, 131
\__cmd_get_grabber:NN .....
  07:1073, 07:1089, 07:1102, 07:1102, 160
\__cmd_get_grabber_auxi:NN .....
  .. 07:1102, 07:1105, 07:1108, 07:1116
\__cmd_get_grabber_auxii:NN .....
  ..... 07:1102, 07:1123, 07:1126
\__cmd_grab_b:w .... 07:1572, 07:1572
\__cmd_grab_b_aux:NNw .....
  ..... 07:1572, 07:1573,
  07:1575, 07:1577, 07:1579, 07:1580
\__cmd_grab_b_end:Nw .....
  ..... 07:1572, 07:1583, 07:1588
\__cmd_grab_b_long:w 07:1572, 07:1574
\__cmd_grab_b_long_obey_spaces:w
  ..... 07:1572, 07:1578
\__cmd_grab_b_obey_spaces:w ....
  ..... 07:1572, 07:1576
\__cmd_grab_c:w .... 07:1599, 07:1599
\__cmd_grab_c_auxi:w .....
  ..... 07:1638, 07:1643, 07:1645
\__cmd_grab_c_auxii:w .....
  ..... 07:1643, 07:1655, 07:1661
\__cmd_grab_c_auxiii:N .....
  .. 07:1643, 07:1667, 07:1669, 07:1678
\__cmd_grab_c_auxiv: .....
  .. 07:1643, 07:1674, 07:1692, 07:1715
\__cmd_grab_c_auxv: .... 07:1643,
  07:1688, 07:1701, 07:1720, 07:1742
\__cmd_grab_c_auxvi:N .....
  .. 07:1643, 07:1683, 07:1708, 07:1730
\__cmd_grab_c_auxvii: .. 07:1643,
  07:1714, 07:1719, 07:1734, 07:1741
\__cmd_grab_c_auxviii: .....
  ..... 07:1643, 07:1725, 07:1736
\__cmd_grab_c_end:n .....
  ..... 07:1745, 07:1761, 07:1767
\__cmd_grab_c_end:w .....
  ..... 07:1739, 07:1745, 07:1747
\__cmd_grab_c_end_auxi:w .....
  ..... 07:1745, 07:1769, 07:1772
\__cmd_grab_c_end_auxii:w .....
  ..... 07:1745, 07:1773, 07:1774
\__cmd_grab_c_end_auxiii:w .....
  ..... 07:1745, 07:1775, 07:1776
\__cmd_grab_c_first:w .....
  ..... 07:1599, 07:1619, 07:1623
\__cmd_grab_c_loop:w .....
  .... 07:1599, 07:1627, 07:1633,
  07:1636, 07:1653, 07:1699, 07:1704
\__cmd_grab_c_obey_spaces:w ....
  ..... 07:1599, 07:1604
\__cmd_grab_c_start:n .....
  .. 07:1599, 07:1602, 07:1607, 07:1609
\__cmd_grab_D:w ..... 07:1778
\__cmd_grab_D_aux:NNnNN . 07:1582,
  07:1809, 07:1813, 07:1849, 07:2061
\__cmd_grab_D_aux:NNnNNNN .....
  ..... 07:1790, 07:1809, 07:1811
\__cmd_grab_D_call:Nw .....
  07:1822, 07:1906, 07:1906, 07:2063, 178

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

_cmd_grab_D_long:w 07:1778
 _cmd_grab_D_long_no_strip:w 07:1778
 _cmd_grab_D_long_obey_spaces:w 07:1778
 _cmd_grab_D_long_obey_spaces_-no_strip:w 07:1778
 _cmd_grab_D_nested:NNnN 07:1855, 07:1869, 07:1872
 _cmd_grab_D_nested:w 07:1869, 07:1887, 07:1904
 _cmd_grab_D_no_strip:w ... 07:1778
 _cmd_grab_D_obey_spaces:w 07:1778
 _cmd_grab_D_obey_spaces_no-strip:w 07:1778
 _cmd_grab_D_verb_safe:NN 07:1809, 07:1817, 07:1830
 _cmd_grab_E:nnNN 07:1925, 07:1927, 07:1933, 07:1939, 07:1945, 07:1949
 _cmd_grab_E:w 07:1925, 07:1925
 _cmd_grab_E_finalize: 07:1925, 07:1958, 07:1974, 07:1981
 _cmd_grab_E_long:w 07:1925, 07:1931
 _cmd_grab_E_long_obey_spaces:w 07:1925, 07:1943
 _cmd_grab_E_loop:NnN .. 07:1925, 07:1954, 07:1969, 07:1971, 07:1978
 _cmd_grab_E_obey_spaces:w 07:1925, 07:1937
 \l_cmd_grab_expandably_bool ... 07:34, 07:125, 07:452, 07:477, 07:479, 07:537, 07:567, 07:592, 07:606, 07:628, 07:649, 07:716, 07:759, 07:832, 144
 _cmd_grab_m:w 07:1982, 07:1982
 _cmd_grab_m_1:w 07:1996
 _cmd_grab_m_2:w 07:1996
 _cmd_grab_m_3:w 07:1996
 _cmd_grab_m_4:w 07:1996
 _cmd_grab_m_5:w 07:1996
 _cmd_grab_m_6:w 07:1996
 _cmd_grab_m_7:w 07:1996
 _cmd_grab_m_8:w 07:1996
 _cmd_grab_m_9:w 07:1996
 _cmd_grab_m_aux:Nnnnnnnnn 07:1996, 07:1996, 07:2005, 07:2011, 07:2017, 07:2023, 07:2029, 07:2035, 07:2041, 07:2047
 _cmd_grab_m_long:w 07:1982, 07:1989
 _cmd_grab_R:w 07:2055, 07:2055
 _cmd_grab_R_aux:NNnN 07:2055, 07:2056, 07:2058, 07:2059
 _cmd_grab_R_long:w 07:2055, 07:2057
 _cmd_grab_t:w 07:2071, 07:2071
 _cmd_grab_t_aux:NNw 07:2071, 07:2072, 07:2074, 07:2075
 _cmd_grab_t_obey_spaces:w 07:2071, 07:2073
 _cmd_grab_v:w 07:2087, 07:2087
 _cmd_grab_v_aux:w 07:2087, 07:2090, 07:2095, 07:2097, 188
 _cmd_grab_v_aux_abort:n 07:2114, 07:2132, 07:2138, 07:2151, 07:2170, 07:2193, 07:2195, 07:2219, 187
 _cmd_grab_v_aux_catcodes: 07:2129, 07:2161, 07:2195, 07:2195, 07:2197, 07:2208, 07:2210, 186
 _cmd_grab_v_aux_loop:N 07:2124, 07:2130, 07:2134, 07:2148
 _cmd_grab_v_aux_loop:NN 07:2124, 07:2137, 07:2140
 _cmd_grab_v_aux_loop_end: 07:2124, 07:2145, 07:2153, 07:2184
 _cmd_grab_v_aux_put:N 07:2128, 07:2147, 07:2163, 07:2181, 07:2189, 07:2239, 07:2239, 07:2241, 07:2251, 07:2253, 07:2267, 07:2269
 _cmd_grab_v_aux_test:N 07:2113, 07:2124, 07:2124
 _cmd_grab_v_bgroup: 07:2109, 07:2159, 07:2159
 _cmd_grab_v_bgroup_loop: 07:2159, 07:2164, 07:2166, 07:2182, 07:2190
 _cmd_grab_v_bgroup_loop:N 07:2159, 07:2169, 07:2172
 _cmd_grab_v_group_end: 07:2087, 07:2118, 07:2155, 07:2221, 186
 _cmd_grab_v_long:w 07:2087, 07:2092
 _cmd_grab_v_token_if_char:NTF . 07:2126, 07:2142, 07:2174, 07:2278, 07:2278, 187
 \g_cmd_grabber_int 07:30, 07:1115, 07:1119, 131
 _cmd_if_novalue_aux:w . 07:3283, 215
 _cmd_if_recursion_tail_stop-do:Nn . 07:50, 07:52, 07:2611, 07:2639
 \c_cmd_ignore_def_tl .. 07:2833, 07:2849, 07:2856, 07:2870, 07:2903, 07:2932, 07:2939, 07:2946, 07:2954, 07:2962, 07:2971, 07:2978, 07:2985, 07:2992, 07:2999, 07:3011, 07:3018, 205
 \l_cmd_last_delimiters_tl 07:36, 07:450, 07:469, 07:591, 07:605, 07:627, 07:671, 07:722, 07:732, 07:791, 132
 \l_cmd_last_mandatory_arg_int .. 07:18, 07:449, 07:792, 07:972, 130

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx


```

\l__cmd_long_bool .. 07:37, 07:454,
    07:543, 07:544, 07:630, 07:750,
    07:761, 07:767, 07:773, 07:799,
    07:839, 07:960, 07:979, 07:1016,
    07:1031, 07:1044, 07:1061, 07:1081,
    07:2089, 07:2094, 07:2203, 07:2214, 132
\l__cmd_m_args_int .. 07:39, 07:802, 07:916,
    07:944, 07:947, 07:949, 07:951, 132
\l__cmd_nesting_a_tl .. 07:1869
\l__cmd_nesting_b_tl .. 07:1869
\__cmd_new_env:nnnn .. 07:3139, 07:3143,
    07:3157, 07:3188, 07:3214, 07:3215
\__cmd_normalize_arg_spec:n .. 07:93, 07:446, 07:446
\__cmd_normalize_arg_spec_loop:n .. 07:446, 07:459,
    07:481, 07:538, 07:583, 07:593,
    07:608, 07:631, 07:637, 07:651, 07:657
\__cmd_normalize_check_gv:N .. 07:655, 07:708, 07:708
\__cmd_normalize_check_lu:N .. 07:708, 07:718
\__cmd_normalize_E_unique_-
    check:w 07:585, 07:601, 07:610, 07:615
\__cmd_normalize_type_! :w ... 07:526
\__cmd_normalize_type_+ :w ... 07:526
\__cmd_normalize_type_= :w ... 07:526
\__cmd_normalize_type_> :w ... 07:526
\__cmd_normalize_type_aux:NnNn .. 07:526, 07:542, 07:548, 07:557, 07:572
\__cmd_normalize_type_b:w .. 07:659, 07:659
\__cmd_normalize_type_b_or_c:nn .. 07:659, 07:660, 07:662, 07:663
\__cmd_normalize_type_c:w .. 07:659, 07:661
\__cmd_normalize_type_D:w .. 07:508, 07:516, 07:518, 07:585, 07:585
\__cmd_normalize_type_d:w .. 07:505, 07:505
\__cmd_normalize_type_E:w .. 07:513, 07:585, 07:595
\__cmd_normalize_type_e:w .. 07:505, 07:510
\__cmd_normalize_type_m:w .. 07:633, 07:633
\__cmd_normalize_type_O:w .. 07:505, 07:517
\__cmd_normalize_type_o:w .. 07:505, 07:515
\__cmd_normalize_type_R:w .. 07:633, 07:639
\__cmd_normalize_type_r:w .. 07:505, 07:519
\__cmd_normalize_type_R_aux:w ... 07:522, 07:633, 07:642, 07:644
\__cmd_normalize_type_s:w .. 07:505, 07:524
\__cmd_normalize_type_t:w .. 07:525, 07:585, 07:618
\__cmd_normalize_type_v:w .. 07:633, 07:653
\l__cmd_obey_spaces_bool .. 07:35, 07:453, 07:549, 07:551,
    07:624, 07:629, 07:770, 07:774,
    07:785, 07:800, 07:846, 07:961,
    07:980, 07:1601, 07:1606, 07:1759, 131
\__cmd_peek_cs_check_equal:NNN .. 07:2767, 07:2781, 07:2787
\__cmd_peek_meaning:NTF .. 07:2758, 07:2767, 07:2767
\__cmd_peek_meaning_aux:NNTF .. 07:2767, 07:2768, 07:2770, 07:2771
\__cmd_peek_meaning_remove:NTF .. 07:1796, 07:1941, 07:1947,
    07:2074, 07:2760, 07:2767, 07:2769
\__cmd_peek_nonspace:NTF .. 07:2757, 07:2757
\__cmd_peek_nonspace_aux:nNNTF .. 07:2757,
    07:2758, 07:2760, 07:2761, 07:2764
\__cmd_peek_nonspace_remove:NTF .. 07:1797, 07:1835, 07:1929, 07:1935,
    07:2062, 07:2072, 07:2757, 07:2759
\__cmd_peek_true_remove:NNw 07:2767
\__cmd_peek_true_remove:Nw .. 07:2778, 07:2782, 07:2790, 07:2794
\l__cmd_prefixed_bool .. 07:40, 07:823, 07:840,
    07:847, 07:853, 07:861, 07:914, 132
\__cmd_prepare_signature:N .. 07:795, 07:811, 07:821,
    07:877, 07:885, 07:898, 07:917,
    07:925, 07:933, 07:940, 07:1017,
    07:1027, 07:1069, 07:1082, 07:1095, 153
\__cmd_prepare_signature:n .. 07:94, 07:795, 07:795
\__cmd_prepare_signature_-
    bypass:N 07:795, 07:824, 07:826,
    07:841, 07:848, 07:856, 07:866, 154
\__cmd_prepare_signature_verb_-
    chk:n .. 07:795, 07:810, 07:814
\l__cmd_process_all_tl .. 07:41, 07:178, 07:313, 07:321,
    07:385, 07:805, 07:948, 07:983, 132

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`\l__cmd_process_one_tl`
 ... 07:42, 07:806, 07:855, 07:864,
 07:894, 07:903, 07:987, 07:990, 132
`\l__cmd_process_some_bool` . 07:43,
 07:177, 07:807, 07:854, 07:863, 132
`\__cmd_provide_env:nnnn`
 07:3139, 07:3151,
 07:3182, 07:3190, 07:3218, 07:3219
`\__cmd_put_arg_expandable:nw` ...
 07:2318,
 07:2323, 07:2324, 07:2355, 07:2360,
 07:2361, 07:2460, 07:2460, 07:2461
`\__cmd_renew_env:nnnn`
 07:3139, 07:3147,
 07:3173, 07:3189, 07:3216, 07:3217
`\__cmd_replicate_processor:nn` ...
 07:893, 07:900, 07:900
`\__cmd_run_code:` .. 07:315, 07:318,
 07:318, 07:1580, 07:1588, 07:1596,
 07:1599, 07:1604, 07:1788, 07:1925,
 07:1931, 07:1937, 07:1943, 07:1967,
 07:1982, 07:1989, 07:2000, 07:2002,
 07:2008, 07:2014, 07:2020, 07:2026,
 07:2032, 07:2038, 07:2044, 07:2055,
 07:2057, 07:2075, 07:2097, 07:2283, 139
`\l__cmd_saved_args_tl`
 07:44, 07:1584, 07:1592, 132
`\__cmd_set_environment_end:n` ...
 07:228, 07:279, 07:323
`\__cmd_set_eq_if_exist:NN`
 07:1133, 07:1151, 07:1153,
 07:1172, 07:1191, 07:1192, 07:1193,
 07:1206, 07:1207, 07:1208, 07:1306
`\__cmd_show:N`
 07:1350, 07:1352, 07:1352, 07:1379,
 07:1402, 07:1485, 07:1487, 07:1569, 171
`\__cmd_show:n`
 .. 07:1369, 07:1464, 07:1472, 07:1476
`\__cmd_show_command:N`
 07:1358, 07:1369, 07:1369
`\__cmd_show_command:NnNNwN`
 07:1369, 07:1370, 07:1371
`\__cmd_show_command_aux:NnNNn` ...
 07:1369, 07:1373,
 07:1386, 07:1395, 07:1404, 07:1464, 168
`\__cmd_show_delim:Nw` 07:1517, 07:1517
`\__cmd_show_delims:Nw` 07:1517, 07:1519
`\__cmd_show_delims_opt:Nw`
 07:1517, 07:1521
`\__cmd_show_E:Nw` 07:1517, 07:1536
`\__cmd_show_e:Nw` 07:1517, 07:1525
`\__cmd_show_environment:N`
 .. 07:1361, 07:1369, 07:1450, 07:1482
`\__cmd_show_environment:Nnnw` ...
 07:1452, 07:1460
`\__cmd_show_environment_end:N` ...
 07:1362, 07:1479
`\__cmd_show_expandable:N`
 07:1359, 07:1369, 07:1376
`\__cmd_show_expandable:NnNNNNnN` .
 07:1369, 07:1377, 07:1382,
 07:1384, 07:1391, 07:1393, 07:1399
`\__cmd_show_opt:Nw` .. 07:1517, 07:1523
`\__cmd_show_optimized:N`
 07:1360, 07:1369, 07:1414
`\__cmd_show_optimized:NN`
 07:1416, 07:1420
`\__cmd_show_optimized_aux:N`
 07:1422, 07:1445
`\__cmd_show_prefix:Nw` 07:1517, 07:1549
`\__cmd_show_processor:Nw`
 07:1517, 07:1551
`\c__cmd_show_type_!_tl` 07:1511
`\c__cmd_show_type+_tl` 07:1511
`\c__cmd_show_type>_tl` 07:1511
`\c__cmd_show_type_D_tl` 07:1511
`\c__cmd_show_type_d_tl` 07:1511
`\c__cmd_show_type_E_tl` 07:1511
`\c__cmd_show_type_e_tl` 07:1511
`\c__cmd_show_type_0_tl` 07:1511
`\c__cmd_show_type_R_tl` 07:1511
`\c__cmd_show_type_r_tl` 07:1511
`\c__cmd_show_type_t_tl` 07:1511
`\l__cmd_signature_tl` 07:45, 07:171,
 07:214, 07:808, 07:884, 07:897,
 07:924, 07:932, 07:946, 07:955,
 07:1099, 07:1583, 07:1611, 07:1851,
 07:1957, 07:1967, 07:1984, 07:1991,
 07:2000, 07:2004, 07:2010, 07:2016,
 07:2022, 07:2028, 07:2034, 07:2040,
 07:2046, 07:2077, 07:2099, 07:2283, 173
`\__cmd_single_token_check:n`
 07:588, 07:589, 07:599,
 07:621, 07:646, 07:647, 07:680, 07:680
`\l__cmd_some_long_bool`
 .. 07:47, 07:118, 07:147, 07:208,
 07:216, 07:457, 07:751, 07:762, 133
`\l__cmd_some_obey_spaces_bool` ...
 .. 07:46, 07:456, 07:552, 07:780, 133
`\l__cmd_some_short_bool`
 07:48, 07:117,
 07:207, 07:218, 07:458, 07:763, 133
`\__cmd_split_add_item:n`
 07:1550, 07:1553, 07:1552, 07:1558, 173
`\__cmd_split_argument:nnn`
 07:2503, 07:2503, 07:3317

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`\__cmd_split_argument_aux:n` 07:2503, 07:2519, 07:2538
`\__cmd_split_argument_aux:nnnn` 07:2503, 07:2506, 07:2510
`\__cmd_split_argument_aux:wn` 07:2503, 07:2539, 07:2540
`\__cmd_split_end_item:n` . 07:1509, 07:1518, 07:1520, 07:1522, 07:1524, 07:1530, 07:1541, 07:1553, 07:1560, 173
`\__cmd_split_list:nn` 07:2468, 07:2470, 07:2505, 07:3318
`\__cmd_split_list_multi:nn` 07:2468, 07:2475, 07:2478, 07:2480, 07:2487, 07:2500
`\l__cmd_split_list_seq` 07:2468, 07:2482, 07:2484, 197
`\__cmd_split_list_single:Nn` 07:2468, 07:2476, 07:2490
`\l__cmd_split_list_tl` 07:2469, 07:2492, 07:2498, 07:2500, 197
`\__cmd_split_N_head_apply:Nn` 07:2548, 07:2568, 07:2580, 07:2655
`\__cmd_split_N_head_apply_-aux:NNw` . . 07:2548, 07:2656, 07:2657
`\__cmd_split_signature:n` 07:1406, 07:1488, 07:1488, 169
`\__cmd_split_signature_loop:Nw` . . 07:1493, 07:1495, 07:1495, 07:1509, 07:1518, 07:1520, 07:1522, 07:1524, 07:1532, 07:1545, 07:1550, 07:1552
`\__cmd_split_start_item:` 07:1498, 07:1529, 07:1540, 07:1553, 07:1553, 173
`\__cmd_start:nNNnnn` 07:167, 07:287, 07:298, 07:2738, 155
`\__cmd_start_aux:NNnnnn` 07:293, 07:304, 07:307, 07:307, 07:317
`\__cmd_start_env:nnnnn` 07:163, 07:287, 07:287, 07:2741, 203
`\__cmd_start_expandable:nNNNNn` . . 07:201, 07:404, 07:404, 07:2739, 1428
`\__cmd_start_optimized:` 07:110, 07:139, 07:153, 07:1225, 07:2740, 07:2810, 203
`\l__cmd_suppress_strip_bool` 07:38, 07:455, 07:558, 07:566, 07:801, 07:862, 07:963, 07:981, 132
`\__cmd_tl_mapthread_function:NNN` 07:356, 07:383, 07:2704, 07:2704
`\__cmd_tl_mapthread_function:nnN` 07:431, 07:1543, 07:2704, 07:2717, 172
`\__cmd_tl_mapthread_loop:w` 07:2704, 07:2707, 07:2719, 07:2723, 07:2728
`\__cmd_tmp:w` . 07:53, 07:56, 07:342, 07:358, 07:399, 07:401, 07:1104, 07:1110, 07:1128, 07:1335, 07:1345, 07:1511, 07:1514, 07:1516, 07:1534, 07:1538, 07:1544, 07:1548, 07:2799, 141
`\l__cmd_tmp_prop` 07:53, 07:1953, 07:1956, 07:1962, 133
`\l__cmd_tmpa_tl` 07:54, 07:354, 07:359, 07:368, 07:1073, 07:1075, 07:1089, 07:1092, 07:1246, 07:1252, 07:1267, 07:1275, 07:1291, 07:1298, 07:1316, 07:1319, 07:1410, 07:1481, 07:1482, 07:1491, 07:1562, 07:1665, 07:1677, 07:1687, 07:1696, 07:1703, 07:1735, 07:1890, 07:1893, 07:2773, 07:2798, 07:2801, 133
`\l__cmd_tmppb_tl` 07:55, 07:1058, 07:1063, 07:1074, 07:1317, 07:1319, 07:1492, 07:1498, 07:1555, 07:1559, 07:1563, 07:1564, 07:1682, 07:1724, 07:1729, 07:1735, 07:1738, 07:1962, 07:1963, 07:1965, 07:2774, 07:2785, 07:2791, 171
`\__cmd_token_if_cs:NTF` 07:1914, 07:2695, 07:2695, 07:2780, 149
`\l__cmd_total_args_int` 07:17, 07:797, 130
`\__cmd_trim_spaces:n` 07:2546, 07:2546, 07:3319
`\__cmd_use_i_delimit_by_q_-recursion_stop:nw` . 07:50, 07:2615
`\l__cmd_v_arg_tl` 07:1613, 07:1649, 07:1652, 07:1694, 07:1703, 07:1756, 07:1762, 07:2086, 07:2103, 07:2122, 07:2156, 07:2227, 07:2234, 07:2243, 07:2255, 07:2271, 187
`\l__cmd_v_nesting_int` . . 07:2158, 07:2162, 07:2178, 07:2179, 07:2188, 188
`\l__cmd_verb_safe_bool` 07:982
`cmd/#1/after (hook)` 342
`cmd/#1/before (hook)` 342
`cmd/<name>/after (hook)` . . . 327, 330, 241
`cmd/<name>/before (hook)` . . . 327, 327, 241
`cmd/foo/before (hook)` 329
`cmd/include/after (hook)` 1163
`cmd/include/before (hook)` 1163
`cmd/section/after (hook)` 330, 330
`cmd/section/before (hook)` 330
`\cmrFont` 33:1213, 33:1216
`\colon` 30:514
`\color` 449
`\columnbreak` 1070
`\columnsep` 20:27, 20:97, 20:154, 54:57, 54:207

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\columnseprule</code>	create commands:
..... 54:58, 54:2887, 54:2930, 54:2961	<code>create_callback</code> 04:781
<code>\columnwidth</code>	<code>\create_callback</code> 44
. 20:24, 20:27, 20:28, 20:30, 20:94,	<code>\CS</code> 82
20:97, 20:98, 20:100, 20:151, 20:154,	<code>\cs</code> 35:201
20:155, 20:158, 40:507, 40:546,	cs commands:
40:564, 40:581, 45:99, 45:168,	<code>\cs:w</code> 08:1543,
45:532, 45:551, 45:569, 54:56,	08:1544, 08:1605, 08:1625, 08:1628,
54:123, 54:124, 54:125, 54:206,	08:1689, 08:1714, 08:1717, 08:1735,
54:207, 54:208, 54:209, 54:210,	08:1736, 08:1760, 08:1761, 08:1762,
54:2420, 54:2422, 54:2885, 54:2889,	08:1769, 08:1776, 08:2254, 08:2273,
54:2928, 54:2932, 54:2957, 54:2963	08:2279, 08:2293, 08:2303, 08:2325,
<code>\cong</code> 30:468	08:2379, 08:2466, 08:2531, 08:2581,
<code>\contentsline</code>	08:2964, 08:2984, 08:2993, 08:2994,
. 44:164, 44:171, 44:177, 44:209, 1387	09:33, 28:335, 28:338, 51:210,
<code>\coprod</code> 30:352	52:211, 52:225, 52:237, 52:238,
<code>\copyright</code> .. 21:305, 21:336, 29:760, 1399	55:313, 07:1917, 07:2744, 08:375,
<code>\coremissesfalse</code> 33:1220	08:1088, 08:1145, 08:1296, 08:1436, 334
<code>\coremisstrue</code> 33:1198	<code>\cs_argument_spec:N</code> 05:155, 1432
<code>\cos</code> 38:12	<code>\cs_end:</code>
<code>\cosh</code> 38:14	08:1543, 08:1544, 08:1605, 08:1625,
<code>\cot</code> 38:18	08:1628, 08:1689, 08:1714, 08:1717,
<code>\coth</code> 38:19	08:1735, 08:1736, 08:1751, 08:1761,
<code>\countdef</code> . 04:75, 04:85, 04:174, 04:190,	08:1762, 08:1769, 08:1776, 08:2131,
04:198, 04:206, 04:224, 34:3, 02:37,	08:2253, 08:2257, 08:2261, 08:2273,
02:38, 02:39, 02:41, 02:47, 57:75, 01:50	08:2278, 08:2279, 08:2293, 08:2299,
counter 36:265, 854	08:2313, 08:2326, 08:2331, 08:2379,
<code>\counterwithin</code> 556	08:2465, 08:2466, 08:2531, 08:2581,
<code>\counterwithin</code> 22:203, 1422	08:2964, 08:2984, 08:2994, 09:33,
<code>\counterwithout</code> 556	28:335, 28:338, 51:210, 52:211,
<code>\counterwithout</code> 22:256, 1437	52:225, 52:237, 52:238, 55:313,
<code>\CountZero</code> 04:224	07:1917, 07:2746, 08:375, 08:789,
<code>\cr</code> 21:523, 21:529,	08:1088, 08:1145, 08:1299, 08:1439
21:539, 21:545, 33:65, 33:69, 33:891,	<code>\cs_generate_from_arg_count:NNnn</code>
33:895, 38:188, 38:192, 38:440,	 10:86, 10:103, 11:553,
38:469, 38:517, 38:633, 41:244,	11:1140, 07:142, 07:156, 07:184,
41:267, 41:280, 41:287, 41:296,	07:192, 07:272, 07:358, 07:399, 1430
41:301, 41:307, 41:454, 42:152,	<code>\cs_generate_variant:Nn</code>
42:154, 42:159, 42:165, 02:321, 1324	 08:1655, 08:1738, 11:140,
<code>\crrcr</code> ... 21:347, 21:382, 21:383, 21:411,	11:153, 11:440, 11:623, 11:649,
21:415, 21:418, 21:499, 21:503,	11:889, 11:891, 11:900, 11:902,
21:507, 21:509, 21:512, 21:756,	11:937, 11:940, 11:1014, 11:1047,
21:784, 21:788, 21:791, 21:858,	11:1129, 20:547, 36:37, 36:52,
21:861, 21:919, 21:1281, 29:762,	36:62, 36:79, 36:96, 36:116, 36:117,
30:348, 30:349, 30:351, 30:470,	36:126, 36:139, 36:236, 36:253,
30:473, 30:477, 30:541, 30:542,	48:298, 51:51, 51:52, 52:461, 52:469,
30:543, 30:544, 30:545, 30:546,	07:11, 07:240, 07:317, 57:649,
30:548, 30:549, 30:550, 30:551,	57:658, 57:659, 07:1153, 07:2285,
30:552, 30:554, 33:70, 33:896,	07:2461, 07:2487, 07:2821, 07:3188,
38:168, 38:170, 38:171, 38:172,	07:3189, 07:3190, 08:37, 08:47,
38:188, 38:190, 38:191, 38:192,	08:48, 08:51, 08:60, 08:1102, 08:1125
38:210, 38:211, 41:184, 41:189,	<code>\cs_gset:Npe</code> 52:213, 08:1069
41:194, 42:152, 55:457, 02:411, 1328	<code>\cs_gset:Npn</code>
	08:1651, 08:2092, 08:2336, 08:2406,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

09:383, 26:355, 26:356, 28:286,
 28:340, 36:24, 52:242, 53:234,
 53:529, 07:3294, 08:1101, 08:1121
 \cs_gset_eq:NN . . 08:2242, 08:2343,
 08:2344, 08:2345, 08:2346, 08:2610,
 09:72, 09:573, 09:601, 11:951,
 24:414, 28:284, 28:294, 28:332,
 28:346, 05:150, 05:151, 05:152,
 05:154, 05:155, 05:156, 05:157,
 05:158, 05:159, 53:113, 53:212,
 53:227, 53:228, 53:233, 53:242,
 53:246, 53:382, 53:440, 53:527,
 53:528, 07:3321, 08:792, 08:1446,
 08:1472, 08:1487, 08:1488, 08:1491
 \cs_gset_nopar:Npe 08:44, 08:46
 \cs_gset_nopar:Npn 08:2992
 \cs_gset_protected:Npe 10:21,
 10:23, 48:353, 53:81, 08:20, 08:1183
 \cs_gset_protected:Npn
 08:2187, 08:2190, 08:2218, 08:2223,
 08:2241, 08:2400, 08:2430, 08:2826,
 08:2969, 09:247, 09:457, 09:579,
 11:555, 11:1133, 11:1150, 51:85,
 07:3280, 08:82, 08:85, 08:94, 08:127,
 08:154, 08:182, 08:185, 08:191,
 08:215, 08:220, 08:245, 08:304,
 08:322, 08:603, 08:606, 08:624,
 08:650, 08:699, 08:707, 08:1470, 08:1531
 \cs_if_eq:NNTF
 51:98, 51:100, 51:167, 07:1110, 07:1973
 \cs_if_exist:N 1155
 \cs_if_exist:NTF . 08:1845, 08:2512,
 08:2541, 08:2549, 09:89, 09:97,
 10:118, 11:130, 11:156, 11:169,
 11:551, 11:613, 11:642, 11:654,
 11:795, 11:820, 11:949, 11:972,
 11:990, 05:153, 36:67, 36:179, 51:66,
 51:83, 51:169, 52:93, 52:438, 52:442,
 07:71, 07:233, 07:249, 07:1113,
 07:1152, 07:1470, 07:2805, 07:3107,
 07:3120, 07:3132, 07:3159, 07:3165,
 07:3175, 07:3184, 07:3196, 07:3199,
 07:3206, 07:3211, 07:3226, 07:3239,
 07:3252, 08:1073, 08:1138, 1177
 \cs_if_exist_p:N . 07:78, 07:79, 08:271
 \cs_if_exist_use:NTF 11:483,
 11:1041, 26:351, 05:198, 53:374,
 53:375, 53:379, 53:380, 07:485,
 07:1269, 08:1305, 08:1349, 08:1374
 \cs_if_free:NTF 36:8, 51:107
 \cs_meaning:N 10:69
 \cs_new:Npe 08:565
 \cs_new:Npn
 08:1568, 08:1569, 08:1740, 08:1741,
 08:2088, 08:2095, 08:2251, 08:2276,
 08:2284, 08:2297, 08:2305, 08:2314,
 08:2323, 08:2358, 08:2539, 08:2547,
 08:2561, 08:2579, 08:2588, 08:2593,
 08:2834, 08:2835, 08:2836, 08:2837,
 08:2841, 08:2842, 09:319, 09:329,
 09:339, 09:341, 09:343, 09:354,
 09:375, 09:394, 09:395, 09:398,
 09:519, 09:627, 10:88, 10:164, 11:11,
 11:21, 11:40, 11:46, 11:100, 11:441,
 11:442, 11:655, 11:664, 11:1061,
 11:1236, 11:1610, 11:1612, 11:1614,
 11:1617, 16:85, 16:115, 16:127,
 17:35, 20:538, 05:196, 36:63, 36:65,
 36:89, 36:97, 36:104, 36:118, 36:127,
 36:189, 36:190, 36:204, 36:205,
 36:219, 36:220, 48:279, 48:289,
 48:290, 48:291, 48:292, 48:297,
 49:111, 49:112, 51:205, 51:207,
 52:30, 52:38, 52:44, 52:57, 52:95,
 52:100, 52:105, 52:112, 52:127,
 52:234, 52:240, 52:411, 52:413,
 52:415, 52:418, 52:422, 52:436,
 52:452, 52:462, 52:582, 53:6, 53:7,
 53:8, 53:23, 53:48, 53:115, 53:120,
 53:138, 53:202, 53:207, 53:224,
 53:239, 53:244, 53:250, 53:253,
 53:267, 53:318, 53:371, 53:384,
 53:387, 53:397, 53:445, 53:526,
 53:585, 53:586, 53:587, 07:100,
 07:102, 07:153, 55:28, 55:42, 55:311,
 55:350, 55:356, 55:406, 55:409,
 07:404, 55:452, 07:409, 07:411,
 07:413, 07:425, 07:437, 07:443,
 07:1176, 07:1231, 07:1312, 07:1337,
 07:1342, 07:1767, 07:1772, 07:1774,
 07:1776, 07:1904, 07:2286, 07:2288,
 07:2304, 07:2311, 07:2312, 07:2334,
 07:2336, 07:2352, 07:2365, 07:2367,
 07:2369, 07:2371, 07:2377, 07:2394,
 07:2396, 07:2398, 07:2400, 07:2402,
 07:2404, 07:2406, 07:2408, 07:2429,
 07:2431, 07:2452, 07:2454, 07:2460,
 07:2538, 07:2540, 07:2565, 07:2571,
 07:2577, 07:2583, 07:2655, 07:2657,
 07:2704, 07:2717, 07:2723, 07:2730,
 07:2751, 07:2803, 07:2822, 07:2835,
 07:3261, 07:3278, 07:3279, 07:3299,
 07:3300, 07:3301, 08:39, 08:40,
 08:324, 08:330, 08:345, 08:355,
 08:356, 08:358, 08:372, 08:378,
 08:573, 08:926, 08:928, 08:930,
 08:1224, 08:1236, 08:1241, 08:1249,
 08:1258, 08:1260, 08:1267, 08:1294,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

08:1301, 08:1303, 08:1418, 08:1434
`\cs_new_eq:NN` . . . 08:1815, 08:1816,
08:1817, 08:1818, 08:1819, 08:1820,
08:2313, 08:2855, 08:2856, 08:2857,
08:2858, 08:2928, 08:2930, 09:13,
09:14, 09:474, 10:7, 10:8, 10:160,
10:161, 10:162, 10:163, 10:167,
10:192, 10:193, 10:194, 10:195,
10:196, 10:197, 10:198, 10:199,
10:200, 10:201, 10:202, 10:203,
10:204, 10:205, 10:206, 10:207,
10:208, 11:890, 11:901, 16:137,
20:816, 20:817, 05:176, 05:177,
05:178, 05:179, 05:195, 05:231,
36:188, 36:203, 36:218, 48:340,
48:356, 48:357, 48:399, 48:401,
48:513, 48:515, 51:8, 52:252, 52:253,
52:509, 52:511, 52:513, 52:515,
52:517, 52:519, 52:521, 52:523,
52:525, 52:527, 53:68, 53:213,
53:394, 53:396, 53:464, 53:467,
53:468, 53:538, 53:584, 07:56, 07:99,
55:3, 55:4, 55:345, 57:679, 57:680,
07:1287, 07:1288, 07:1295, 07:1302,
07:3101, 07:3102, 07:3296, 07:3297,
07:3298, 07:3304, 07:3305, 07:3306,
07:3316, 07:3317, 07:3318, 07:3319,
07:3320, 08:7, 08:23, 08:34, 08:57,
08:1208, 08:1391, 08:1398, 08:1441
`\cs_new_protected:Npe`
. 09:509, 11:366, 11:913,
48:114, 07:298, 07:1636, 07:1661,
07:1708, 07:1786, 07:2050, 07:2159
`\cs_new_protected:Npn`
08:1577, 08:1659, 08:1742, 08:1749,
08:1758, 08:1765, 08:1772, 08:1779,
08:1787, 08:1793, 08:1804, 08:1821,
08:1828, 08:1840, 08:1847, 08:1854,
08:1860, 08:1862, 08:1866, 08:1984,
08:2106, 08:2124, 08:2129, 08:2138,
08:2156, 08:2170, 08:2192, 08:2205,
08:2232, 08:2236, 08:2246, 08:2259,
08:2268, 08:2282, 08:2288, 08:2311,
08:2318, 08:2341, 08:2364, 08:2374,
08:2385, 08:2391, 08:2411, 08:2441,
08:2443, 08:2453, 08:2595, 08:2601,
08:2607, 08:2778, 08:2779, 08:2780,
08:2803, 08:2814, 08:2844, 08:2845,
08:2846, 08:2847, 08:2859, 08:2866,
08:2873, 08:2880, 08:2887, 08:2889,
08:2891, 08:2898, 08:2905, 08:2912,
08:2919, 09:18, 09:22, 09:24, 09:39,
09:41, 09:50, 09:52, 09:62, 09:70,
09:79, 09:93, 09:113, 09:133, 09:146,
09:151, 09:160, 09:165, 09:172,
09:312, 09:400, 09:438, 09:475,
09:503, 09:532, 10:9, 10:14, 10:19,
10:27, 10:58, 10:77, 10:78, 10:98,
10:122, 10:144, 10:154, 11:105,
11:111, 11:116, 11:117, 11:118,
11:128, 11:134, 11:141, 11:147,
11:154, 11:179, 11:187, 11:203,
11:211, 11:222, 11:232, 11:248,
11:258, 11:275, 11:281, 11:296,
11:314, 11:332, 11:344, 11:361,
11:387, 11:411, 11:443, 11:460,
11:462, 11:481, 11:492, 11:494,
11:496, 11:503, 11:511, 11:523,
11:549, 11:557, 11:576, 11:578,
11:592, 11:596, 11:611, 11:624,
11:638, 11:650, 11:652, 11:659,
11:678, 11:695, 11:716, 11:733,
11:740, 11:746, 11:751, 11:760,
11:762, 11:764, 11:773, 11:787,
11:810, 11:817, 11:831, 11:836,
11:852, 11:860, 11:866, 11:876,
11:884, 11:892, 11:906, 11:925,
11:930, 11:935, 11:938, 11:941,
11:954, 11:963, 11:984, 11:997,
11:1009, 11:1015, 11:1021, 11:1026,
11:1048, 11:1056, 11:1066, 11:1081,
11:1087, 11:1107, 11:1124, 11:1130,
11:1136, 11:1147, 11:1153, 11:1165,
11:1175, 11:1194, 11:1196, 11:1207,
11:1212, 11:1218, 11:1224, 11:1230,
11:1237, 11:1239, 11:1248, 11:1250,
11:1264, 11:1266, 11:1275, 11:1284,
11:1293, 11:1301, 11:1549, 11:1551,
11:1553, 11:1555, 11:1557, 11:1559,
11:1561, 11:1563, 11:1565, 11:1567,
11:1600, 11:1602, 11:1604, 11:1606,
11:1608, 11:1616, 11:1619, 11:1620,
11:1622, 11:1624, 11:1632, 11:1633,
16:88, 16:95, 24:413, 28:279, 36:6,
36:17, 36:22, 36:38, 36:43, 36:48,
36:50, 36:53, 36:150, 36:164,
36:171, 36:221, 36:228, 36:237,
48:7, 48:17, 48:54, 48:67, 48:75,
48:102, 48:132, 48:137, 48:162,
48:171, 48:213, 48:224, 48:235,
48:243, 48:246, 48:277, 48:341,
48:346, 48:351, 48:359, 48:372,
48:388, 48:417, 48:449, 51:22, 51:28,
51:44, 51:58, 51:60, 51:96, 51:105,
51:117, 51:133, 51:138, 51:147,
51:159, 51:164, 51:181, 51:197,
51:208, 51:212, 51:230, 51:244,
52:49, 52:62, 52:70, 52:79, 52:208,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

52:222, 52:392, 52:397, 52:420,
 52:547, 53:69, 53:74, 53:79, 53:390,
 53:547, 07:59, 07:64, 07:69, 07:90,
 07:110, 07:132, 07:154, 07:182,
 07:230, 07:244, 07:265, 07:279,
 07:287, 07:307, 07:318, 07:328,
 07:339, 07:347, 55:5, 55:6, 55:7,
 07:352, 55:20, 55:23, 55:29, 55:30,
 55:31, 55:32, 55:52, 55:53, 07:361,
 07:366, 07:380, 55:301, 55:302,
 07:389, 55:306, 55:315, 55:346,
 07:397, 55:377, 55:398, 55:402,
 55:412, 55:415, 55:418, 55:421,
 55:424, 55:441, 07:446, 57:629,
 57:660, 57:665, 57:670, 57:678,
 57:703, 57:708, 57:713, 07:481,
 07:505, 07:510, 07:515, 07:517,
 07:519, 07:524, 07:526, 07:540,
 07:546, 07:555, 07:572, 07:585,
 07:595, 07:610, 07:618, 07:633,
 07:639, 07:644, 07:653, 07:659,
 07:661, 07:663, 07:680, 07:689,
 07:708, 07:718, 07:730, 07:741,
 07:746, 07:747, 07:776, 07:778,
 07:795, 07:814, 07:821, 07:826,
 07:836, 07:843, 07:850, 07:858,
 07:868, 07:870, 07:872, 07:879,
 07:887, 07:900, 07:911, 07:919,
 07:927, 07:935, 07:942, 07:953,
 07:992, 07:1002, 07:1007, 07:1014,
 07:1019, 07:1021, 07:1029, 07:1042,
 07:1055, 07:1071, 07:1077, 07:1084,
 07:1086, 07:1097, 07:1102, 07:1108,
 07:1126, 07:1133, 07:1151, 07:1154,
 07:1169, 07:1188, 07:1203, 07:1217,
 07:1248, 07:1256, 07:1264, 07:1273,
 07:1285, 07:1289, 07:1296, 07:1301,
 07:1303, 07:1314, 07:1321, 07:1329,
 07:1352, 07:1369, 07:1371, 07:1376,
 07:1384, 07:1393, 07:1404, 07:1414,
 07:1420, 07:1450, 07:1460, 07:1472,
 07:1476, 07:1479, 07:1488, 07:1495,
 07:1517, 07:1519, 07:1521, 07:1523,
 07:1525, 07:1536, 07:1549, 07:1551,
 07:1553, 07:1558, 07:1560, 07:1572,
 07:1574, 07:1576, 07:1578, 07:1580,
 07:1588, 07:1599, 07:1604, 07:1609,
 07:1623, 07:1645, 07:1669, 07:1692,
 07:1701, 07:1734, 07:1736, 07:1747,
 07:1811, 07:1830, 07:1849, 07:1872,
 07:1925, 07:1931, 07:1937, 07:1943,
 07:1949, 07:1971, 07:1981, 07:1982,
 07:1989, 07:2002, 07:2008, 07:2014,
 07:2020, 07:2026, 07:2032, 07:2038,
 07:2044, 07:2055, 07:2057, 07:2059,
 07:2071, 07:2073, 07:2075, 07:2087,
 07:2092, 07:2097, 07:2118, 07:2124,
 07:2134, 07:2140, 07:2153, 07:2166,
 07:2172, 07:2197, 07:2210, 07:2219,
 07:2241, 07:2253, 07:2269, 07:2278,
 07:2280, 07:2462, 07:2470, 07:2480,
 07:2490, 07:2503, 07:2510, 07:2546,
 07:2548, 07:2553, 07:2589, 07:2594,
 07:2605, 07:2607, 07:2609, 07:2626,
 07:2637, 07:2647, 07:2649, 07:2651,
 07:2653, 07:2663, 07:2667, 07:2695,
 07:2757, 07:2759, 07:2761, 07:2767,
 07:2769, 07:2771, 07:2787, 07:2794,
 07:3103, 07:3116, 07:3129, 07:3134,
 07:3141, 07:3145, 07:3149, 07:3153,
 07:3157, 07:3173, 07:3182, 07:3194,
 07:3204, 07:3210, 07:3212, 07:3222,
 07:3235, 07:3248, 07:3256, 08:8,
 08:13, 08:18, 08:41, 08:43, 08:45,
 08:49, 08:52, 08:58, 08:63, 08:65,
 08:67, 08:98, 08:142, 08:166, 08:168,
 08:170, 08:195, 08:197, 08:199,
 08:225, 08:260, 08:262, 08:279,
 08:284, 08:286, 08:379, 08:388,
 08:393, 08:401, 08:425, 08:433,
 08:449, 08:457, 08:467, 08:482,
 08:493, 08:499, 08:505, 08:521,
 08:534, 08:577, 08:594, 08:654,
 08:673, 08:681, 08:688, 08:720,
 08:726, 08:732, 08:742, 08:784,
 08:786, 08:794, 08:796, 08:799,
 08:801, 08:982, 08:984, 08:1019,
 08:1056, 08:1068, 08:1071, 08:1100,
 08:1113, 08:1115, 08:1117, 08:1119,
 08:1136, 08:1160, 08:1167, 08:1325,
 08:1332, 08:1363, 08:1385, 08:1392,
 08:1399, 08:1405, 08:1410, 08:1415,
 08:1416, 08:1444, 08:1497, 1209
 \cs_new_protected_nopar:Npn
 57:675, 07:1906, 07:1996
 \cs_parameter_spec:N 05:153,
 05:154, 07:1442, 07:2824, 08:1252, 345
 \cs_prefix_spec:N 09:223, 09:298, 05:152
 \cs_replacement_spec 277
 \cs_replacement_spec:N
 08:1908, 08:1921,
 08:1930, 05:157, 07:1411, 07:1447,
 07:1456, 08:1227, 08:1462, 08:1484
 \cs_set:Npe 52:211, 52:225,
 07:223, 07:1032, 07:1045, 07:1422
 \cs_set:Npn
 08:1905, 08:2599, 08:2605,
 08:2978, 09:215, 09:290, 09:490,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

09:549, 09:585, 10:105, 11:419,
11:879, 16:37, 16:68, 26:344, 26:350,
26:354, 48:374, 53:15, 07:194,
07:273, 07:358, 07:399, 07:1104,
08:1169, 08:1180, 08:1187, 08:1196, 173
\cs_set_eq:NN . . . 08:1549, 08:1551,
08:1553, 08:1555, 08:1842, 08:1849,
08:1856, 09:187, 09:188, 09:208,
09:209, 09:214, 09:262, 09:263,
09:283, 09:284, 09:289, 09:486,
09:487, 09:545, 09:546, 09:581,
09:582, 09:640, 11:992, 16:93, 16:94,
16:138, 16:139, 16:140, 16:142,
16:143, 16:144, 16:178, 16:179,
16:180, 16:183, 28:299, 28:349,
36:254, 48:113, 48:280, 48:281,
48:282, 51:88, 53:85, 53:105, 53:144,
53:156, 53:186, 53:193, 53:470,
53:472, 53:474, 53:476, 53:478,
07:275, 07:276, 55:340, 55:343,
07:1128, 07:1152, 07:1171, 07:1190,
07:1205, 07:1219, 07:1280, 07:1305,
07:1310, 07:1325, 07:1327, 07:1616,
07:1832, 07:2005, 07:2011, 07:2017,
07:2023, 07:2029, 07:2035, 07:2041,
07:2047, 07:2199, 07:2212, 07:2799,
08:1516, 08:1517, 08:1518, 08:1519
\cs_set_nopar:Npe
. . . 07:135, 07:199, 07:220, 07:225,
07:270, 07:281, 07:1033, 07:1046,
07:1196, 07:1222, 07:1323, 08:42
\cs_set_nopar:Npn . 57:631, 57:632, 173
\cs_set_nopar:Npx 07:1210
\cs_set_protected:Npe 11:974
\cs_set_protected:Npn
. 08:2110, 08:2141, 09:189,
09:264, 11:107, 11:108, 11:113,
11:114, 11:415, 11:1134, 11:1151,
28:327, 28:359, 51:5, 51:218, 53:18,
53:19, 53:107, 53:196, 53:465,
07:158, 57:647, 07:1335, 07:1511,
07:1534, 07:1538, 07:1575, 07:1579,
07:1793, 07:1934, 07:1946, 07:1958,
07:1992, 07:2058, 07:2078, 08:414
\cs_set_protected_nopar:Npe
. 07:136, 07:159, 07:199,
07:1173, 07:1196, 07:1221, 07:1308
\cs_set_protected_nopar:Npn
. 07:1573, 07:1577, 07:1794,
07:1928, 07:1940, 07:1985, 07:2056
\cs_show:N 11:1265, 412
\cs_to_str:N 09:128, 09:149,
09:163, 09:168, 05:150, 07:78,
07:79, 07:92, 07:1145, 07:1147,
07:1365, 07:1417, 07:1455, 07:1457,
07:1481, 07:1917, 07:2746, 07:2813, 131
\cs_undefine:N . . 08:2351, 08:2437,
08:2575, 09:75, 09:226, 09:528,
09:536, 52:227, 07:3214, 07:3215,
07:3216, 07:3217, 07:3218, 07:3219,
07:3281, 07:3310, 07:3311, 07:3312,
07:3322, 08:73, 08:74, 08:84, 08:126,
08:184, 08:190, 08:265, 08:671,
08:1106, 08:1107, 08:1108, 08:1109,
08:1129, 08:1130, 08:1131, 08:1132,
08:1156, 08:1212, 08:1232, 08:1290,
08:1311, 08:1312, 08:1323, 08:1417
cs\check@icr commands:
 \cs_gset_eq:NN 76
\csc 38:21
\csname 1173
\csname\endcsname 1173
\csstring 04:212, 1440
\cup 30:384
\CurrentFile 50:981, 52:3,
52:67, 52:84, 52:166, 52:171, 52:174,
52:377, 52:381, 52:555, 52:556, 1170
\CurrentFilePath 52:3, 52:67,
52:83, 52:165, 52:378, 52:381, 1161
\CurrentFilePathUsed 52:3, 52:66,
52:81, 52:163, 52:375, 52:378, 1161
\CurrentFileUsed
50:981, 52:3, 52:66, 52:82, 52:164,
52:375, 52:377, 52:554, 52:555, 1161
\currentgrouplevel 50:823
\currentgrouptype 39:148,
39:151, 39:153, 41:247, 41:270, 1439
\CurrentOption
. 21:1564, 21:1567, 21:1572,
21:1588, 50:14, 50:555, 50:566,
50:579, 50:580, 50:585, 50:598,
50:599, 50:601, 50:615, 50:616,
50:619, 50:633, 50:638, 50:639,
50:652, 50:657, 50:658, 50:671,
50:673, 50:679, 50:681, 50:693,
50:694, 50:695, 50:703, 50:704,
50:705, 50:1031, 50:1096, 50:1215,
50:1216, 50:1226, 51:70, 51:71, 1363
CurrentOption commands:
 \CurrentOption: 50:578,
50:597, 50:614, 50:632, 50:637,
50:651, 50:656, 50:692, 50:702, 50:1225
\currsubencoding 33:1212,
33:1223, 33:1379, 33:1381, 33:1382
\CYRA 21:1536
\cyra 21:1536, 21:1591
\CYRABHCH 21:1536
\cyrabhch 21:1536

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

\CYRABHCHDSC	21:1536	\CYRGUP	21:1543
\cyrabhchdsc	21:1536	\cyrgup	21:1543
\CYRABHDZE	21:1537	\CYRHH	21:1543
\cyrabhdze	21:1536	\cyrh	21:1543
\CYRABHHA	21:1537	\CYRHDSC	21:1544
\cyrabhha	21:1537	\cyrhds	21:1544
\CYRAE	21:1537	\CYRHHCRS	21:1544
\cyrae	21:1537	\cyrhcrs	21:1544
\CYRB	21:1537	\CYRHHK	21:1544
\cyrb	21:1537	\cyrhk	21:1544
\CYRBYUS	21:1538	\CYRHRDSN	21:1545
\cyrbyus	21:1537	\cyhrdsn	21:1544
\CYRC	21:1538	\CYRI	21:1545
\cyrc	21:1538	\cyri	21:1545
\CYRCH	21:1538	\CYRIE	21:1545
\cyrch	21:1538	\cyrie	21:1545
\CYRCHLDSC	21:1538	\CYRII	21:1545
\cyrchldsc	21:1538	\cyrii	21:1545
\CYRCHRDSC	21:1539	\CYRISHRT	21:1545
\cyrchrdsc	21:1538	\cyrishrt	21:1545
\CYRCHVCRS	21:1539	\CYRISHRTDSC	21:1546
\cyrchvcrs	21:1539	\cyrishrtdsc	21:1546
\CYRD	21:1539	\CYRIZH	21:1546
\cyrd	21:1539	\cyrizh	21:1546
\CYRDELTA	21:1539	\CYRJE	21:1546
\cyrdelta	21:1539	\cyrje	21:1546
\CYRDJE	21:1540	\CYRK	21:1546
\cyrdje	21:1540	\cyrk	21:1546
\CYRDZE	21:1540	\CYRKBEAK	21:1547
\cyrdze	21:1540	\cyrkbeak	21:1547
\CYRDZHE	21:1540	\CYRKDSC	21:1547
\cyrdzhe	21:1540	\cyrkds	21:1547
\CYRE	21:1540	\CYRKHCRS	21:1547
\cyre	21:1540	\cyrkcrs	21:1547
\CYREPS	21:1541	\CYRKHK	21:1548
\cyreps	21:1540	\cyrkhk	21:1547
\CYREREV	21:1541	\CYRKVCRS	21:1548
\cyrerev	21:1541	\cyrkvcrs	21:1548
\CYRERY	21:1541	\CYRL	21:1548
\cyrery	21:1541	\cyr	21:1548
\CYRF	21:1541	\CYRLDSC	21:1548
\cyrf	21:1541	\cyrlds	21:1548
\CYRFITA	21:1542	\CYRLHK	21:1549
\cyrfita	21:1541	\cyrh	21:1548
\CYRG	21:1542	\CYRLJE	21:1549
\cyrg	21:1542	\cyrlje	21:1549
\CYRGDSC	21:1542	\CYRM	21:1549
\cyrgds	21:1542	\cyrm	21:1549
\CYRGDSCHCRS	21:1542	\CYRMDSC	21:1549
\cyrgdschcrs	21:1542	\cyrmds	21:1549
\CYRGHCRS	21:1543	\CYRMMHK	21:1549
\cyrghcrs	21:1543	\cyrmh	21:1549
\CYRGHK	21:1543	\CYRN	21:1550
\cyrghk	21:1543	\cyrn	21:1550

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\CYRNDSC	21:1550	\CYRTSHE	21:1556
\cyrndsc	21:1550	\cyrtshe	21:1556
\CYRNG	21:1550	\CYRU	21:1557
\cyrng	21:1550	\cyru	21:1557
\CYRNHK	21:1550	\CYRUSHRT	21:1557
\cyrnhk	21:1550	\cyrushrt	21:1557
\CYRNJE	21:1551	\CYRV	21:1557
\cyrnje	21:1550	\cyrv	21:1557
\CYRNLHK	21:1551	\CYRW	21:1557
\cyrnlhk	21:1551	\cyrw	21:1557
\CYRO	21:1551	\CYRY	21:1557
\cyro	21:1551	\cyry	21:1557
\CYROTLD	21:1551	\CYRYA	21:1558
\cyrotld	21:1551	\cyrya	21:1558
\CYRP	21:1551	\CYRYAT	21:1558
\cyrp	21:1551	\cyr yat	21:1558
\CYRPHK	21:1552	\CYRYHCRS	21:1558
\cyrphk	21:1552	\cyr yhcrs	21:1558
\CYRQ	21:1552	\CYRYI	21:1558
\cyrq	21:1552	\cyr yi	21:1558
\CYRR	21:1552	\CYRYO	21:1559
\cyrr	21:1552	\cyryo	21:1558
\CYRRDSC	21:1552	\CYRYU	21:1559
\cyrrdsc	21:1552	\cyr yu	21:1559
\CYRRHK	21:1553, 1402	\CYRZ	21:1559
\cyr rhk	21:1552, 1402	\cyrz	21:1559
\CYRRHOOK	1402	\CYRZDSC	21:1559
\cyr rhook	1402	\cyrz dsc	21:1559
\CYRRTICK	21:1553	\CYRZH	21:1559
\cyr rtick	21:1553	\cyr zh	21:1559
\CYRS	21:1553	\CYRZHDSC	21:1560
\cyrs	21:1553	\cyrzhdsc	21:1560
\CYRSACRS	21:1553		
\cyr sacrs	21:1553		
\CYRSCHWA	21:1554		
\cyr schwa	21:1554		
\CYRSDSC	21:1554		
\cyr sdsc	21:1554		
\CYRSEMISFTSN	21:1554		
\cyr semisftsn	21:1554		
\CYRSFTSN	21:1555		
\cyr sftsn	21:1555		
\CYRSH	21:1555		
\cyr sh	21:1555		
\CYRSHCH	21:1555		
\cyr shch	21:1555		
\CYRSHHA	21:1555		
\cyr shha	21:1555		
\CYRT	21:1556		
\cyrt	21:1556		
\CYRTDSC	21:1556		
\cyrt dsc	21:1556		
\CYRTETSE	21:1556		
\cyr tetse	21:1556		

D

\d	21:255, 21:416, 21:505, 21:789, 21:1277, 21:1491, 21:1492, 21:1493, 21:1494, 21:1497, 21:1498, 21:1499, 21:1500, 21:1501, 21:1502, 21:1503, 21:1504, 21:1505, 21:1506, 21:1507, 21:1508, 21:1509, 21:1510, 21:1511, 21:1512, 21:1513, 21:1514, 21:1515, 21:1516, 21:1517, 21:1518, 21:1519, 21:1520, 21:1521, 21:1522, 21:1523, 21:1524, 21:1525, 21:1526, 21:1527, 21:1528, 21:1529, 21:1530, 1377
\dag	21:332, 1382
\dagger	21:332, 22:332, 22:338, 22:346, 22:347, 30:386
\dashbox	42:320, 42:818, 42:835
\dashv	30:414
\date	1012
\date	44:9, 44:25
\day	03:11, 01:169, 50:1341, 50:1474, 50:1563, 02:259

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=lt para.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=lt page.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\dblfigrule</code>	54:1118, 54:3026, 1373	33:371, 33:372, 33:373, 33:374,
<code>\dblfloatpagefraction</code>	45:287, 45:301, 54:3003	33:375, 33:376, 33:377, 33:378,
<code>\dblfloatsep</code>	54:1104,	33:379, 33:380, 33:381, 33:382,
54:1116, 54:2217, 54:2343, 54:3010		33:383, 33:384, 33:385, 33:386,
<code>\dbltextfloatsep</code>	54:227, 54:235,	33:387, 33:388, 33:389, 33:390,
54:1120, 54:2216, 54:2342, 54:3010		33:391, 33:392, 33:393, 33:394,
<code>\dbltopfraction</code>	45:284, 45:298, 54:3002	33:395, 33:396, 33:397, 33:398,
<code>\ddag</code>	21:333, 1382	33:399, 33:400, 33:401, 33:402,
<code>\ddagger</code>	21:333,	33:403, 33:404, 33:405, 33:406,
22:333, 22:339, 22:346, 22:348, 30:385		33:407, 33:408, 33:409, 33:410,
<code>\ddot</code>	30:529	33:411, 33:412, 33:413, 33:414,
<code>\ddots</code>	30:524	33:415, 33:416, 33:417, 33:418,
<code>\deadcycles</code> 20:354, 20:410, 20:449, 37:35,		33:419, 33:420, 33:421, 33:422,
37:100, 37:176, 37:242, 54:304, 1398		33:423, 33:424, 33:425, 33:426,
debug commands:		33:427, 33:428, 33:429, 33:430,
<code>\debug_resume:</code>		33:431, 33:432, 33:433, 33:434,
.	11:185, 11:201, 11:209, 11:220, 247	33:435, 33:436, 33:437, 33:438,
<code>\debug_suspend:</code>		33:439, 33:440, 33:441, 33:442,
.	11:181, 11:189, 11:205, 11:213, 247	33:443, 33:444, 33:445, 33:446,
<code>\DebugHooksOff</code>	08:2846, 08:2954, 229	33:447, 33:448, 33:449, 33:450,
<code>\DebugHooksOn</code>	08:2846, 08:2953, 229	33:451, 33:452, 33:453, 33:454,
<code>\DebugMarksOff</code>	48:356, 1063	33:455, 33:456, 33:457, 33:458,
<code>\DebugMarksOn</code>	48:356, 1063	33:459, 33:460, 33:461, 33:462,
<code>\DebugShipoutsOff</code>	53:467, 53:499, 1189	33:463, 33:464, 33:465, 33:466,
<code>\DebugShipoutsOn</code>	53:467, 53:498, 1189	33:467, 33:468, 33:469, 33:470,
<code>\DebugSocketsOff</code>	10:192, 10:222, 361	33:471, 33:472, 33:473, 33:474,
<code>\DebugSocketsOn</code>	10:192, 10:221, 360	33:475, 33:476, 33:477, 33:478,
<code>\DebugTablesOff</code>	55:339	33:479, 33:480, 33:481, 33:482,
<code>\DebugTablesOn</code>	55:339	33:483, 33:484, 33:485, 33:486,
<code>\DebugTemplatesOff</code>	11:1632, 379	33:487, 33:488, 33:489, 33:490,
<code>\DebugTemplatesOn</code>	11:1632, 379	33:491, 33:492, 33:493, 33:494,
declare commands:		33:495, 33:496, 33:497, 33:498,
<code>declare_callback_rule</code>	04:894	33:499, 33:500, 33:501, 33:502,
<code>\Declare...</code>	117	33:503, 33:504, 33:505, 33:506,
<code>\declare_callback_rule</code>	44	33:507, 33:508, 33:509, 33:510,
<code>\DeclareAccent</code>	1366	33:511, 33:512, 33:513, 33:514,
<code>\DeclareCaseChangeEquivalent</code> 57:620, 1426		33:515, 33:516, 33:517, 33:518,
<code>\DeclareCommandCopy</code>		33:519, 33:520, 33:521, 33:522,
.	06:564, 06:565, 06:567, 98	33:523, 33:524, 33:525, 33:526,
<code>\DeclareCurrentRelease</code>	33:777, 50:1808	33:527, 33:528, 33:529, 33:530,
<code>\DeclareDefaultHookRule</code>		33:531, 33:532, 33:533, 33:534,
.	08:2850, 08:2957, 234	33:535, 33:536, 33:537, 33:538,
<code>\DeclareDocumentCommand</code>		33:539, 33:584, 33:782, 33:783,
.	44:186, 44:187, 07:3103, 90	33:784, 33:785, 33:826, 33:833,
<code>\DeclareDocumentEnvironment</code> 07:3139, 117		33:834, 33:835, 33:836, 33:1112,
<code>\DeclareEmphSequence</code>	755	33:1113, 33:1114, 33:1115, 33:1116,
<code>\DeclareEmphSequence</code>		33:1117, 33:1118, 33:1119, 33:1120,
.	29:624, 29:660, 29:661, 29:673, 756	33:1121, 33:1122, 33:1123, 33:1124,
<code>\DeclareEncoding</code>	1366	33:1125, 33:1126, 33:1127, 33:1128,
<code>\DeclareEncodingSubset</code>		33:1129, 33:1130, 33:1131, 33:1132,
.	24:186, 33:361, 33:362,	33:1133, 33:1134, 33:1135, 33:1136,
33:363, 33:364, 33:365, 33:366,		33:1137, 33:1138, 33:1139, 33:1140,
33:367, 33:368, 33:369, 33:370,		33:1141, 33:1142, 33:1143, 33:1144,
		33:1145, 33:1146, 33:1147, 33:1148,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

33:1149, 33:1150, 33:1151, 33:1152,	25:144, 25:145, 25:146, 25:147,
33:1153, 33:1154, 33:1155, 33:1156,	25:148, 25:149, 25:150, 25:151,
33:1157, 33:1158, 33:1159, 33:1160,	25:152, 25:153, 25:154, 25:155,
33:1161, 33:1162, 33:1163, 33:1164,	25:156, 25:157, 25:158, 25:159,
33:1165, 33:1166, 33:1167, 33:1168,	25:160, 25:161, 25:162, 25:163,
33:1169, 33:1170, 33:1171, 33:1172, 578	25:164, 25:165, 25:166, 25:167,
\DeclareEncodingsubset 830	25:168, 25:169, 25:170, 25:171,
\DeclareEnvironmentCopy	25:172, 25:173, 25:174, 25:175,
. 06:605, 06:606, 06:608, 1428	25:176, 25:177, 25:178, 25:179,
\DeclareErrorFont	25:180, 25:181, 25:182, 25:183,
. 24:574, 28:403, 28:453, 29:891, 30:61	25:184, 25:185, 25:186, 25:187,
\DeclareExpandableDocumentCommand . .	25:188, 25:189, 25:190, 25:191,
. 44:184, 44:185, 07:3222, 125	25:192, 25:193, 25:194, 25:195,
\DeclareFixedFont	25:196, 25:197, 25:198, 25:199,
. 24:76, 33:1213, 33:1215, 831	25:200, 25:201, 25:202, 25:203,
\DeclareFontEncoding	25:204, 25:205, 25:206, 25:207,
. . . 21:397, 21:485, 21:740, 21:762,	25:208, 25:209, 25:210, 25:211,
21:768, 21:854, 21:1066, 24:119,	25:212, 25:213, 25:214, 25:215,
30:137, 30:138, 30:139, 30:140, 1377	25:216, 25:217, 25:218, 25:219,
\DeclareFontEncodingDefaults	25:220, 25:221, 25:222, 25:223,
. 24:169, 27:90, 27:91, 30:36	25:224, 25:225, 25:226, 25:227,
\DeclareFontFamily 24:94, 27:85, 27:86, 590	25:228, 25:229, 25:230, 25:231,
\DeclareFontFamilySubstitution . 24:534	25:232, 25:233, 25:234, 25:235,
\DeclareFontSeriesChangeRule	25:236, 25:237, 25:238, 25:239,
. 25:4, 25:5, 25:7, 25:8, 25:9,	25:240, 25:241, 25:242, 25:243,
25:10, 25:11, 25:12, 25:13, 25:14,	25:244, 25:245, 25:246, 25:247,
25:15, 25:16, 25:17, 25:18, 25:19,	25:248, 25:249, 25:250, 25:251,
25:20, 25:21, 25:22, 25:23, 25:24,	25:252, 25:253, 25:254, 25:255,
25:25, 25:26, 25:27, 25:28, 25:29,	25:256, 25:257, 25:258, 25:259,
25:30, 25:31, 25:32, 25:33, 25:34,	25:260, 25:261, 25:262, 25:263,
25:35, 25:36, 25:37, 25:38, 25:39,	25:264, 25:265, 25:266, 25:267,
25:40, 25:41, 25:42, 25:43, 25:44,	25:268, 25:269, 25:270, 25:271,
25:45, 25:46, 25:47, 25:48, 25:49,	25:272, 25:273, 25:274, 25:275,
25:50, 25:51, 25:52, 25:53, 25:54,	25:276, 25:277, 25:278, 25:279,
25:55, 25:56, 25:57, 25:58, 25:59,	25:280, 25:281, 25:282, 25:283,
25:60, 25:61, 25:62, 25:63, 25:64,	25:284, 25:285, 25:286, 25:287,
25:65, 25:66, 25:67, 25:68, 25:69,	25:288, 25:289, 25:290, 25:291,
25:70, 25:71, 25:72, 25:73, 25:74,	25:292, 25:293, 25:294, 25:295,
25:75, 25:76, 25:77, 25:78, 25:79,	25:296, 25:297, 25:298, 25:299,
25:80, 25:81, 25:82, 25:83, 25:84,	25:300, 25:301, 25:302, 25:303,
25:85, 25:86, 25:87, 25:88, 25:89,	25:304, 25:305, 25:306, 25:307,
25:90, 25:91, 25:92, 25:93, 25:94,	25:308, 25:309, 25:310, 25:311,
25:95, 25:96, 25:97, 25:98, 25:99,	25:312, 25:313, 25:314, 25:315,
25:100, 25:101, 25:102, 25:103,	25:316, 25:317, 25:318, 25:319,
25:104, 25:105, 25:106, 25:107,	25:320, 25:321, 25:322, 25:323,
25:108, 25:109, 25:110, 25:111,	25:324, 25:325, 25:326, 25:327,
25:112, 25:113, 25:114, 25:115,	25:328, 25:329, 25:330, 25:331,
25:116, 25:117, 25:118, 25:119,	25:332, 25:333, 25:334, 25:335,
25:120, 25:121, 25:122, 25:123,	25:336, 25:337, 25:338, 25:339,
25:124, 25:125, 25:126, 25:127,	25:340, 25:341, 25:342, 25:343,
25:128, 25:129, 25:130, 25:131,	25:344, 25:345, 25:346, 25:347,
25:132, 25:133, 25:134, 25:135,	25:348, 25:349, 25:350, 25:351,
25:136, 25:137, 25:138, 25:139,	25:352, 25:353, 25:354, 25:355,
25:140, 25:141, 25:142, 25:143,	25:356, 25:357, 25:358, 25:359,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

25:360, 25:361, 25:362, 25:363, 25:576, 25:577, 25:578, 25:579,
 25:364, 25:365, 25:366, 25:367, 25:580, 25:581, 25:582, 25:583,
 25:368, 25:369, 25:370, 25:371, 25:584, 25:585, 25:586, 25:587,
 25:372, 25:373, 25:374, 25:375, 25:588, 25:589, 25:590, 25:591,
 25:376, 25:377, 25:378, 25:379, 25:592, 25:593, 25:594, 25:595,
 25:380, 25:381, 25:382, 25:383, 25:596, 25:597, 25:598, 25:599,
 25:384, 25:385, 25:386, 25:387, 25:600, 25:601, 25:602, 25:603,
 25:388, 25:389, 25:390, 25:391, 25:604, 25:605, 25:606, 25:607,
 25:392, 25:393, 25:394, 25:395, 25:608, 25:609, 25:610, 25:611,
 25:396, 25:397, 25:398, 25:399, 25:612, 25:613, 25:614, 25:615,
 25:400, 25:401, 25:402, 25:403, 25:616, 25:617, 25:618, 25:619,
 25:404, 25:405, 25:406, 25:407, 25:620, 25:621, 25:622, 25:623,
 25:408, 25:409, 25:410, 25:411, 25:624, 25:625, 25:626, 25:627,
 25:412, 25:413, 25:414, 25:415, 25:628, 25:629, 25:630, 25:631,
 25:416, 25:417, 25:418, 25:419, 25:632, 25:633, 25:634, 25:635,
 25:420, 25:421, 25:422, 25:423, 25:636, 25:637, 25:638, 25:639,
 25:424, 25:425, 25:426, 25:427, 25:640, 25:641, 25:642, 25:643,
 25:428, 25:429, 25:430, 25:431, 25:644, 25:645, 25:646, 25:647,
 25:432, 25:433, 25:434, 25:435, 25:648, 25:649, 25:650, 25:651,
 25:436, 25:437, 25:438, 25:439, 25:652, 25:653, 25:654, 25:655,
 25:440, 25:441, 25:442, 25:443, 25:656, 25:657, 25:658, 25:659,
 25:444, 25:445, 25:446, 25:447, 25:660, 25:661, 25:662, 25:663,
 25:448, 25:449, 25:450, 25:451, 25:664, 25:665, 25:666, 25:667,
 25:452, 25:453, 25:454, 25:455, 25:668, 25:669, 25:670, 25:671,
 25:456, 25:457, 25:458, 25:459, 25:672, 25:673, 25:674, 25:675,
 25:460, 25:461, 25:462, 25:463, 25:676, 25:677, 25:678, 25:679,
 25:464, 25:465, 25:466, 25:467, 25:680, 25:681, 25:682, 25:683,
 25:468, 25:469, 25:470, 25:471, 25:684, 25:685, 25:686, 25:687,
 25:472, 25:473, 25:474, 25:475, 25:688, 25:689, 25:690, 25:691,
 25:476, 25:477, 25:478, 25:479, 25:692, 25:693, 25:694, 25:695,
 25:480, 25:481, 25:482, 25:483, 25:696, 25:697, 25:698, 25:699,
 25:484, 25:485, 25:486, 25:487, 25:700, 25:701, 25:702, 25:703,
 25:488, 25:489, 25:490, 25:491, 25:704, 25:705, 25:706, 25:707,
 25:492, 25:493, 25:494, 25:495, 25:708, 25:709, 25:710, 25:711,
 25:496, 25:497, 25:498, 25:499, 25:712, 25:713, 25:714, 25:715,
 25:500, 25:501, 25:502, 25:503, 25:716, 25:717, 25:718, 25:719,
 25:504, 25:505, 25:506, 25:507, 25:720, 25:721, 25:722, 25:723,
 25:508, 25:509, 25:510, 25:511, 25:724, 25:725, 25:726, 25:727,
 25:512, 25:513, 25:514, 25:515, 25:728, 25:729, 25:730, 25:731,
 25:516, 25:517, 25:518, 25:519, 25:732, 25:733, 25:734, 25:735,
 25:520, 25:521, 25:522, 25:523, 25:736, 25:737, 25:738, 25:739,
 25:524, 25:525, 25:526, 25:527, 25:740, 25:741, 25:742, 25:743,
 25:528, 25:529, 25:530, 25:531, 25:744, 25:745, 25:746, 25:747,
 25:532, 25:533, 25:534, 25:535, 25:748, 25:749, 25:750, 25:751,
 25:536, 25:537, 25:538, 25:539, 25:752, 25:753, 25:754, 25:755,
 25:540, 25:541, 25:542, 25:543, 25:756, 25:757, 25:758, 25:759,
 25:544, 25:545, 25:546, 25:547, 25:760, 25:761, 25:762, 25:763,
 25:548, 25:549, 25:550, 25:551, 25:764, 25:765, 25:766, 25:767,
 25:552, 25:553, 25:554, 25:555, 25:768, 25:769, 25:770, 25:771,
 25:556, 25:557, 25:558, 25:559, 25:772, 25:773, 25:774, 25:775,
 25:560, 25:561, 25:562, 25:563, 25:776, 25:777, 25:778, 25:779,
 25:564, 25:565, 25:566, 25:567, 25:780, 25:781, 25:782, 25:783,
 25:568, 25:569, 25:570, 25:571, 25:784, 25:785, 25:786, 25:787,
 25:572, 25:573, 25:574, 25:575, 25:788, 25:789, 25:790, 25:791,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:792, 25:793, 25:794, 25:795,
 25:796, 25:797, 25:798, 25:799,
 25:800, 25:801, 25:802, 25:803,
 25:804, 25:805, 25:806, 25:807,
 25:808, 25:809, 25:810, 25:811,
 25:812, 25:813, 25:814, 25:815,
 25:816, 25:817, 25:818, 25:819,
 25:820, 25:821, 25:822, 25:823,
 25:824, 25:825, 25:826, 25:827,
 25:828, 25:829, 25:830, 25:831,
 25:832, 25:833, 25:834, 25:835,
 25:836, 25:837, 25:838, 25:839,
 25:840, 25:841, 25:842, 25:843,
 25:844, 25:845, 25:846, 25:847,
 25:848, 25:849, 25:850, 25:851,
 25:852, 25:853, 25:854, 25:855,
 25:856, 25:857, 25:858, 25:859,
 25:860, 25:861, 25:862, 25:863,
 25:864, 25:865, 25:866, 25:867,
 25:868, 25:869, 25:870, 25:871,
 25:872, 25:873, 25:874, 25:875,
 25:876, 25:877, 25:878, 25:879,
 25:880, 25:881, 25:882, 25:883,
 25:884, 25:885, 25:886, 25:887,
 25:888, 25:889, 25:890, 25:891,
 25:892, 25:893, 25:894, 25:895,
 25:896, 25:897, 25:898, 25:899,
 25:900, 25:901, 25:902, 25:903,
 25:904, 25:905, 25:906, 25:907,
 25:908, 25:909, 25:910, 25:911,
 25:912, 25:913, 25:914, 25:915,
 25:916, 25:917, 25:918, 25:919,
 25:920, 25:921, 25:922, 25:923,
 25:924, 25:925, 25:926, 25:927,
 25:928, 25:929, 25:930, 25:931,
 25:932, 25:933, 25:934, 25:935,
 25:936, 25:937, 25:938, 25:939,
 25:940, 25:941, 25:942, 25:943,
 25:944, 25:945, 25:946, 25:947,
 25:948, 25:949, 25:950, 25:951,
 25:952, 25:953, 25:954, 25:955,
 25:956, 25:957, 25:958, 25:959,
 25:960, 25:961, 25:962, 25:963,
 25:964, 25:965, 25:966, 25:967,
 25:968, 25:969, 25:970, 25:971,
 25:972, 25:973, 25:974, 25:975,
 25:976, 25:977, 25:978, 25:979,
 25:980, 25:981, 25:982, 25:983,
 25:984, 25:985, 25:986, 25:987,
 25:988, 25:989, 25:990, 25:991,
 25:992, 25:993, 25:994, 25:995,
 25:996, 25:997, 25:998, 25:999,
 25:1000, 25:1001, 25:1002, 25:1003,
 25:1004, 25:1005, 25:1006, 25:1007,
 25:1008, 25:1009, 25:1010, 25:1011,
 25:1012, 25:1013, 25:1014, 25:1015,
 25:1016, 25:1017, 25:1018, 25:1019,
 25:1020, 25:1021, 25:1022, 25:1023,
 25:1024, 25:1025, 25:1026, 25:1027,
 25:1028, 25:1029, 25:1030, 25:1031,
 25:1032, 25:1033, 25:1034, 25:1035,
 25:1036, 25:1037, 25:1038, 25:1039,
 25:1040, 25:1041, 25:1042, 25:1043,
 25:1044, 25:1045, 25:1046, 25:1047,
 25:1048, 25:1049, 25:1050, 25:1051,
 25:1052, 25:1053, 25:1054, 25:1055,
 25:1056, 25:1057, 25:1058, 25:1059,
 25:1060, 25:1061, 25:1062, 25:1063,
 25:1064, 25:1065, 25:1066, 25:1067,
 25:1068, 25:1069, 25:1070, 25:1071,
 25:1072, 25:1073, 25:1074, 25:1075,
 25:1076, 25:1077, 25:1078, 25:1079,
 25:1080, 25:1081, 25:1082, 25:1083,
 25:1084, 25:1085, 25:1086, 25:1087,
 25:1088, 25:1089, 25:1090, 25:1091,
 25:1092, 25:1093, 25:1094, 25:1095,
 25:1096, 25:1097, 25:1098, 25:1099,
 25:1100, 25:1101, 25:1102, 25:1103,
 25:1104, 25:1105, 25:1106, 25:1107,
 25:1108, 25:1109, 25:1110, 25:1111,
 25:1112, 25:1113, 25:1114, 25:1115,
 25:1116, 25:1117, 25:1118, 25:1119,
 25:1120, 25:1121, 25:1122, 25:1123,
 25:1124, 25:1125, 25:1126, 25:1127,
 25:1128, 25:1129, 25:1130, 25:1131,
 25:1132, 25:1133, 25:1134, 25:1135,
 25:1136, 25:1137, 25:1138, 25:1139,
 25:1140, 25:1141, 25:1142, 25:1143,
 25:1144, 25:1145, 25:1146, 25:1147,
 25:1148, 25:1149, 25:1150, 25:1151,
 25:1152, 25:1153, 25:1154, 25:1155,
 25:1156, 25:1157, 25:1158, 25:1159,
 25:1160, 25:1161, 25:1162, 25:1163,
 25:1164, 25:1165, 25:1166, 25:1167,
 25:1168, 25:1169, 25:1170, 25:1171,
 25:1172, 25:1173, 25:1174, 25:1175,
 25:1176, 25:1177, 25:1178, 25:1179,
 25:1180, 25:1181, 25:1182, 25:1183,
 25:1184, 25:1185, 25:1186, 25:1187,
 25:1188, 25:1189, 25:1190, 25:1191,
 25:1192, 25:1193, 25:1194, 25:1195,
 25:1196, 25:1197, 25:1198, 25:1199,
 25:1200, 25:1201, 25:1202, 25:1203,
 25:1204, 25:1205, 25:1206, 25:1207,
 25:1208, 25:1209, 25:1210, 25:1211,
 25:1212, 25:1213, 25:1214, 25:1215,
 25:1216, 25:1217, 25:1218, 25:1219,
 25:1220, 25:1221, 25:1222, 25:1223,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

25:1224, 25:1225, 25:1226, 25:1227,
 25:1228, 25:1229, 25:1230, 25:1231,
 25:1232, 25:1233, 25:1234, 25:1235,
 25:1236, 25:1237, 25:1238, 25:1239,
 25:1240, 25:1241, 25:1242, 25:1243,
 25:1244, 25:1245, 25:1246, 25:1247,
 25:1248, 25:1249, 25:1250, 25:1251,
 25:1252, 25:1253, 25:1254, 25:1255,
 25:1256, 25:1257, 25:1258, 25:1259,
 25:1260, 25:1261, 25:1262, 25:1263,
 25:1264, 25:1265, 25:1266, 25:1267,
 25:1268, 25:1269, 25:1270, 25:1271,
 25:1272, 25:1273, 25:1274, 25:1275,
 25:1276, 25:1277, 25:1278, 25:1279,
 25:1280, 25:1281, 25:1282, 25:1283,
 25:1284, 25:1285, 25:1286, 25:1287,
 25:1288, 25:1289, 25:1290, 25:1291,
 25:1292, 25:1293, 25:1294, 25:1295,
 25:1296, 25:1297, 25:1298, 25:1299,
 25:1300, 25:1301, 25:1302, 25:1303,
 25:1304, 25:1305, 25:1306, 25:1307,
 25:1308, 25:1309, 25:1310, 25:1311,
 25:1312, 25:1313, 25:1314, 25:1315,
 25:1316, 25:1317, 25:1318, 25:1319,
 25:1320, 25:1321, 25:1322, 25:1323,
 25:1324, 25:1325, 25:1326, 25:1327,
 25:1328, 25:1329, 25:1330, 25:1331,
 25:1332, 25:1333, 25:1334, 25:1335,
 25:1336, 25:1337, 25:1338, 25:1339,
 25:1340, 25:1341, 25:1342, 25:1343,
 25:1344, 25:1345, 25:1346, 25:1347,
 25:1348, 25:1349, 25:1350, 25:1351,
 25:1352, 25:1353, 25:1354, 25:1355,
 25:1356, 25:1357, 25:1358, 25:1359,
 25:1360, 25:1361, 25:1362, 25:1363,
 25:1364, 25:1365, 25:1366, 25:1367,
 25:1368, 25:1369, 25:1370, 25:1371,
 25:1372, 25:1373, 25:1374, 25:1375,
 25:1376, 25:1377, 25:1378, 25:1379,
 25:1380, 25:1381, 25:1382, 25:1383,
 25:1384, 25:1385, 25:1386, 25:1387,
 25:1388, 25:1389, 25:1390, 25:1391,
 25:1392, 25:1393, 25:1394, 25:1395,
 25:1396, 25:1397, 25:1398, 25:1399,
 25:1400, 25:1401, 25:1402, 25:1403,
 25:1404, 25:1405, 25:1406, 25:1407,
 25:1408, 25:1409, 25:1410, 25:1411,
 25:1412, 25:1413, 25:1414, 25:1415,
 25:1416, 25:1417, 25:1418, 25:1419,
 25:1420, 25:1421, 25:1422, 25:1423,
 25:1424, 25:1425, 25:1426, 25:1427,
 25:1428, 25:1429, 25:1433, 25:1435,
 25:1438, 25:1439, 25:1440, 25:1441,
 25:1442, 25:1443, 25:1444, 25:1445,
 25:1446, 25:1447, 25:1448, 25:1449,
 25:1450, 25:1451, 25:1452, 25:1453,
 25:1454, 25:1455, 25:1456, 25:1457,
 25:1458, 25:1459, 25:1460, 25:1461,
 25:1462, 25:1463, 25:1464, 25:1465,
 25:1466, 25:1467, 25:1468, 25:1469,
 25:1470, 25:1471, 25:1472, 25:1473,
 25:1474, 25:1475, 25:1476, 25:1477,
 25:1478, 25:1479, 25:1480, 25:1481,
 25:1482, 25:1483, 25:1484, 25:1485,
 25:1486, 25:1487, 25:1488, 25:1489,
 25:1490, 25:1491, 25:1492, 25:1493,
 25:1494, 25:1495, 25:1496, 25:1497,
 25:1498, 25:1499, 25:1500, 25:1501,
 25:1502, 25:1503, 25:1504, 25:1505,
 25:1506, 25:1507, 25:1508, 25:1509,
 25:1510, 25:1511, 25:1512, 25:1513,
 25:1514, 25:1515, 25:1516, 25:1517,
 25:1518, 25:1519, 25:1520, 25:1521,
 25:1522, 25:1523, 25:1524, 25:1525,
 25:1526, 25:1527, 25:1528, 25:1529,
 25:1530, 25:1531, 25:1532, 25:1533,
 25:1534, 25:1535, 25:1536, 25:1537,
 25:1538, 25:1539, 25:1540, 25:1541,
 25:1542, 25:1543, 25:1544, 25:1545,
 25:1546, 25:1547, 25:1548, 25:1549,
 25:1550, 25:1551, 25:1552, 25:1553,
 25:1554, 25:1555, 25:1556, 25:1557,
 25:1558, 25:1559, 25:1560, 25:1561,
 25:1562, 25:1563, 25:1564, 25:1565,
 25:1566, 25:1567, 25:1568, 25:1569,
 25:1570, 25:1571, 25:1572, 25:1573,
 25:1574, 25:1575, 25:1576, 25:1577,
 25:1578, 25:1579, 25:1580, 25:1581,
 25:1582, 25:1583, 25:1584, 25:1585,
 25:1586, 25:1587, 25:1588, 25:1589,
 25:1590, 25:1591, 25:1592, 25:1593,
 25:1594, 25:1595, 25:1596, 25:1597,
 25:1598, 25:1599, 25:1600, 25:1601,
 25:1602, 25:1603, 25:1604, 25:1605,
 25:1606, 25:1607, 25:1608, 25:1609,
 25:1610, 25:1611, 25:1612, 25:1613,
 25:1614, 25:1615, 25:1616, 25:1617,
 25:1618, 25:1619, 25:1620, 25:1621,
 25:1622, 25:1623, 25:1624, 25:1625,
 25:1626, 25:1627, 25:1628, 25:1629,
 25:1630, 25:1631, 25:1632, 25:1633,
 25:1634, 25:1635, 25:1636, 25:1637,
 25:1638, 25:1639, 25:1640, 25:1641,
 25:1642, 25:1643, 25:1644, 25:1645,
 25:1646, 25:1647, 25:1648, 25:1649,
 25:1650, 25:1651, 25:1652, 25:1653,
 25:1654, 25:1655, 25:1656, 25:1657,
 25:1658, 25:1659, 25:1660, 25:1661,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:1662, 25:1663, 25:1664, 25:1665,
 25:1666, 25:1667, 25:1668, 25:1669,
 25:1670, 25:1671, 25:1672, 25:1673,
 25:1674, 25:1675, 25:1676, 25:1677,
 25:1678, 25:1679, 25:1680, 25:1681,
 25:1682, 25:1683, 25:1684, 25:1685,
 25:1686, 25:1687, 25:1688, 25:1689,
 25:1690, 25:1691, 25:1692, 25:1693,
 25:1694, 25:1695, 25:1696, 25:1697,
 25:1698, 25:1699, 25:1700, 25:1701,
 25:1702, 25:1703, 25:1704, 25:1705,
 25:1706, 25:1707, 25:1708, 25:1709,
 25:1710, 25:1711, 25:1712, 25:1713,
 25:1714, 25:1715, 25:1716, 25:1717,
 25:1718, 25:1719, 25:1720, 25:1721,
 25:1722, 25:1723, 25:1724, 25:1725,
 25:1726, 25:1727, 25:1728, 25:1729,
 25:1730, 25:1731, 25:1732, 25:1733,
 25:1734, 25:1735, 25:1736, 25:1737,
 25:1738, 25:1739, 25:1740, 25:1741,
 25:1742, 25:1743, 25:1744, 25:1745,
 25:1746, 25:1747, 25:1748, 25:1749,
 25:1750, 25:1751, 25:1752, 25:1753,
 25:1754, 25:1755, 25:1756, 25:1757,
 25:1758, 25:1759, 25:1760, 25:1761,
 25:1762, 25:1763, 25:1764, 25:1765,
 25:1766, 25:1767, 25:1768, 25:1769,
 25:1770, 25:1771, 25:1772, 25:1773,
 25:1774, 25:1775, 25:1776, 25:1777,
 25:1778, 25:1779, 25:1780, 25:1781,
 25:1782, 25:1783, 25:1784, 25:1785,
 25:1786, 25:1787, 25:1788, 25:1789,
 25:1790, 25:1791, 25:1792, 25:1793,
 25:1794, 25:1795, 25:1796, 25:1797,
 25:1798, 25:1799, 25:1800, 25:1801,
 25:1802, 25:1803, 25:1804, 25:1805,
 25:1806, 25:1807, 25:1808, 25:1809,
 25:1810, 25:1811, 25:1812, 25:1813,
 25:1814, 25:1815, 25:1816, 25:1817,
 25:1818, 25:1819, 25:1820, 25:1821,
 25:1822, 25:1823, 25:1824, 25:1825,
 25:1826, 25:1827, 25:1828, 25:1829,
 25:1830, 25:1831, 25:1832, 25:1833,
 25:1834, 25:1835, 25:1836, 25:1837,
 25:1838, 25:1839, 25:1840, 25:1841,
 25:1842, 25:1843, 25:1844, 25:1845,
 25:1846, 25:1847, 25:1848, 25:1849,
 25:1850, 25:1851, 25:1852, 25:1853,
 25:1854, 25:1855, 25:1856, 25:1857,
 25:1858, 25:1859, 25:1860, 25:1861,
 25:1862, 25:1863, 25:1864, 25:1865,
 25:1866, 25:1867, 25:1868, 25:1869,
 25:1870, 25:1871, 25:1872, 25:1873,
 25:1874, 25:1875, 25:1876, 25:1877,
 25:1878, 25:1879, 25:1880, 25:1881,
 25:1882, 25:1883, 25:1884, 25:1885,
 25:1886, 25:1887, 25:1888, 25:1889,
 25:1890, 25:1891, 25:1892, 25:1893,
 25:1894, 25:1895, 25:1896, 25:1897,
 25:1898, 25:1899, 25:1900, 25:1901,
 25:1902, 25:1903, 25:1904, 25:1905,
 25:1906, 25:1907, 25:1908, 25:1909,
 25:1910, 25:1911, 25:1912, 25:1913,
 25:1914, 25:1915, 25:1916, 25:1917,
 25:1918, 25:1919, 25:1920, 25:1921,
 25:1922, 25:1923, 25:1924, 25:1925,
 25:1926, 25:1927, 25:1928, 25:1929,
 25:1930, 25:1931, 25:1932, 25:1933,
 25:1934, 25:1935, 25:1936, 25:1937,
 25:1938, 25:1939, 25:1940, 25:1941,
 25:1942, 25:1943, 25:1944, 25:1945,
 25:1946, 25:1947, 25:1948, 25:1949,
 25:1950, 25:1951, 25:1952, 25:1953,
 25:1954, 25:1955, 25:1956, 25:1957,
 25:1958, 25:1959, 25:1960, 25:1961,
 25:1962, 25:1963, 25:1964, 25:1965,
 25:1966, 25:1967, 25:1968, 25:1969,
 25:1970, 25:1971, 25:1972, 25:1973,
 25:1974, 25:1975, 25:1976, 25:1977,
 25:1978, 25:1979, 25:1980, 25:1981,
 25:1982, 25:1983, 25:1984, 25:1985,
 25:1986, 25:1987, 25:1988, 25:1989,
 25:1990, 25:1991, 25:1992, 25:1993,
 25:1994, 25:1995, 25:1996, 25:1997,
 25:1998, 25:1999, 25:2000, 25:2001,
 25:2002, 25:2003, 25:2004, 25:2005,
 25:2006, 25:2007, 25:2008, 25:2009,
 25:2010, 25:2011, 25:2012, 25:2013,
 25:2014, 25:2015, 25:2016, 25:2017,
 25:2018, 25:2019, 25:2020, 25:2021,
 25:2022, 25:2023, 25:2024, 25:2025,
 25:2026, 25:2027, 25:2028, 25:2029,
 25:2030, 25:2031, 25:2032, 25:2033,
 25:2034, 25:2035, 25:2036, 25:2037,
 25:2038, 25:2039, 25:2040, 25:2041,
 25:2042, 25:2043, 25:2044, 25:2045,
 25:2046, 25:2047, 25:2048, 25:2049,
 25:2050, 25:2051, 25:2052, 25:2053,
 25:2054, 25:2055, 25:2056, 25:2057,
 25:2058, 25:2059, 25:2060, 25:2061,
 25:2062, 25:2063, 25:2064, 25:2065,
 25:2066, 25:2067, 25:2068, 25:2069,
 25:2070, 25:2071, 25:2072, 25:2073,
 25:2074, 25:2075, 25:2076, 25:2077,
 25:2078, 25:2079, 25:2080, 25:2081,
 25:2082, 25:2083, 25:2084, 25:2085,
 25:2086, 25:2087, 25:2088, 25:2089,
 25:2090, 25:2091, 25:2092, 25:2093,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:2094, 25:2095, 25:2096, 25:2097,
 25:2098, 25:2099, 25:2100, 25:2101,
 25:2102, 25:2103, 25:2104, 25:2105,
 25:2106, 25:2107, 25:2108, 25:2109,
 25:2110, 25:2111, 25:2112, 25:2113,
 25:2114, 25:2115, 25:2116, 25:2117,
 25:2118, 25:2119, 25:2120, 25:2121,
 25:2122, 25:2123, 25:2124, 25:2125,
 25:2126, 25:2127, 25:2128, 25:2129,
 25:2130, 25:2131, 25:2132, 25:2133,
 25:2134, 25:2135, 25:2136, 25:2137,
 25:2138, 25:2139, 25:2140, 25:2141,
 25:2142, 25:2143, 25:2144, 25:2145,
 25:2146, 25:2147, 25:2148, 25:2149,
 25:2150, 25:2151, 25:2152, 25:2153,
 25:2154, 25:2155, 25:2156, 25:2157,
 25:2158, 25:2159, 25:2160, 25:2161,
 25:2162, 25:2163, 25:2164, 25:2165,
 25:2166, 25:2167, 25:2168, 25:2169,
 25:2170, 25:2171, 25:2172, 25:2173,
 25:2174, 25:2175, 25:2176, 25:2177,
 25:2178, 25:2179, 25:2180, 25:2181,
 25:2182, 25:2183, 25:2184, 25:2185,
 25:2186, 25:2187, 25:2188, 25:2189,
 25:2190, 25:2191, 25:2192, 25:2193,
 25:2194, 25:2195, 25:2196, 25:2197,
 25:2198, 25:2199, 25:2200, 25:2201,
 25:2202, 25:2203, 25:2204, 25:2205,
 25:2206, 25:2207, 25:2208, 25:2209,
 25:2210, 25:2211, 25:2212, 25:2213,
 25:2214, 25:2215, 25:2216, 25:2217,
 25:2218, 25:2219, 25:2220, 25:2221,
 25:2222, 25:2223, 25:2224, 25:2225,
 25:2226, 25:2227, 25:2228, 25:2229,
 25:2230, 25:2231, 25:2232, 25:2233,
 25:2234, 25:2235, 25:2236, 25:2237,
 25:2238, 25:2239, 25:2240, 25:2241,
 25:2242, 25:2243, 25:2244, 25:2245,
 25:2246, 25:2247, 25:2248, 25:2249,
 25:2250, 25:2251, 25:2252, 25:2253,
 25:2254, 25:2255, 25:2256, 25:2257,
 25:2258, 25:2259, 25:2260, 25:2261,
 25:2262, 25:2263, 25:2264, 25:2265,
 25:2266, 25:2267, 25:2268, 25:2269,
 25:2270, 25:2271, 25:2272, 25:2273,
 25:2274, 25:2275, 25:2276, 25:2277,
 25:2278, 25:2279, 25:2280, 25:2281,
 25:2282, 25:2283, 25:2284, 25:2285,
 25:2286, 25:2287, 25:2288, 25:2289,
 25:2290, 25:2291, 25:2292, 25:2293,
 25:2294, 25:2295, 25:2296, 25:2297,
 25:2298, 25:2299, 25:2300, 25:2301,
 25:2302, 25:2303, 25:2304, 25:2305,
 25:2306, 25:2307, 25:2308, 25:2309,
 25:2310, 25:2311, 25:2312, 25:2313,
 25:2314, 25:2315, 25:2316, 25:2317,
 25:2318, 25:2319, 25:2320, 25:2321,
 25:2322, 25:2323, 25:2324, 25:2325,
 25:2326, 25:2327, 25:2328, 25:2329,
 25:2330, 25:2331, 25:2332, 25:2333,
 25:2334, 25:2335, 25:2336, 25:2337,
 25:2338, 25:2339, 25:2340, 25:2341,
 25:2342, 25:2343, 25:2344, 25:2345,
 25:2346, 25:2347, 25:2348, 25:2349,
 25:2350, 25:2351, 25:2352, 25:2353,
 25:2354, 25:2355, 25:2356, 25:2357,
 25:2358, 25:2359, 25:2360, 25:2361,
 25:2362, 25:2363, 25:2364, 25:2365,
 25:2366, 25:2367, 25:2368, 25:2369,
 25:2370, 25:2371, 25:2372, 25:2373,
 25:2374, 25:2375, 25:2376, 25:2377,
 25:2378, 25:2379, 25:2380, 25:2381,
 25:2382, 25:2383, 25:2384, 25:2385,
 25:2386, 25:2387, 25:2388, 25:2389,
 25:2390, 25:2391, 25:2392, 25:2393,
 25:2394, 25:2395, 25:2396, 25:2397,
 25:2398, 25:2399, 25:2400, 25:2401,
 25:2402, 25:2403, 25:2404, 25:2405,
 25:2406, 25:2407, 25:2408, 25:2409,
 25:2410, 25:2411, 25:2412, 25:2413,
 25:2414, 25:2415, 25:2416, 25:2417,
 25:2418, 25:2419, 25:2420, 25:2421,
 25:2422, 25:2423, 25:2424, 25:2425,
 25:2426, 25:2427, 25:2428, 25:2429,
 25:2430, 25:2431, 25:2432, 25:2433,
 25:2434, 25:2435, 25:2436, 25:2437,
 25:2438, 25:2439, 25:2440, 25:2441,
 25:2442, 25:2443, 25:2444, 25:2445,
 25:2446, 25:2447, 25:2448, 25:2449,
 25:2450, 25:2451, 25:2452, 25:2453,
 25:2454, 25:2455, 25:2456, 25:2457,
 25:2458, 25:2459, 25:2460, 25:2461,
 25:2462, 25:2463, 25:2464, 25:2465,
 25:2466, 25:2467, 25:2468, 25:2469,
 25:2470, 25:2471, 25:2472, 25:2473,
 25:2474, 25:2475, 25:2476, 25:2477,
 25:2478, 25:2479, 25:2480, 25:2481,
 25:2482, 25:2483, 25:2484, 25:2485,
 25:2486, 25:2487, 25:2488, 25:2489,
 25:2490, 25:2491, 25:2492, 25:2493,
 25:2494, 25:2495, 25:2496, 25:2497,
 25:2498, 25:2499, 25:2500, 25:2501,
 25:2502, 25:2503, 25:2504, 25:2505,
 25:2506, 25:2507, 25:2508, 25:2509,
 25:2510, 25:2511, 25:2512, 25:2513,
 25:2514, 25:2515, 25:2516, 25:2517,
 25:2518, 25:2519, 25:2520, 25:2521,
 25:2522, 25:2523, 25:2524, 25:2525,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:2526, 25:2527, 25:2528, 25:2529,	25:2742, 25:2743, 25:2744, 25:2745,
25:2530, 25:2531, 25:2532, 25:2533,	25:2746, 25:2747, 25:2748, 25:2749,
25:2534, 25:2535, 25:2536, 25:2537,	25:2750, 25:2751, 25:2752, 25:2753,
25:2538, 25:2539, 25:2540, 25:2541,	25:2754, 25:2755, 25:2756, 25:2757,
25:2542, 25:2543, 25:2544, 25:2545,	25:2758, 25:2759, 25:2760, 25:2761,
25:2546, 25:2547, 25:2548, 25:2549,	25:2762, 25:2763, 25:2764, 25:2765,
25:2550, 25:2551, 25:2552, 25:2553,	25:2766, 25:2767, 25:2768, 25:2769,
25:2554, 25:2555, 25:2556, 25:2557,	25:2770, 25:2774, 25:2776, 1438
25:2558, 25:2559, 25:2560, 25:2561,	<code>\DeclareFontSeriesDefault</code> 736
25:2562, 25:2563, 25:2564, 25:2565,	<code>\DeclareFontSeriesDefault</code>
25:2566, 25:2567, 25:2568, 25:2569,	 29:35, 29:36, 29:70, 29:72,
25:2570, 25:2571, 25:2572, 25:2573,	29:73, 29:83, 29:94, 29:103, 29:105, 744
25:2574, 25:2575, 25:2576, 25:2577,	<code>\DeclareFontShape</code> 24:19,
25:2578, 25:2579, 25:2580, 25:2581,	24:543, 24:544, 24:545, 24:546,
25:2582, 25:2583, 25:2584, 25:2585,	24:547, 24:548, 24:549, 24:550,
25:2586, 25:2587, 25:2588, 25:2589,	24:551, 24:552, 24:553, 24:554,
25:2590, 25:2591, 25:2592, 25:2593,	24:555, 24:556, 24:557, 24:558,
25:2594, 25:2595, 25:2596, 25:2597,	24:559, 24:560, 24:561, 24:562,
25:2598, 25:2599, 25:2600, 25:2601,	24:563, 27:25, 27:27, 27:81, 27:82, 1360
25:2602, 25:2603, 25:2604, 25:2605,	<code>\DeclareFontShapeChangeRule</code> 25:2912,
25:2606, 25:2607, 25:2608, 25:2609,	25:2934, 25:2947, 25:2949, 25:2950,
25:2610, 25:2611, 25:2612, 25:2613,	25:2951, 25:2952, 25:2953, 25:2954,
25:2614, 25:2615, 25:2616, 25:2617,	25:2955, 25:2956, 25:2957, 25:2958,
25:2618, 25:2619, 25:2620, 25:2621,	25:2959, 25:2960, 25:2961, 25:2962,
25:2622, 25:2623, 25:2624, 25:2625,	25:2963, 25:2964, 25:2965, 25:2966,
25:2626, 25:2627, 25:2628, 25:2629,	25:2967, 25:2968, 25:2969, 25:2970,
25:2630, 25:2631, 25:2632, 25:2633,	25:2971, 25:2972, 25:2973, 25:2974,
25:2634, 25:2635, 25:2636, 25:2637,	25:2975, 25:2976, 25:2977, 25:2978,
25:2638, 25:2639, 25:2640, 25:2641,	25:2979, 25:2980, 25:2981, 25:2982,
25:2642, 25:2643, 25:2644, 25:2645,	25:2983, 25:2984, 25:2985, 25:2986,
25:2646, 25:2647, 25:2648, 25:2649,	25:2987, 25:2988, 25:2989, 25:2990,
25:2650, 25:2651, 25:2652, 25:2653,	25:2991, 25:2992, 25:2993, 25:2994,
25:2654, 25:2655, 25:2656, 25:2657,	25:2995, 25:2996, 25:2997, 25:2998,
25:2658, 25:2659, 25:2660, 25:2661,	25:2999, 25:3000, 25:3001, 25:3002,
25:2662, 25:2663, 25:2664, 25:2665,	25:3003, 25:3004, 25:3005, 25:3006,
25:2666, 25:2667, 25:2668, 25:2669,	25:3007, 25:3008, 25:3009, 25:3010,
25:2670, 25:2671, 25:2672, 25:2673,	25:3011, 25:3012, 25:3013, 25:3014,
25:2674, 25:2675, 25:2676, 25:2677,	25:3015, 25:3016, 25:3017, 25:3018,
25:2678, 25:2679, 25:2680, 25:2681,	25:3019, 25:3020, 25:3021, 25:3022,
25:2682, 25:2683, 25:2684, 25:2685,	25:3023, 25:3024, 25:3025, 25:3026,
25:2686, 25:2687, 25:2688, 25:2689,	25:3027, 25:3028, 25:3029, 25:3030,
25:2690, 25:2691, 25:2692, 25:2693,	25:3031, 25:3032, 25:3033, 25:3034,
25:2694, 25:2695, 25:2696, 25:2697,	25:3035, 25:3036, 25:3037, 25:3038,
25:2698, 25:2699, 25:2700, 25:2701,	25:3039, 25:3040, 25:3041, 25:3042,
25:2702, 25:2703, 25:2704, 25:2705,	25:3043, 25:3044, 25:3045, 25:3049,
25:2706, 25:2707, 25:2708, 25:2709,	25:3051, 25:3052, 25:3053, 25:3054,
25:2710, 25:2711, 25:2712, 25:2713,	25:3055, 25:3056, 25:3057, 25:3058,
25:2714, 25:2715, 25:2716, 25:2717,	25:3059, 25:3060, 25:3061, 25:3062,
25:2718, 25:2719, 25:2720, 25:2721,	25:3063, 25:3064, 25:3065, 25:3066,
25:2722, 25:2723, 25:2724, 25:2725,	25:3067, 25:3068, 25:3069, 25:3070,
25:2726, 25:2727, 25:2728, 25:2729,	25:3071, 25:3072, 25:3073, 25:3074,
25:2730, 25:2731, 25:2732, 25:2733,	25:3075, 25:3076, 25:3077, 25:3078,
25:2734, 25:2735, 25:2736, 25:2737,	25:3079, 25:3080, 25:3081, 25:3082,
25:2738, 25:2739, 25:2740, 25:2741,	25:3083, 25:3084, 25:3085, 25:3086,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

25:3087, 25:3088, 25:3090, 25:3091,	30:197, 30:198, 30:199, 30:200,
25:3092, 25:3093, 25:3095, 25:3096,	30:201, 30:202, 30:203, 30:204,
25:3097, 25:3099, 25:3100, 25:3101,	30:205, 30:206, 30:207, 30:208,
25:3102, 25:3103, 25:3104, 25:3106,	30:209, 30:210, 30:211, 30:212,
25:3107, 25:3108, 25:3109, 25:3110,	30:213, 30:214, 30:215, 30:216,
25:3111, 25:3112, 25:3114, 25:3115,	30:217, 30:218, 30:219, 30:220,
25:3116, 25:3117, 25:3118, 25:3119,	30:221, 30:222, 30:223, 30:224,
25:3121, 25:3122, 25:3123, 25:3124,	30:225, 30:226, 30:227, 30:228,
25:3125, 25:3126, 25:3127, 25:3130	30:229, 30:230, 30:231, 30:232,
\DeclareFontSubstitution 21:398,	30:233, 30:234, 30:235, 30:236,
21:486, 21:769, 21:855, 24:142,	30:237, 30:238, 30:239, 30:240,
30:26, 30:34, 30:37, 30:38, 30:39,	30:241, 30:242, 30:243, 30:244,
30:141, 30:142, 30:143, 30:144, 1439	30:245, 30:246, 30:247, 30:248,
\DeclareHookRule . . . 08:2848, 08:2956,	30:249, 30:250, 30:251, 30:252,
37:52, 37:53, 37:117, 37:118, 220	30:253, 30:254, 30:255, 30:256,
\DeclareInstance 11:1549, 377	30:257, 30:258, 30:259, 30:260,
\DeclareInstanceCopy 11:1549, 376	30:261, 30:262, 30:272, 30:273,
\DeclareKeys 51:214, 1150	30:275, 30:279, 30:280, 30:281,
\DeclareLowercaseExclusions 57:620, 1439	30:282, 30:283, 30:284, 30:285,
\DeclareLowercaseMapping . . . 57:620, 1430	30:286, 30:287, 30:288, 30:289,
\DeclareMathAccent 28:860,	30:290, 30:291, 30:292, 30:293,
30:527, 30:528, 30:529, 30:530,	30:294, 30:295, 30:296, 30:297,
30:531, 30:532, 30:533, 30:534,	30:298, 30:299, 30:300, 30:301,
30:535, 30:536, 30:537, 30:538, 30:539	30:302, 30:303, 30:304, 30:305,
\DeclareMathAlphabet 27:119,	30:306, 30:307, 30:308, 30:309,
27:123, 27:125, 27:132, 28:686,	30:310, 30:311, 30:312, 30:313,
28:849, 30:162, 30:163, 30:164, 30:165	30:314, 30:315, 30:316, 30:317,
\DeclareMathAlphabetCharacter . 28:1035	30:318, 30:319, 30:320, 30:321,
\DeclareMathDelimiter	30:322, 30:323, 30:324, 30:325,
. 28:1037, 30:266, 30:267,	30:326, 30:327, 30:328, 30:329,
30:268, 30:269, 30:270, 30:271,	30:330, 30:331, 30:332, 30:333,
30:274, 30:276, 30:277, 30:574,	30:334, 30:335, 30:336, 30:338,
30:576, 30:578, 30:580, 30:582,	30:339, 30:340, 30:341, 30:342,
30:585, 30:587, 30:589, 30:591,	30:343, 30:344, 30:345, 30:352,
30:593, 30:595, 30:597, 30:599,	30:353, 30:354, 30:355, 30:356,
30:601, 30:603, 30:605, 30:607,	30:357, 30:358, 30:360, 30:361,
30:609, 30:611, 30:613, 30:615,	30:362, 30:363, 30:364, 30:365,
30:617, 30:619, 30:621, 30:623, 1401	30:367, 30:368, 30:369, 30:370,
\DeclareMathOperator 1386	30:371, 30:372, 30:375, 30:376,
\DeclareMathRadical 28:1172, 30:540	30:377, 30:378, 30:381, 30:382,
\DeclareMathScriptfontMapping	30:383, 30:384, 30:385, 30:386,
. 26:354, 1440	30:387, 30:388, 30:389, 30:390,
\DeclareMathSizes	30:391, 30:392, 30:393, 30:394,
. 24:268, 24:274, 24:296,	30:395, 30:396, 30:397, 30:398,
30:168, 30:169, 30:170, 30:171,	30:399, 30:400, 30:401, 30:402,
30:172, 30:173, 30:174, 30:175,	30:403, 30:404, 30:405, 30:406,
30:176, 30:177, 30:178, 30:179, 1384	30:407, 30:408, 30:409, 30:410,
\DeclareMathSizes* 24:268	30:411, 30:412, 30:413, 30:414,
\DeclareMathSymbol	30:415, 30:416, 30:417, 30:418,
. 28:973, 28:1036, 28:1053, 30:180,	30:419, 30:420, 30:421, 30:422,
30:181, 30:182, 30:183, 30:184,	30:425, 30:426, 30:429, 30:430,
30:185, 30:186, 30:187, 30:188,	30:431, 30:432, 30:433, 30:434,
30:189, 30:190, 30:191, 30:192,	30:435, 30:436, 30:437, 30:438,
30:193, 30:194, 30:195, 30:196,	30:439, 30:440, 30:441, 30:443,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 30:444, 30:445, 30:446, 30:447,
 30:448, 30:449, 30:452, 30:453,
 30:454, 30:456, 30:457, 30:458,
 30:459, 30:460, 30:461, 30:462,
 30:463, 30:464, 30:465, 30:466,
 30:488, 30:490, 30:512, 30:513,
 30:514, 30:564, 30:565, 30:566,
 30:567, 30:625, 30:626, 30:627, 779
 \DeclareMathVersion ... 28:468, 29:3, 29:4
 \DeclareOldFontCommand .. 32:126, 32:142
 \DeclareOption 1098
 \DeclareOption
 21:1563, 26:30, 26:38, 26:46,
 26:54, 26:57, 26:61, 33:782, 33:783,
 33:784, 33:785, 33:786, 33:788,
 33:790, 33:792, 33:793, 33:833,
 33:834, 33:835, 33:836, 33:837,
 33:839, 33:840, 50:531, 50:1236, 1388
 \DeclareOption* 1098
 \DeclareOption* 50:531
 \DeclarePreloadSizes
 24:248, 27:95, 27:96,
 31:19, 31:21, 31:22, 31:23, 31:25,
 31:26, 31:27, 31:28, 31:29, 31:30,
 31:34, 31:38, 31:43, 31:45, 31:49,
 31:50, 31:53, 31:54, 31:57, 31:58, 31:64
 \DeclareProtectedCommand 1376
 \DeclareRelease .. 33:775, 33:776, 50:1714
 DeclareRelease commands:
 \DeclareRelease: 50:1717
 \DeclareRobustCommand
 14:4, 14:11, 14:30, 14:57, 18:7, 18:8,
 18:9, 18:10, 18:11, 18:69, 18:93,
 18:399, 18:475, 18:489, 18:534,
 18:559, 19:2, 19:3, 19:13, 20:488,
 20:635, 21:168, 21:176, 21:327,
 21:330, 21:331, 21:332, 21:333,
 21:334, 21:336, 21:338, 21:340,
 22:356, 23:32, 23:33, 23:34, 24:314,
 24:342, 24:343, 24:344, 24:349,
 24:361, 24:371, 24:379, 24:381,
 24:399, 24:738, 24:748, 25:2782,
 25:2786, 25:2795, 25:2797, 25:2807,
 25:2914, 25:2919, 25:2924, 25:3135,
 25:3138, 25:3146, 25:3147, 25:3153,
 25:3233, 26:116, 26:165, 29:5, 29:8,
 29:11, 29:14, 29:17, 29:20, 29:23,
 29:26, 29:29, 29:333, 29:356, 29:386,
 29:407, 29:431, 29:442, 29:459,
 29:462, 29:484, 29:489, 29:494,
 29:527, 29:532, 29:537, 29:553,
 29:556, 29:559, 29:562, 29:565,
 29:579, 29:632, 29:633, 29:668,
 29:674, 29:684, 29:706, 29:724,
 29:726, 29:735, 29:737, 29:744,
 29:760, 29:768, 29:801, 29:817,
 29:833, 29:850, 29:866, 29:873,
 30:346, 30:347, 30:348, 30:359,
 30:366, 30:423, 30:424, 30:455,
 30:467, 30:471, 30:474, 30:479,
 30:481, 30:483, 30:486, 30:489,
 30:491, 30:492, 30:494, 30:496,
 30:498, 30:500, 30:502, 30:504,
 30:506, 30:508, 30:510, 30:516,
 30:518, 30:520, 30:523, 30:541,
 30:544, 30:547, 30:551, 30:555,
 30:558, 30:561, 30:568, 30:571,
 30:628, 30:629, 30:630, 30:635,
 30:637, 30:639, 30:641, 32:3, 32:127,
 33:4, 33:11, 33:574, 33:1097, 35:213,
 35:219, 06:274, 37:271, 37:452,
 37:457, 37:462, 37:472, 37:476,
 37:480, 37:578, 37:582, 38:3, 38:4,
 38:5, 38:6, 38:7, 38:8, 38:9, 38:10,
 38:11, 38:12, 38:13, 38:14, 38:15,
 38:16, 38:17, 38:18, 38:19, 38:20,
 38:21, 38:22, 38:23, 38:24, 38:25,
 38:26, 38:27, 38:28, 38:29, 38:30,
 38:31, 38:32, 38:33, 38:34, 38:35,
 38:39, 38:41, 38:42, 38:43, 38:44,
 38:45, 38:46, 38:47, 38:48, 38:49,
 38:50, 38:51, 38:52, 38:81, 38:82,
 38:83, 38:84, 38:126, 38:167, 38:169,
 38:173, 38:218, 38:220, 38:222,
 38:224, 38:227, 38:228, 38:230,
 38:237, 38:239, 38:240, 38:251,
 38:274, 38:276, 38:309, 38:322,
 38:337, 38:348, 38:411, 38:412,
 38:413, 38:522, 38:556, 38:580,
 40:7, 40:24, 40:164, 40:177, 40:200,
 40:201, 40:217, 40:228, 40:307,
 40:597, 40:619, 40:657, 40:671,
 40:726, 40:727, 40:728, 40:729,
 40:730, 40:731, 41:139, 41:142,
 41:154, 42:135, 42:138, 42:141, 44:7,
 44:8, 44:9, 44:10, 44:14, 44:247,
 45:402, 45:454, 47:16, 47:28, 48:520,
 48:521, 49:23, 49:28, 49:38, 49:49,
 49:64, 49:72, 49:126, 49:128, 49:130,
 49:137, 50:1171, 50:1172, 50:1173,
 50:1179, 50:1180, 06:925, 06:926,
 06:932, 06:947, 50:1838, 52:148,
 52:184, 53:501, 54:63, 57:786, 1376
 \DeclareSizeFunction 26:434,
 26:507, 26:508, 26:519, 26:520,
 26:524, 26:525, 26:531, 26:532,
 26:560, 26:574, 26:575, 26:582, 26:583
 \DeclareStrongEmphSequence 756

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltxyphen.dtx, 57=ltfinal.dtx

21:641,	21:642,	21:643,	21:644,	21:595,	21:596,	21:597,	21:598,
21:645,	21:646,	21:647,	21:648,	21:599,	21:600,	21:601,	21:602,
21:649,	21:650,	21:651,	21:652,	21:603,	21:604,	21:605,	21:606,
21:653,	21:654,	21:655,	21:656,	21:607,	21:608,	21:609,	21:741,
21:657,	21:658,	21:659,	21:660,	21:742,	21:743,	21:744,	21:745,
21:661,	21:662,	21:663,	21:664,	21:746,	21:747,	21:748,	21:749,
21:665,	21:666,	21:667,	21:668,	21:750,	21:751,	21:752,	21:753,
21:669,	21:670,	21:671,	21:672,	21:763,	21:764,	21:792,	21:793,
21:673,	21:674,	21:675,	21:676,	21:794,	21:795,	21:796,	21:797,
21:677,	21:678,	21:679,	21:680,	21:798,	21:800,	21:801,	21:802,
21:681,	21:682,	21:683,	21:684,	21:803,	21:804,	21:805,	21:806,
21:685,	21:686,	21:687,	21:688,	21:807,	21:808,	21:809,	21:810,
21:689,	21:690,	21:691,	21:692,	21:811,	21:812,	21:813,	21:814,
21:693,	21:694,	21:695,	21:696,	21:815,	21:816,	21:817,	21:818,
21:697,	21:698,	21:699,	21:700,	21:877,	21:878,	21:879,	21:880,
21:701,	21:702,	21:703,	21:704,	21:881,	21:882,	21:883,	21:884,
21:705,	21:706,	21:707,	21:708,	21:885,	21:886,	21:887,	21:888,
21:709,	21:710,	21:711,	21:712,	21:900,	21:901,	21:902,	21:903,
21:713,	21:714,	21:715,	21:716,	21:904,	21:905,	21:906,	21:907,
21:717,	21:718,	21:719,	21:720,	21:908,	21:909,	21:910,	21:911,
21:836,	21:837,	21:838,	21:839,	21:912,	21:913,	21:914,	21:915,
21:840,	21:841,	21:842,	21:843,	21:916,	21:923,	21:924,	21:925,
21:844,	21:845,	21:846,	21:847,	21:926,	21:927,	21:928,	21:929,
21:848,	21:849,	21:850,	21:851,	21:930,	21:931,	21:932,	21:933,
\DeclareTextCompositeCommand				21:934,	21:935,	21:936,	21:937,
. 21:94,				21:938,	21:939,	21:940,	21:941,
21:476,				21:942,	21:943,	21:944,	21:945,
21:481,				21:946,	21:947,	21:948,	21:949,
21:727,				21:950,	21:951,	21:952,	21:953,
21:731,				21:954,	21:955,	21:956,	21:957,
21:735,				21:958,	21:959,	21:960,	21:961,
\DeclareTextFontCommand				21:962,	21:963,	21:964,	21:965,
. 29:705,				21:966,	21:967,	21:968,	21:969,
32:16,				21:970,	21:971,	21:972,	21:973,
32:21,				21:974,	21:975,	21:976,	21:977,
32:30,				21:978,	21:979,	21:980,	21:981,
\DeclareTextFoo 1366				21:982,	21:983,	21:984,	21:985,
\DeclareTextGlyph 1373				21:986,	21:987,	21:988,	21:989,
\DeclareTextSymbol 21:3,				21:990,	21:991,	21:992,	21:993,
21:420,				21:994,	21:995,	21:996,	21:997,
21:424,				21:998,	21:999,	21:1000,	21:1001,
21:428,				21:1002,	21:1107,	33:332,	33:333,
21:437,				33:334,	33:335,	33:336,	33:337,
21:441,				33:338,	33:339,	33:350,	33:351,
21:551,				33:352,	33:353,	33:354,	33:355,
21:555,				33:356,	33:357,	33:358,	33:359,
21:559,				33:553,	33:554,	33:555,	33:556,
21:563,				33:557,	33:558,	33:559,	33:560,
21:568,				\DeclareTextSymbolDefault			
21:572,				. . . 21:203,			
21:576,				21:263,			
21:580,				21:267,			
21:584,				21:271,			
21:588,				21:275,			

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

21:279, 21:280, 21:281, 21:282,
 21:283, 21:284, 21:285, 21:286,
 21:287, 21:288, 21:289, 21:290,
 21:291, 21:292, 21:293, 21:294,
 21:295, 21:296, 21:297, 21:298,
 21:300, 21:301, 21:311, 33:78,
 33:80, 33:82, 33:84, 33:85, 33:86,
 33:87, 33:88, 33:89, 33:90, 33:91,
 33:92, 33:93, 33:94, 33:95, 33:96,
 33:97, 33:98, 33:99, 33:100, 33:101,
 33:102, 33:103, 33:104, 33:105,
 33:106, 33:107, 33:108, 33:109,
 33:110, 33:111, 33:112, 33:113,
 33:114, 33:115, 33:116, 33:117,
 33:118, 33:541, 33:542, 33:543,
 33:544, 33:545, 33:546, 33:547,
 33:548, 33:561, 33:562, 33:563,
 33:564, 33:565, 33:566, 33:567,
 33:568, 33:587, 33:588, 33:606,
 33:607, 33:608, 33:609, 33:610,
 33:611, 33:612, 33:614, 33:914,
 33:915, 33:916, 33:917, 33:918,
 33:919, 33:920, 33:921, 33:922,
 33:923, 33:924, 33:925, 33:926,
 33:927, 33:928, 33:929, 33:930,
 33:931, 33:932, 33:933, 33:934,
 33:935, 33:936, 33:937, 33:938,
 33:939, 33:940, 33:941, 33:942,
 33:943, 33:944, 33:945, 33:946,
 33:947, 33:948, 33:949, 33:950,
 33:951, 33:952, 33:953, 33:954,
 33:955, 33:956, 33:957, 33:958,
 33:959, 33:960, 33:961, 33:962,
 33:963, 33:964, 33:965, 33:966, 1381
 \DeclareTitlecaseExclusions 57:620, 1439
 \DeclareTitlecaseMapping ... 57:620, 1430
 \DeclareUnicode... 544
 \DeclareUnicodeAccent
 21:1070, 21:1265, 21:1266, 21:1267,
 21:1268, 21:1269, 21:1270, 21:1271,
 21:1272, 21:1273, 21:1274, 21:1275,
 21:1276, 21:1277, 21:1278, 21:1279, 543
 \DeclareUnicodeCharacter ... 57:433, 1411
 \DeclareUnicodeCommand
 21:1108, 21:1109,
 21:1110, 21:1111, 21:1192, 21:1194,
 21:1196, 21:1243, 21:1280, 1424
 \DeclareUnicodeComposite
 21:1085, 21:1284, 21:1285, 21:1286,
 21:1287, 21:1288, 21:1289, 21:1290,
 21:1291, 21:1292, 21:1293, 21:1294,
 21:1295, 21:1296, 21:1297, 21:1298,
 21:1299, 21:1300, 21:1301, 21:1302,
 21:1303, 21:1304, 21:1305, 21:1306,
 21:1307, 21:1308, 21:1309, 21:1310,
 21:1311, 21:1312, 21:1313, 21:1314,
 21:1315, 21:1316, 21:1317, 21:1318,
 21:1319, 21:1320, 21:1321, 21:1322,
 21:1323, 21:1324, 21:1325, 21:1326,
 21:1327, 21:1328, 21:1329, 21:1330,
 21:1331, 21:1332, 21:1333, 21:1334,
 21:1335, 21:1336, 21:1337, 21:1338,
 21:1339, 21:1340, 21:1341, 21:1342,
 21:1343, 21:1344, 21:1345, 21:1346,
 21:1347, 21:1348, 21:1349, 21:1350,
 21:1351, 21:1352, 21:1353, 21:1354,
 21:1355, 21:1356, 21:1357, 21:1358,
 21:1359, 21:1360, 21:1361, 21:1362,
 21:1363, 21:1364, 21:1365, 21:1366,
 21:1367, 21:1368, 21:1369, 21:1370,
 21:1371, 21:1372, 21:1373, 21:1374,
 21:1375, 21:1376, 21:1377, 21:1378,
 21:1379, 21:1380, 21:1381, 21:1382,
 21:1383, 21:1384, 21:1385, 21:1386,
 21:1387, 21:1388, 21:1389, 21:1390,
 21:1391, 21:1392, 21:1393, 21:1394,
 21:1395, 21:1396, 21:1397, 21:1398,
 21:1399, 21:1400, 21:1401, 21:1402,
 21:1403, 21:1404, 21:1405, 21:1406,
 21:1407, 21:1408, 21:1409, 21:1410,
 21:1411, 21:1412, 21:1413, 21:1414,
 21:1415, 21:1416, 21:1417, 21:1418,
 21:1419, 21:1420, 21:1421, 21:1422,
 21:1423, 21:1424, 21:1425, 21:1426,
 21:1427, 21:1428, 21:1429, 21:1430,
 21:1431, 21:1432, 21:1433, 21:1434,
 21:1435, 21:1436, 21:1437, 21:1438,
 21:1439, 21:1440, 21:1441, 21:1442,
 21:1443, 21:1444, 21:1445, 21:1446,
 21:1447, 21:1448, 21:1449, 21:1450,
 21:1451, 21:1452, 21:1453, 21:1454,
 21:1455, 21:1456, 21:1457, 21:1458,
 21:1459, 21:1460, 21:1461, 21:1462,
 21:1463, 21:1464, 21:1465, 21:1466,
 21:1467, 21:1468, 21:1469, 21:1470,
 21:1471, 21:1472, 21:1473, 21:1474,
 21:1475, 21:1476, 21:1477, 21:1478,
 21:1479, 21:1480, 21:1481, 21:1482,
 21:1483, 21:1484, 21:1485, 21:1486,
 21:1487, 21:1488, 21:1489, 21:1490,
 21:1491, 21:1492, 21:1493, 21:1494,
 21:1495, 21:1496, 21:1497, 21:1498,
 21:1499, 21:1500, 21:1501, 21:1502,
 21:1503, 21:1504, 21:1505, 21:1506,
 21:1507, 21:1508, 21:1509, 21:1510,
 21:1511, 21:1512, 21:1513, 21:1514,
 21:1515, 21:1516, 21:1517, 21:1518,
 21:1519, 21:1520, 21:1521, 21:1522,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

21:1523, 21:1524, 21:1525, 21:1526, 21:1527, 21:1528, 21:1529, 21:1530, <i>543</i>	\delimitershortfall 02:272
\DeclareUnicodeSymbol	\Delta 30:309
..... 21:1107, 21:1112, 21:1113, 21:1114, 21:1115, 21:1116, 21:1117, 21:1118, 21:1119, 21:1120, 21:1121, 21:1122, 21:1123, 21:1124, 21:1125, 21:1126, 21:1127, 21:1128, 21:1129, 21:1130, 21:1131, 21:1132, 21:1134, 21:1135, 21:1136, 21:1137, 21:1138, 21:1139, 21:1140, 21:1141, 21:1142, 21:1143, 21:1144, 21:1145, 21:1146, 21:1147, 21:1148, 21:1150, 21:1151, 21:1152, 21:1153, 21:1154, 21:1155, 21:1156, 21:1157, 21:1158, 21:1159, 21:1160, 21:1161, 21:1162, 21:1163, 21:1164, 21:1165, 21:1166, 21:1167, 21:1168, 21:1169, 21:1170, 21:1171, 21:1172, 21:1173, 21:1174, 21:1175, 21:1176, 21:1177, 21:1178, 21:1179, 21:1180, 21:1181, 21:1182, 21:1183, 21:1184, 21:1185, 21:1186, 21:1187, 21:1188, 21:1189, 21:1190, 21:1191, 21:1198, 21:1199, 21:1200, 21:1201, 21:1202, 21:1203, 21:1204, 21:1205, 21:1206, 21:1207, 21:1208, 21:1209, 21:1210, 21:1211, 21:1212, 21:1213, 21:1214, 21:1215, 21:1216, 21:1217, 21:1218, 21:1219, 21:1220, 21:1221, 21:1222, 21:1223, 21:1224, 21:1225, 21:1226, 21:1227, 21:1228, 21:1229, 21:1230, 21:1231, 21:1232, 21:1233, 21:1234, 21:1235, 21:1236, 21:1237, 21:1238, 21:1239, 21:1241, 21:1242, 21:1254, 21:1255, 21:1256, 21:1257, 21:1258, 21:1259, 21:1260, 21:1261, 21:1262, 21:1263, 21:1264, <i>1424</i>	\depth 40:32, 40:35 \det 38:30 \Details 33:1589, 33:1590 \detokenize 03:96, 03:97, 03:182, 14:249, 14:250, 20:273, 20:524, 21:1022, 21:1068, 24:190, 25:2822, 05:137, 29:646, 37:207, 06:666, 06:684, 06:748, 50:599, 50:633, 50:673, 50:945, 50:1312, 50:1316, 50:1444, 50:1445, 50:1449, 57:796, <i>111</i> \DH 21:551, 21:1156, 57:720, <i>1377</i> \dh 21:561, 21:1162, 57:720, <i>1377</i> \Diamond 29:883 \diamond 30:391 \diamondsuit 30:343 \dim 38:28 dim commands: \dimeval:n 05:178 \dimeval:N 11:93, 53:260, 53:261, 53:262, 53:263 \dimeval:Nn 53:254, 53:255, 53:256, 53:257 \dimeval:NN 48:69, 48:71 \dimeval:N 53:585, 53:586, 53:587 \c_max_dim 48:69, 48:71, 48:109, 48:113, 53:274, 53:300, 53:325, 53:352, <i>1072</i> \l_tmpa_dim <i>373</i> \c_zero_dim 16:104, 53:30, 53:36, 53:37, 53:54, 53:56, 53:57, 53:280, 53:281, 53:282, 53:288, 53:306, 53:307, 53:308, 53:331, 53:332, 53:333, 53:360, 53:361, 53:362 \dimen <i>1396</i> \dimenend 04:225, 02:42, 02:43, 02:44, 02:48 \dimenzero 04:225 \dimeval <i>77</i> \dimeval 05:172, 05:188, <i>77</i> \dimexpr 21:526, 21:542, 21:893, 21:896, 21:1247, 21:1250, 40:343, 40:344, 40:346, 40:636, 40:637, 40:640, 42:13, 45:414, 45:420, 45:428, 45:473, 45:479, 45:487 \directlua . 04:2, 04:14, 04:29, 04:212, 04:230, 17:62, 04:255, 04:260, 04:264, 21:1016, 21:1035, 01:12, 01:15, 06:34, 01:20, 40:350, 01:23, 40:644, 45:418, 45:477, 06:846, 01:28, 06:931, 50:1297, 50:1428, 02:61, 02:77, 02:82, 57:98, 57:100,
\DeclareUnknownKeyHandler .. <i>51:216, 1150</i>	
\DeclareUnknownKeysHandler <i>1425</i>	
\DeclareUppercaseExclusions <i>57:620, 1439</i>	
\DeclareUppercaseMapping ... <i>57:620, 1430</i>	
\def <i>1357</i>	
default (plug) <i>1234</i>	
default (tag plug) <i>55:108</i>	
\defaultencoding <i>1378</i>	
\defaultthyphenchar . 06:935, 06:950, 02:253	
\defaultscriptratio . 24:721, <i>24:728, 1371</i>	
\defaultscriptscriptratio <i>24:722, 24:728, 1371</i>	
\defaultskewchar 02:254	
\deg 38:34	
\delcode 28:1170	
\delimiter 28:1083, 28:1153, 28:1164, <i>1409</i>	
\delimiterfactor 02:257	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 57:108, 57:113, 57:115, 57:318,
57:356, 57:370, 02:142, 02:228, 02:238
- disable commands:
- `\disable_callback` 04:989
 - `\disable_callback` 44
 - `\DisableGenericHook`
..... 08:2786, 08:2792, 08:2940, 250
 - `\DisableHook` 08:2891, 218
 - `\DiscardShipoutBox`
..... 53:143, 53:464, 53:497, 53:574, 1186
 - `\discretionary`
..... 38:251, 06:933, 06:948, 06:958
 - `\displayindent` 02:278
 - `\displaylines` 38:203
 - `\displaymath (env.)` 38:389
 - `\displaymath` 38:391
 - `\displaystyle` .. 30:543, 30:546, 30:550,
30:552, 38:62, 38:210, 38:434,
38:437, 38:463, 38:466, 38:521,
38:563, 38:575, 38:603, 38:628, 38:631
 - `\displaywidowpenalty` 02:197
 - `\displaywidth` 38:210,
38:433, 38:462, 38:550, 38:606, 02:277
 - `\div` 30:394
 - `\DJ` 21:552, 21:1166, 57:720, 1377
 - `\dj` 21:562, 21:1167, 57:720, 1377
 - `\do` 01:110, 13:3, 13:7, 13:16,
13:26, 20:66, 20:69, 20:72, 20:135,
20:138, 20:190, 20:193, 20:252,
20:324, 20:386, 20:433, 20:606,
20:623, 20:774, 20:780, 06:69, 32:91,
37:543, 37:564, 37:724, 37:734,
37:740, 37:746, 40:78, 40:97, 41:321,
41:346, 42:115, 42:127, 42:134,
42:214, 42:348, 42:350, 42:373,
42:376, 42:405, 42:407, 42:428,
42:433, 42:488, 42:516, 42:545,
42:741, 42:797, 45:65, 45:134,
47:36, 47:65, 47:82, 02:13, 02:14,
50:267, 50:284, 50:578, 50:597,
50:614, 50:632, 50:637, 50:651,
50:656, 50:692, 50:702, 50:1225,
50:1262, 50:1344, 50:1397, 50:1477,
50:1566, 50:1853, 07:1616, 07:1832,
07:2199, 07:2212, 01:58, 01:59, 1391
 - `\DocInput` 26:8, 30:5, 31:5, 56:4
 - `\docclearpage` 1403
 - `\document` 485
 - `document (env.)` 37:8
 - `\document`
..... 16:145, 20:9, 37:260, 47:64, 47:81, 739
 - `\documentclass` 04:16, 26:2, 30:2,
31:2, 50:709, 50:716, 50:775, 50:778,
50:1018, 50:1113, 50:1245, 56:2, 224
 - `\DocumentMetadata` . 17:8, 17:17, 17:21,
17:23, 17:181, 50:808, 50:811, 457
 - `\documentstyle` 50:714, 50:1245
 - `\doesglyphexist` 33:1182,
33:1208, 33:1226, 33:1227, 33:1228,
33:1229, 33:1230, 33:1231, 33:1232,
33:1235, 33:1236, 33:1237, 33:1238,
33:1241, 33:1242, 33:1243, 33:1244,
33:1247, 33:1248, 33:1251, 33:1252,
33:1255, 33:1256, 33:1257, 33:1258,
33:1259, 33:1260, 33:1261, 33:1262,
33:1265, 33:1266, 33:1269, 33:1270,
33:1271, 33:1272, 33:1273, 33:1274,
33:1275, 33:1276, 33:1277, 33:1278,
33:1279, 33:1280, 33:1281, 33:1282,
33:1283, 33:1284, 33:1285, 33:1286,
33:1287, 33:1288, 33:1289, 33:1290,
33:1291, 33:1292, 33:1293, 33:1294,
33:1295, 33:1296, 33:1297, 33:1298,
33:1299, 33:1300, 33:1301, 33:1302,
33:1303, 33:1304, 33:1305, 33:1306,
33:1307, 33:1308, 33:1309, 33:1310,
33:1311, 33:1312, 33:1313, 33:1314,
33:1315, 33:1316, 33:1317, 33:1318,
33:1319, 33:1320, 33:1321, 33:1322,
33:1323, 33:1324, 33:1325, 33:1326,
33:1327, 33:1328, 33:1329, 33:1330,
33:1331, 33:1332, 33:1333, 33:1334,
33:1337, 33:1341, 33:1342, 33:1343,
33:1344, 33:1345, 33:1346, 33:1347,
33:1348, 33:1349, 33:1350, 33:1351,
33:1352, 33:1353, 33:1354, 33:1355,
33:1356, 33:1357, 33:1358, 33:1359,
33:1360, 33:1361, 33:1362, 33:1363,
33:1364, 33:1365, 33:1366, 33:1367,
33:1368, 33:1369, 33:1370, 33:1371, 831
 - `\dollar` 1378
 - `\dospecials` 01:110, 37:543,
37:564, 37:724, 37:734, 02:13,
50:1344, 50:1477, 50:1566, 07:1617,
07:1833, 07:2200, 07:2213, 01:58, 1434
 - `\dot` 30:536
 - `\doteq` 30:480
 - `\dotfill` 06:973, 06:994, 02:421
 - `\dots` 21:340, 21:342, 1385
 - `\doublehyphendemerits` 02:204
 - `\doublerulesep` 41:388, 41:415, 41:439
 - `\Downarrow` 30:597
 - `\downarrow` 30:591
 - `\downbracefill` 30:549, 30:568
 - `\dump` 57:808
- E**
- `\E` 50:1351,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 50:1354, 50:1382, 50:1484, 50:1487,
50:1514, 50:1573, 50:1576, 50:1604
- `\edef` 749
- `\EditInstance` 11:1549, 377
- `\EditTemplateDefaults` 11:1549, 377
- `\egroup` 02:325, 1358
- `\eject` 02:388
- `\ell` 30:322
- `\else` 1369
- else commands:
- `\else:`
 08:1754, 08:2300, 08:2328, 08:2470,
 08:2486, 08:2533, 08:2568, 08:2583,
 09:363, 09:387, 16:26, 16:32, 16:64,
 48:112, 06:813, 52:241, 07:1343,
 07:1840, 07:1841, 07:1842, 07:1843,
 07:3290, 08:789, 08:1063, 08:1220,
 08:1282, 08:1423, 08:1430, 08:1438
- `\em` 29:633, 29:660, 32:42, 758
- `\emergencystretch` .. 49:132, 49:138, 1367
- `\emforce` 755
- `\emforce` 29:629, 29:656, 29:665, 757
- `\emminnershape` . 29:637, 29:643, 29:660, 756
- `\emph` 32:42, 756
- `\empty` 02:323
- `\emptyset` 30:329
- `\emreset` 755
- `\emreset` 29:629, 29:632, 29:663, 755
- `\emrest` 29:632
- `\ENC-cmd` 1377
- `\encodingdefault` 04:259, 04:274, 04:282,
21:1013, 21:1564, 21:1601, 21:1602,
28:408, 28:458, 29:769, 29:802,
29:818, 29:834, 29:851, 30:62, 1378
- `\EncodingSpecific` 1373
- `\EncodingSpecificAccent` 1366
- `\EncodingSpecificAccentedLetter` .. 1366
- `\EncodingSpecificCommand` 1366
- `\end` 14:250, 26:9, 06:23, 30:6, 31:6, 37:270,
37:298, 37:312, 37:348, 37:364,
37:378, 37:386, 37:522, 37:523,
38:584, 38:593, 39:112, 44:15,
44:17, 06:811, 06:868, 50:1359,
50:1363, 50:1370, 50:1492, 50:1496,
50:1502, 50:1581, 50:1585, 50:1591,
53:449, 56:5, 07:1455, 07:1582,
07:1597, 07:1764, 01:53, 07:2882, 1420
- `\end\verbvisiblespace` 37:345
- `\endaligned` 1322
- `\endarray` 41:184, 1321
- `\endcenter` 37:447
- `\endcsname` 359
- `\enddisplaymath` 38:392
- `enddocument` (hook) ... 1206, 238, 239, 243
- `\enddocument` 37:8, 37:79,
37:80, 37:143, 37:145, 53:399, 1391
- `enddocument/afteraux` (hook) 243
- `enddocument/afterlastpage` (hook) ...
..... 1206, 243
- `enddocument/end` (hook) 243
- `enddocument/info` (hook) 1182, 243
- `\endenumerate` 39:271
- `\endeqnarray` 38:442, 38:471, 38:520
- `\endequation` 38:400, 38:407
- `\endfilecontents` 50:1249
- `\endflushleft` 37:497
- `\endflushright` 37:499
- `\endgraf` 16:142, 16:180,
40:138, 40:144, 40:149, 02:320, 1370
- `\endgroup` 918
- `\EndIncludeInRelease`
 08:1567, 02:564, 08:1656, 02:574,
 02:579, 08:1739, 08:1980, 08:2084,
 08:2089, 08:2094, 08:2184, 08:2191,
 08:2215, 08:2231, 08:2238, 08:2243,
 08:2265, 08:2285, 08:2315, 08:2333,
 08:2338, 08:2348, 08:2352, 08:2355,
 08:2382, 08:2397, 08:2408, 08:2426,
 08:2438, 08:2450, 08:2460, 03:71,
 08:2490, 08:2509, 08:2572, 08:2576,
 08:2658, 08:2661, 08:2775, 08:2781,
 08:2788, 08:2793, 08:2800, 08:2804,
 08:2811, 08:2815, 08:2838, 08:2843,
 09:36, 09:47, 09:61, 09:244, 09:311,
 09:454, 09:473, 09:525, 09:529,
 09:576, 09:603, 09:606, 09:618,
 14:157, 14:161, 14:204, 14:210,
 16:44, 16:75, 04:233, 17:138, 17:162,
 17:173, 04:256, 18:21, 18:31, 18:65,
 18:78, 18:86, 18:91, 18:104, 18:110,
 18:155, 18:174, 18:188, 18:200,
 18:211, 18:228, 04:279, 18:240,
 04:283, 18:272, 18:288, 18:305,
 18:341, 18:375, 18:397, 18:434,
 18:467, 18:503, 18:507, 18:516,
 18:520, 18:528, 18:532, 18:542,
 18:548, 18:562, 18:569, 20:83,
 20:140, 20:195, 20:221, 20:234,
 20:275, 20:296, 20:308, 20:371,
 20:375, 20:454, 20:492, 20:515,
 20:550, 20:571, 20:590, 20:597,
 20:616, 20:633, 20:646, 20:650,
 20:668, 20:679, 20:689, 20:728,
 20:755, 21:121, 21:141, 21:186,
 21:193, 21:350, 21:372, 21:387,
 21:395, 22:45, 22:50, 22:75, 22:95,
 22:112, 22:132, 22:141, 22:145,
 22:169, 22:196, 22:201, 22:218,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

22:232, 22:248, 22:254, 22:269,
 22:280, 22:294, 22:301, 22:343,
 22:349, 22:369, 22:372, 23:10, 23:14,
 23:25, 23:30, 24:53, 24:74, 24:227,
 24:246, 24:294, 24:311, 24:357,
 24:367, 24:377, 24:438, 24:447,
 24:527, 24:532, 24:567, 24:572,
 24:589, 24:607, 24:646, 24:679,
 24:797, 24:809, 25:1431, 25:2772,
 25:2778, 25:2791, 25:2803, 25:2810,
 25:2893, 25:2908, 25:2930, 25:2942,
 25:3047, 25:3128, 25:3131, 25:3142,
 25:3149, 25:3156, 25:3229, 25:3247,
 25:3254, 25:3258, 25:3261, 26:158,
 26:161, 26:177, 05:13, 05:22, 05:31,
 26:566, 26:572, 27:21, 27:143,
 28:78, 28:106, 28:180, 28:210,
 28:240, 28:272, 28:322, 28:365,
 28:417, 05:115, 28:464, 28:482,
 28:488, 28:493, 28:576, 28:583,
 28:628, 05:136, 28:635, 05:161,
 28:909, 28:951, 28:964, 28:971,
 05:170, 28:1159, 28:1167, 05:182,
 05:190, 29:68, 29:101, 29:123,
 29:143, 29:156, 05:205, 29:232,
 05:211, 29:285, 29:293, 29:320,
 29:327, 29:379, 05:219, 29:426,
 29:455, 05:226, 29:466, 29:513,
 05:234, 29:547, 05:239, 29:573,
 29:611, 29:622, 06:12, 29:659,
 29:671, 29:678, 06:18, 29:740,
 29:746, 29:797, 29:813, 29:828,
 29:845, 29:860, 29:869, 29:874,
 30:92, 30:102, 30:120, 30:129,
 06:60, 30:644, 30:651, 06:64, 32:33,
 32:40, 06:82, 06:91, 06:148, 06:159,
 35:28, 35:45, 35:61, 35:75, 35:82,
 35:109, 35:124, 35:131, 35:163,
 35:173, 35:182, 35:190, 35:197,
 35:209, 35:215, 35:221, 35:228,
 06:265, 06:270, 37:77, 37:141,
 37:180, 37:188, 37:194, 37:227,
 37:240, 06:306, 37:268, 37:279,
 37:310, 37:335, 37:343, 37:361,
 37:376, 37:384, 37:389, 37:403,
 37:412, 37:422, 06:327, 37:429,
 37:439, 37:444, 37:468, 37:484,
 37:493, 37:512, 37:518, 37:548,
 37:569, 37:591, 37:601, 37:618,
 37:630, 37:634, 37:642, 37:646,
 37:667, 37:680, 37:695, 37:707,
 37:729, 37:737, 38:86, 38:95, 38:117,
 38:124, 38:143, 38:148, 38:156,
 38:160, 01:290, 38:175, 38:183,
 38:232, 38:249, 38:279, 38:287,
 38:296, 38:302, 38:335, 38:360,
 38:387, 38:402, 38:408, 38:449,
 38:477, 38:500, 38:509, 38:541,
 38:546, 38:566, 38:578, 38:587,
 38:596, 06:416, 39:134, 39:139,
 39:161, 39:169, 06:444, 40:13, 40:22,
 01:300, 40:57, 40:65, 40:90, 40:107,
 40:121, 40:132, 40:141, 40:146,
 40:151, 40:157, 06:472, 40:168,
 40:175, 06:477, 40:232, 40:239,
 40:265, 40:276, 40:312, 40:320,
 40:356, 40:363, 40:391, 06:496,
 40:417, 40:438, 40:462, 40:480,
 06:510, 40:557, 40:573, 40:590,
 40:599, 40:604, 40:654, 40:665,
 40:675, 40:682, 40:716, 40:723,
 41:64, 41:69, 41:156, 41:164, 41:176,
 41:182, 41:191, 41:195, 41:210,
 41:216, 41:224, 41:228, 41:254,
 41:276, 41:303, 41:308, 06:551,
 06:561, 42:15, 42:19, 42:27, 42:31,
 42:49, 42:58, 42:79, 42:87, 42:99,
 42:107, 42:122, 42:132, 42:161,
 42:166, 42:182, 42:193, 42:260,
 42:274, 06:593, 42:379, 42:436,
 06:602, 42:473, 42:479, 42:510,
 42:540, 42:565, 42:581, 42:592,
 42:605, 42:613, 42:634, 42:651,
 42:661, 42:665, 06:629, 42:755,
 06:635, 42:810, 42:829, 42:846,
 43:30, 43:39, 43:47, 43:53, 43:70,
 43:77, 44:19, 44:29, 06:649, 06:654,
 44:167, 44:173, 44:178, 44:195,
 44:207, 44:217, 44:223, 01:25,
 44:249, 44:271, 45:104, 45:172,
 45:231, 45:246, 45:293, 45:306,
 45:351, 45:368, 45:437, 45:442,
 45:448, 45:457, 45:461, 45:496,
 45:501, 45:507, 45:511, 45:543,
 45:560, 45:578, 45:620, 45:626,
 47:24, 47:32, 47:76, 47:92, 06:755,
 06:770, 49:34, 49:60, 49:81, 49:104,
 49:115, 49:124, 50:21, 50:26, 06:821,
 50:50, 50:66, 50:78, 06:825, 50:101,
 50:110, 50:116, 50:128, 50:143,
 50:153, 50:177, 50:184, 50:209,
 50:216, 50:234, 50:245, 50:279,
 50:296, 50:307, 50:316, 50:347,
 50:377, 50:394, 06:857, 50:406,
 50:427, 50:440, 50:453, 50:463,
 06:866, 50:499, 50:515, 50:524,
 50:558, 50:568, 50:608, 50:624,
 50:646, 50:661, 50:675, 50:682,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

50:697, 50:706, 50:740, 50:747,	<code>\enditemize</code> 39:282
50:764, 50:771, 50:832, 50:839,	<code>\endline</code> 38:188, <u>02:320</u>
50:877, 50:904, 50:932, 50:1088,	<code>\endlinechar</code>
06:916, 50:1145, 06:923, 50:1175,	... 20:703, 20:737, 01:188, 06:37,
50:1182, 50:1200, 50:1210, 50:1385,	06:39, 06:44, 50:468, 50:469, 50:470,
06:945, 50:1516, 06:956, 06:959,	57:381, <u>02:255</u> , 01:76, 01:77, 01:78, 870
50:1606, 06:987, 06:1008, 52:14,	<code>\endlist</code> <u>39:98</u> , 39:271, 39:282
52:23, 52:88, 52:140, 52:179, 52:191,	<code>\endlrbox</code> 40:199
52:200, 52:245, 52:256, 52:263,	<code>\endmath</code> 38:390
52:298, 52:321, 52:342, 52:356,	<code>\endminipage</code> <u>40:523</u>
52:361, 52:384, 52:404, 52:429,	<code>\EndModuleRelease</code> 03:111,
52:474, 52:494, 52:501, 52:531,	03:112, 08:2962, <u>03:152</u> , 09:642,
52:536, 53:44, 53:63, 53:482, 53:519,	10:234, 11:1637, 16:185, 17:183,
53:534, 53:540, 53:557, 53:561,	33:773, 36:303, 48:531, 55:469, 07:3327
54:157, 54:183, 54:201, 54:370,	<code>\endpicture</code> <u>42:60</u>
54:375, 54:423, 54:469, 54:733,	<code>\endscname</code> 1357
54:817, 54:946, 54:1005, 54:1063,	<code>\endsloppypar</code> 49:136
54:1162, 54:1181, 54:1244, 54:1262,	<code>\endsmallmatrix</code> 1322
54:1304, 54:1325, 54:1568, 54:1712,	<code>\endtabbing</code> <u>41:84</u>
54:1881, 07:241, 54:1964, 54:2044,	<code>\endtabular</code> <u>41:184</u>
07:264, 54:2138, 54:2260, 54:2387,	<code>\endtabular*</code> <u>41:184</u>
54:2497, 54:2505, 54:2539, 54:2568,	<code>\endtrivlist</code>
54:2606, 54:2610, 01:34, 54:2774,	... 37:447, 37:497, 37:499, 37:572,
54:2823, 54:2853, 54:2871, 54:2902,	38:608, 39:100, <u>39:101</u> , 41:85, 43:80
54:2943, 54:2987, 57:15, 57:19,	<code>\endverbatim</code> 37:571, 37:589
57:29, 57:33, 57:51, 57:69, 57:79,	<code>\enlargethispage</code> <u>54:2432</u> , 1071
57:86, 57:94, 57:141, 57:146, 57:198,	<code>\enlargethispage*</code> <u>54:2432</u>
57:222, 57:294, 57:299, 57:367,	<code>\enskip</code> 18:571
57:389, 57:393, 57:489, 57:536,	<code>\enspace</code> <u>18:559</u> , 18:567
57:727, 57:731, 07:1182, 07:1185,	<code>\ensuremath</code> <u>22:345</u> , <u>38:522</u> ,
07:1200, 07:1213, 07:1216, 07:1346,	45:441, 45:447, 45:500, 45:506, 1394
07:1349, 07:1378, 07:1381, 07:1390,	<code>enumerate (env.)</code> 39:262
07:1398, 07:1401, 07:1567, 07:1571,	<code>\enumerate</code> 39:262
02:234, 02:242, 07:2207, 07:2218,	<code>env (hook)</code> 227, 236
07:2250, 07:2266, 07:2277, 07:3191,	<code>env/(env)/after (hook)</code> 240, 240, 241
02:346, 07:3221, 07:3307, 07:3314,	<code>env/(env)/before (hook)</code> 240, 240, 241
02:363, 08:79, 08:95, 08:123, 08:139,	<code>env/(env)/begin (hook)</code> 487, 240, 241
08:151, 08:163, 08:179, 08:192,	<code>env/(env)/end (hook)</code> 240, 241
08:212, 08:222, 08:242, 08:257,	<code>env/document/after (hook)</code> 244
08:275, 08:281, 08:301, 08:319,	<code>env/document/before (hook)</code> 242
08:323, 02:453, 08:600, 08:651,	<code>env/document/begin (hook)</code> 489, 242
08:668, 08:672, 08:695, 08:716,	<code>env/document/end (hook)</code> 244
08:750, 08:808, 08:838, 08:870,	<code>env/quote/after (hook)</code> 234, 234
02:487, 08:897, 08:900, 08:920,	environments:
08:923, 08:936, 08:954, 02:495,	array 41:168
08:959, 08:962, 08:971, 08:976,	center 37:446
08:979, 08:1016, 08:1053, 08:1103,	displaymath 38:389
08:1110, 08:1126, 08:1133, 08:1153,	document 37:8
08:1157, 02:519, 08:1209, 08:1213,	enumerate 39:262
08:1229, 08:1233, 08:1261, 08:1264,	eqnarray <u>38:415</u> , <u>38:609</u>
08:1287, 08:1291, 08:1308, 08:1313,	eqnarray* <u>38:518</u>
08:1320, 08:1324, 08:1360, 08:1384,	equation <u>38:393</u> , <u>38:597</u>
08:1467, 08:1494, 08:1528, 02:559, 26	filecontents 50:1249, 1095
<code>\endinput</code> 1371	flushleft <u>37:496</u>

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- flushright 37:498
- itemize 39:273
- list 39:34
- lrbox 925
- math 38:389
- minipage 926
- picture 42:21
- sloppypar 49:135
- tabbing 41:71
- tabular 41:198
- thebibliography 1053
- trivlist 39:89
- verbatim 37:571
- verbatim* 37:586
- \epsilon 30:283
- eqnarray (env.) 38:415, 38:609
- \eqnarray 38:423, 38:424, 38:451, 38:453, 38:519
- eqnarray* (env.) 38:518
- \eqno 38:400, 38:407, 38:532, 1381
- equation (env.) 38:393, 38:597
- \equation .. 38:397, 38:398, 38:404, 38:406
- \equiv 30:459
- \errhelp 03:31, 14:39, 14:66, 01:201, 56:12, 57:309, 57:785
- \errmessage 03:32, 10:77, 04:63, 14:47, 14:72, 01:7, 24:618, 24:653, 01:206, 26:442, 26:542, 27:65, 05:84, 05:99, 05:140, 56:16, 57:63, 57:311, 02:108, 01:42, 02:174
- \ERROR 08:1740, 08:1741, 07:1883, 07:1893, 07:1902, 07:2300, 07:2311, 07:2331, 07:2348, 07:2420, 07:2443, 191
- \errorcontextlines 14:212, 02:264, 02:443, 02:462, 02:478, 02:493, 02:507, 02:530, 02:547, 1359
- \ERRORmissingcells 55:367
- \ERRORnewtaggingsocket 55:18
- \errorstopmode 57:807, 02:429, 228
- \ERRORusetaggingsocket 55:40, 55:50
- \escapchar 499
- \escapechar 20:473, 24:464, 24:688, 26:230, 28:59, 28:87, 28:159, 28:190, 28:220, 28:251, 28:392, 28:442, 06:166, 06:210, 06:214, 06:222, 06:379, 06:380, 06:404, 06:541, 06:664, 06:682, 52:280, 52:303, 52:326, 52:347, 02:252, 105
- \eta 30:285
- \etatcatcode 04:1041
- \eTeXversion 01:41
- \evensidemargin 54:49, 54:898, 54:968, 54:1027
- \everycr 38:205, 38:208, 38:433, 38:462, 38:625, 02:409, 1362
- \everydisplay 24:419, 24:420, 24:431, 24:443
- \everyeof 57:381
- \everyjob 03:37, 04:215, 04:261, 04:262, 28:414, 28:462, 53:90, 53:91, 57:112, 57:349, 57:350, 57:383, 57:465, 57:755, 57:756, 57:758, 1340
- \everymath . 24:418, 24:420, 24:436, 24:445
- \everypar 438
- \everypar 16:35, 16:41, 16:72, 16:77, 16:122, 16:132, 16:145, 16:183, 20:43, 20:113, 20:171, 24:742, 24:756, 24:803, 37:243, 37:545, 37:567, 39:130, 39:131, 39:133, 39:137, 39:138, 39:211, 39:228, 40:452, 40:473, 41:81, 44:48, 44:96, 44:107, 44:127, 44:136, 45:187, 54:144, 54:171, 54:197, 54:1503, 54:1652, 54:1813, 451
- \EveryShipout 1191
- \ExecuteOptions 26:58, 26:71, 33:841, 50:685
- \exhyphenpenalty 02:192, 02:382
- \exists 30:335
- \exp 38:31
- exp commands:
 - \exp:w ... 07:2656, 08:779, 08:833, 269
 - \exp_after:wN 08:1543, 08:1753, 08:1872, 08:1905, 08:1990, 08:2134, 08:2279, 08:2463, 08:2466, 08:2530, 08:2992, 09:29, 09:31, 09:186, 09:261, 09:315, 09:316, 09:378, 09:386, 09:389, 09:505, 11:376, 11:833, 11:1053, 11:1195, 11:1202, 16:41, 16:72, 28:335, 28:338, 51:46, 52:73, 52:236, 52:237, 07:326, 07:342, 07:395, 07:401, 07:415, 07:1174, 07:1194, 07:1198, 07:1209, 07:1211, 07:1260, 07:1309, 07:1339, 07:1370, 07:1377, 07:1452, 07:1593, 07:1852, 07:1876, 07:1892, 07:1910, 07:1916, 07:1920, 07:1951, 07:1985, 07:1992, 07:2005, 07:2011, 07:2017, 07:2023, 07:2029, 07:2035, 07:2041, 07:2047, 07:2078, 07:2222, 07:2659, 07:2660, 07:2706, 07:2707, 07:2708, 07:2709, 07:2710, 07:2711, 07:2712, 07:2743, 08:50, 08:55, 08:374, 08:375, 08:420, 08:567, 08:778, 08:832, 08:1062, 08:1064, 08:1205, 08:1370, 269
 - \exp_args:Nc 08:2220, 09:45, 09:58, 09:90, 09:148, 48:22, 07:267, 07:1105,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

07:1116, 07:1416, 07:1456, 07:1482,	\exp_not:N 08:1646, 08:1648,
07:1586, 07:2812, 08:1226, 08:1239	09:124, 09:126, 09:127, 09:128,
\exp_args:Ncc 07:142, 07:184	09:200, 09:215, 09:222, 09:223,
\exp_args:NcV 08:591	09:230, 09:235, 09:275, 09:290,
\exp_args:Ne	09:297, 09:298, 09:302, 09:306,
. 08:2368, 08:2369, 09:539,	09:323, 09:372, 09:377, 09:385,
10:139, 11:1167, 36:10, 51:111,	09:409, 09:419, 09:493, 09:494,
51:142, 51:191, 52:46, 52:97, 53:90,	09:514, 09:519, 09:542, 09:569,
07:2673, 07:2676, 07:2699, 07:2732,	09:570, 09:597, 09:598, 10:25,
08:427, 08:475, 08:551, 08:736, 08:738	11:10, 11:15, 11:368, 11:369, 11:370,
\exp_args:Nee 52:129	11:371, 11:381, 11:387, 11:390,
\exp_args:Nf 09:177,	11:392, 11:393, 11:397, 11:399,
09:252, 52:32, 52:34, 52:394, 07:420,	11:402, 11:407, 11:633, 11:793,
07:2424, 07:2447, 07:2506, 07:2734	11:810, 11:811, 11:916, 11:918,
\exp_args:NNc 09:162, 09:167	11:919, 11:921, 11:922, 11:976,
\exp_args:NNe 08:1645,	11:977, 11:978, 11:980, 11:1094,
11:56, 07:1386, 07:1481, 08:1524	11:1102, 11:1142, 11:1157, 11:1182,
\exp_args:Nne 08:1882	11:1189, 16:39, 16:40, 16:70,
\exp_args:NNNo 07:401, 07:1755, 07:2120	16:71, 16:79, 16:80, 36:72, 48:115,
\exp_args:NnnV 11:1259	48:118, 48:120, 48:125, 48:128,
\exp_args:Nnnv 08:251	51:222, 51:223, 51:224, 53:92,
\exp_args:NNo . . . 08:1734, 08:2993,	53:156, 53:186, 07:124, 07:139,
09:185, 09:260, 08:1086, 08:1143	07:140, 07:168, 07:169, 07:201,
\exp_args:NNV	07:203, 07:204, 07:210, 07:212,
. . . . 08:1618, 08:1705, 09:204, 09:279	07:271, 07:283, 07:300, 07:301,
\exp_args:NNx 08:1562	07:302, 07:304, 07:371, 57:653,
\exp_args:No	07:817, 07:818, 07:947, 07:957,
. 08:2093, 08:2152, 08:2982,	07:1038, 07:1051, 07:1075, 07:1093,
08:2993, 09:185, 09:204, 09:260,	07:1100, 07:1138, 07:1146, 07:1147,
09:279, 11:1249, 36:41, 36:46, 07:94,	07:1179, 07:1225, 07:1226, 07:1233,
07:323, 07:412, 07:431, 07:440,	07:1234, 07:1235, 07:1239, 07:1242,
07:1345, 07:2314, 07:2674, 07:2700,	07:1243, 07:1245, 07:1264, 07:1267,
07:2809, 08:1086, 08:1143, 08:1204	07:1268, 07:1269, 07:1277, 07:1324,
\exp_args:Nof 07:2306	07:1343, 07:1357, 07:1364, 07:1365,
\exp_args:Noo 07:1318	07:1465, 07:1466, 07:1638, 07:1645,
\exp_args:NV	07:1661, 07:1665, 07:1667, 07:1697,
. . . 09:205, 09:280, 11:881, 51:222,	07:1710, 07:1714, 07:1715, 07:1719,
52:53, 07:260, 07:364, 07:1761, 07:1764	07:1720, 07:1724, 07:1725, 07:1729,
\exp_args:Nv	07:1730, 07:1790, 07:1796, 07:1797,
. . 08:2022, 08:2030, 08:2132, 08:1271	07:1799, 07:1800, 07:1802, 07:1803,
\exp_args:NVV 11:1626	07:2052, 07:2053, 07:2161, 07:2163,
\exp_args_generate:n 57:646	07:2164, 07:2246, 07:2247, 07:2258,
\exp_end: 07:2659, 08:779, 08:833	07:2261, 07:2274, 07:2735, 07:2738,
\exp_last_unbraced:Ne	07:2739, 07:2740, 07:2741, 07:2743,
. 08:2984, 07:2745,	08:385, 08:567, 08:568, 08:569,
08:1088, 08:1121, 08:1145, 08:1238	08:573, 08:574, 08:1173, 08:1177,
\exp_last_unbraced:Nf	08:1196, 08:1199, 08:1200, 08:1202,
. 08:1929, 09:126, 08:1251	08:1246, 08:1508, 08:1510, 1073
\exp_last_unbraced:NNf 08:1179	\exp_not:n 08:1652, 08:2982, 09:187,
\exp_last_unbraced:NNNo	09:200, 09:203, 09:205, 09:240,
. 08:2087, 09:132, 09:508, 08:377	09:262, 09:275, 09:278, 09:280,
\exp_last_unbraced:NnNo 07:2518	09:308, 09:370, 09:420, 09:422,
\exp_last_unbraced:NNo 09:397	09:423, 09:425, 09:426, 11:373,
\exp_last_unbraced:Nno 26:345	11:520, 11:979, 11:1095, 11:1096,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 11:1103, 11:1141, 11:1143, 11:1144,
 11:1157, 11:1160, 11:1161, 11:1182,
 11:1183, 11:1189, 11:1190, 36:122,
 36:131, 36:261, 48:186, 48:193,
 48:265, 48:266, 48:267, 48:297,
 48:374, 51:207, 52:211, 52:225,
 53:91, 07:163, 07:167, 07:171,
 07:174, 07:178, 07:195, 07:202,
 07:214, 07:284, 07:303, 07:371,
 07:374, 07:375, 07:625, 07:771,
 07:894, 07:905, 07:907, 07:908,
 07:987, 07:1039, 07:1052, 07:1063,
 07:1075, 07:1092, 07:1178, 07:1180,
 07:1233, 07:1246, 07:1278, 07:1313,
 07:1577, 07:1579, 07:1592, 07:1631,
 07:1632, 07:1696, 07:1738, 07:1752,
 07:1753, 07:1760, 07:1777, 07:1861,
 07:1862, 07:1905, 07:1965, 07:2162,
 07:2321, 07:2322, 07:2358, 07:2359,
 07:2542, 07:2558, 07:2573, 07:2585,
 07:2735, 08:108, 08:235, 08:554,
 08:557, 08:1070, 08:1094, 08:1193,
 08:1197, 08:1203, 08:1204, 1080
 \exp_stop_f: 07:1504,
 07:2308, 08:1272, 08:1420, 08:1428
 expand@font@defaults (hook) 244
 expandable commands:
 \expandable_grab_⟨type⟩:w 165
 \expandableinput .. 05:227, 05:236, 05:238
 \expandafter 1385
 expandafter commands:
 \expandafter: 32:89, 41:346
 \ExpandArgs 77
 \ExpandArgs 05:191, 05:207,
 05:210, 06:626, 06:627, 57:650, 117
 \expanded 50:945, 57:111, 57:794, 259
 \ExplLoaderFileDate 05:215, 05:223
 \ExplSyntaxOff
 .. 08:2999, 09:414, 09:643, 10:235,
 11:1638, 16:186, 17:58, 17:137,
 17:161, 17:184, 20:548, 20:768,
 20:818, 24:417, 26:358, 28:320,
 28:363, 05:160, 05:180, 05:203,
 05:232, 36:304, 48:532, 49:32, 49:59,
 49:113, 50:550, 51:287, 52:12, 52:86,
 52:138, 52:243, 52:254, 52:402,
 52:427, 52:472, 52:529, 52:584,
 53:42, 53:62, 53:480, 53:532, 53:539,
 53:548, 53:588, 55:213, 55:303,
 55:319, 55:459, 57:344, 57:346,
 57:348, 57:619, 57:718, 07:3328, 1101
 \ExplSyntaxOn 09:3, 09:415,
 10:2, 11:6, 16:3, 17:2, 17:25, 17:61,
 17:141, 20:537, 20:762, 20:815,
 24:412, 26:343, 28:278, 28:326,
 05:149, 05:175, 05:194, 05:230,
 36:2, 48:3, 49:22, 49:37, 49:110,
 50:548, 51:3, 52:7, 52:29, 52:92,
 52:207, 52:251, 52:391, 52:410,
 52:435, 52:508, 52:577, 53:5, 53:47,
 53:66, 07:8, 53:525, 53:537, 53:546,
 53:583, 55:2, 55:300, 55:305, 55:321,
 57:343, 57:609, 57:620, 08:3, 186
 \extracolsep 41:167
 \extrafloats 02:100, 02:112, 02:160
- ## F
- \fam 04:22, 04:26, 04:38, 24:16, 02:250
 \family 1358
 \familydefault 21:1602, 28:409,
 28:459, 29:523, 29:770, 29:803,
 29:819, 29:835, 29:852, 30:131, 744
 \fbox 925
 \fbox 40:217, 40:230, 40:237, 1375
 \fboxrule 40:215, 40:262, 40:274, 40:279,
 40:285, 40:287, 40:294, 40:295, 57:151
 \fboxsep 40:215, 40:221, 40:254, 40:261,
 40:273, 40:280, 40:290, 40:292, 57:150
 \fi 1358
 fi commands:
 \fi: 08:1756, 08:2135, 08:2255,
 08:2263, 08:2280, 08:2302, 08:2330,
 08:2467, 08:2472, 08:2488, 08:2535,
 08:2570, 08:2585, 09:358, 09:360,
 09:361, 09:365, 09:379, 09:390,
 09:391, 09:394, 16:26, 16:33, 16:65,
 16:105, 48:130, 52:241, 07:905,
 07:908, 07:986, 07:1343, 07:1506,
 07:1844, 07:2656, 07:2660, 07:3292,
 08:791, 08:1065, 08:1222, 08:1245,
 08:1285, 08:1424, 08:1432, 08:1438, 344
 \filbreak 02:386
 file (hook) 227, 236
 file commands:
 \g_file_curr_name_str 08:2829
 \file_full_name:n . 20:544, 52:35, 1169
 \file_get:nnN ... 57:344, 57:346, 1340
 \file_input_raw:n 05:231, 1438
 \file_md5five_hash:n 20:816
 \file_parse_full_name_apply:nN ..
 52:32, 52:47, 52:90, 52:93, 52:95
 \l_file_search_path_seq 1169
 \file_size:n 20:817
 file internal commands:
 __file_parse_full_name_area:nw .
 52:102, 52:105, 52:109
 __file_parse_full_name_auxi:nN .
 52:97, 52:100

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

```

    \__file_parse_full_name_base:nw .
        . . . . . 52:108, 52:112, 52:124
    \__file_parse_full_name_tidy:nnnN
        . . . . . 52:119, 52:120, 52:122, 52:127
file/ (hook) . . . . . 1169
file/.../after (hook) . . . . . 1169
file/.../after . . . . . 1160
file/.../before . . . . . 1160
file/<file name>/after (hook) 1160, 1161
file/<file name>/before (hook) 1160, 1161
file/<file:name>/after (hook) . . . 1160
file/<file:name>/before (hook) . . . 1160
file/<package name>.sty/after (hook)
    . . . . . 1162
file/<package name>.sty/before (hook)
    . . . . . 1162
file/after (hook) . . . . .
    . . . . . 1160, 1160, 1162, 1169, 1182
file/after . . . . . 1160
file/array.sty/after (hook) . . . . 1161
file/before (hook) . . . . .
    . . . . . 315, 1160, 1160, 1162, 1169
file/before . . . . . 1160
filecontents (env.) . . . . . 50:1249, 1095
\filecontents . . . . . 50:1249
filehook internal commands:
    \__filehook_clear_replacement_-
        flag: . . . . . 52:417, 52:420, 52:520
    \__filehook_drop_extension:N . . .
        . . . . . 52:44, 52:49, 52:522
    \__filehook_drop_extension_-
        aux:nnn . . . . . 52:54, 52:57
    \__filehook_file_name_compose:nnn
        . . . . . 52:218, 52:406, 52:414, 52:415, 52:425
    \__filehook_file_parse_full_-
        name:nN 52:28, 52:30, 52:30, 52:53,
        . . . . . 52:238, 52:394, 52:412, 52:414, 1166
    \__filehook_file_pop: . . . . .
        . . . . . 52:59, 52:70, 52:526
    \__filehook_file_pop_assign:nnnn
        . . . . . 52:59, 52:73, 52:79, 52:528
    \__filehook_file_push: . . . . .
        . . . . . 52:59, 52:62, 52:524
    \__filehook_file_replaced . . . . 52:417
    \__filehook_file_subst_begin:nnn
        . . . . . 52:412, 52:422, 52:422, 1176
    \__filehook_file_subst_cycle_-
        error:NN . . . . . 52:457,
        . . . . . 52:462, 52:462, 52:467, 52:469, 1178
    \__filehook_file_subst_loop:NN . .
        . . . . . 52:431, 52:444, 52:452, 52:461, 1177
    \__filehook_file_subst_tortoise_-
        hare:nn . . . . . 52:424,
        . . . . . 52:431, 52:434, 52:436, 52:459, 1178
    \__filehook_full_name:nn . . . . .
        . . . . . 52:30, 52:34, 52:38
    \__filehook_if_file_replaced:TF .
        . . . . . 52:417, 52:418, 52:518, 1177
    \__filehook_if_no_extension:nTF .
        . . . . . 52:44, 52:44, 52:510
    \g__filehook_input_file_seq . . .
        . . . . . 52:59, 1167
    \l__filehook_internal_tl . . . . 52:59
    \__filehook_log_file_record:n . .
        . . . . . 52:547, 52:547, 52:561, 52:563
    \g__filehook_nesting_level_int . .
        . . . . . 52:544, 52:549, 52:552, 52:553, 52:559
    \__filehook_normalize_file_-
        name:w . . . . . 52:406, 52:413, 52:516
    \__filehook_resolve_file_subst:w
        . . . . . 52:406, 52:409, 52:411, 52:514
    \__filehook_set_curr_file:nNN . .
        . . . . . 52:387, 52:390, 52:392, 52:512
    \__filehook_set_curr_file_-
        assign:nnnnN 52:387, 52:395, 52:397
    \__filehook_subst_add:nn . . . . .
        . . . . . 52:206, 52:208, 52:208, 52:252, 1171
    \__filehook_subst_empty_name_-
        chk:NN . . . . . 52:208, 52:236, 52:240
    \__filehook_subst_file_normalize:Nn
        . . . . . 52:208, 52:216, 52:218, 52:230, 52:234
    \__filehook_subst_remove:n . . . .
        . . . . . 52:208, 52:222, 52:253
\filename@parse . . . . . 6, 1
\filesize . . . . . 05:77, 05:123
\fill . . . . . 18:552
\finalhyphendemerits . . . . .
    . . . . . 37:455, 37:459, 37:465, 02:205, 1416
\finalstrut . . . . . 1384
\FirstMark . . . . . 48:402, 48:522, 1061
\firstmark . . . . .
    . . . . . 49:120, 54:1002, 54:1061, 54:2921, 1058
flag internal commands:
    \flag__filehook_file_replaced 1177
flag commands:
    \flag_clear:n . . . . . 52:421
    \flag_if_raised:nTF . . . . .
        . . . . . 36:109, 36:223, 52:419, 52:440
    \flag_new:n . . . . . 36:88, 52:417
    \flag_raise:n . . . . . 36:110, 52:441
\flat . . . . . 30:339
float/begin (tag socket) . . . . . 55:260
float/end (tag socket) . . . . . 55:260
float/hmode/begin (tag socket) . . . 55:258
float/hmode/end (tag socket) . . . . 55:258
\floatingpenalty . . . . .
    . . . . . 45:531, 45:550, 45:568, 02:202
\floatpagefraction . . . . . 45:278, 54:2999

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- floats:footnotes (plug) 54:697, 1234
floats:space:footnotes (plug) 54:681, 1233
\floatsep . . . 54:1077, 54:1095, 54:1102,
54:2730, 54:2780, 54:2830, 54:3004
\flushbottom 49:128, 1248
flushleft (env.) 37:496
\flushleft 37:496
flushright (env.) 37:498
\flushright 37:498
\fmtname 03:1, 03:41, 03:43, 03:46, 03:48,
03:51, 03:60, 50:784, 50:788, 1363
\fmtversion
. . 03:1, 03:19, 03:41, 03:43, 03:46,
03:48, 03:51, 03:60, 03:105, 03:118,
03:166, 14:2, 04:275, 21:1566,
29:318, 41:1, 42:1, 50:204, 50:212,
50:801, 50:804, 57:739, 57:765, 1363
\fnsymbol 556
\fnsymbol 22:308
\font 21:317, 21:318,
21:319, 21:460, 21:467, 21:823,
21:830, 21:890, 21:1027, 21:1028,
21:1088, 21:1193, 21:1195, 21:1197,
21:1244, 24:82, 24:88, 24:90, 26:85,
29:636, 29:669, 29:675, 29:729, 31:8,
31:9, 31:10, 32:86, 33:6, 33:576,
33:590, 33:597, 37:566, 06:934,
06:937, 06:949, 06:952, 02:414, 02:419
\fontdimen 21:317, 21:318, 21:319, 21:460,
21:467, 21:823, 21:830, 29:636,
29:669, 29:675, 32:86, 33:6, 33:576,
33:590, 33:597, 40:353, 40:649,
40:650, 40:651, 42:137, 42:140,
42:689, 45:428, 45:429, 45:431,
45:487, 45:488, 45:490, 02:414, 02:419
\fontencding 04:258,
04:259, 04:281, 04:282, 21:891,
21:1601, 24:314, 24:349, 24:361,
24:371, 28:408, 28:458, 29:769,
29:802, 29:818, 29:834, 30:16, 30:24,
30:79, 30:86, 30:95, 30:97, 33:36, 1375
\fontfamily 24:342, 29:7,
29:10, 29:13, 29:168, 29:178, 29:239,
29:249, 29:561, 29:564, 29:567,
29:862, 29:894, 30:70, 30:81, 30:88,
30:99, 33:28, 33:30, 33:32, 33:34,
33:49, 33:51, 33:53, 33:55, 33:877, 751
\FontFamilyToCheck
. 33:1584, 33:1585, 33:1598
\fontname 21:1028, 24:90
\fontseries 24:342, 25:2781,
25:2782, 25:2793, 25:2795, 25:2805,
25:2807, 29:16, 29:19, 29:337,
29:338, 29:339, 29:340, 29:360,
29:361, 29:362, 29:363, 29:398,
29:399, 29:400, 29:401, 29:418,
29:419, 29:420, 29:421, 29:434,
29:435, 29:436, 29:437, 29:445,
29:446, 29:447, 29:448, 29:461,
29:464, 29:555, 29:558, 29:896, 1414
\fontseriesforce
. 25:2786, 25:2797, 25:2808, 656
\fontshape 21:470, 21:833,
24:342, 25:2916, 25:2921, 25:2926,
25:3134, 25:3135, 25:3144, 25:3146,
25:3151, 25:3153, 25:3204, 25:3208,
25:3211, 25:3214, 25:3217, 25:3220,
25:3223, 25:3226, 25:3233, 29:22,
29:25, 29:28, 29:31, 29:897, 33:600, 764
\fontshapeforce
. 25:3138, 25:3147, 25:3154, 25:3234
\fontsize 19:6, 21:322, 21:348,
21:380, 21:892, 21:1246, 21:1282,
24:80, 24:381, 29:752, 29:898, 33:68,
33:636, 33:894, 45:413, 45:441,
45:447, 45:472, 45:500, 45:506, 1376
\fontsubfuzz 26:458,
26:492, 37:55, 37:120, 37:159, 1372
\F00 81
foo (hook) 313
foo (socket) 356, 356
\foo 338
\footins 45:390, 45:527,
45:546, 45:564, 54:319, 54:320,
54:321, 54:322, 54:380, 54:427,
54:612, 54:617, 54:623, 54:650,
54:738, 54:744, 54:748, 54:820, 1247
\footnote 45:514, 1359
\footnotemark 44:10, 45:580, 1359
\footnoterule
. 40:529, 45:394, 54:621, 54:747, 1389
footnotes:floats (plug) 54:702, 1234
footnotes:floats:legacy (plug)
. 54:707, 1234
footnotes:space:floats (plug) 54:674, 1233
\footnotesep 40:553,
40:570, 40:587, 45:513, 45:530,
45:539, 45:549, 45:557, 45:567, 45:575
\footnotesize 40:545,
40:563, 40:580, 45:528, 45:547, 45:565
\footnotetext 44:12, 45:597, 1359
\footref 45:609, 1420
\footskip
. 54:53, 54:929, 54:992, 54:1051, 1244
\forall 30:334
fp commands:
\fp_eval:n 05:176
\l_tmpa_fp 373

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=lt page.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \fpeval 76
 - \fpeval 05:172, 05:184, 05:186, 76
 - \frac 38:412, 1365
 - \frame 40:201, 40:302
 - \framebox 925
 - \framebox 40:224, 1380
 - \frenchspacing
 - ... 20:46, 20:116, 20:174, 37:571,
 - 37:588, 37:739, 06:974, 06:995, 02:306
 - \frown 30:462
 - \fussy 49:137
 - \futurelet 18:479, 18:487, 05:54,
 - 32:84, 38:256, 41:436, 06:874, 06:888
- G**
- \g__hook_\meta_\hook}_code_prop ... 248
 - \g__hook_\meta_\hook}_declared_tl . 248
 - \g__hook_\meta_\hook}_parameter_tl 248
 - \g__hook_\meta_\hook}_reversed_tl . 248
 - \Gamma 30:308
 - \gamma 30:281
 - \gathered 1322
 - \gcd 38:33
 - \gdef 1103
 - gdef commands:
 - \gdef_ 38:269
 - \ge 30:428, 30:430
 - \Generic* 1383
 - \GenericError 14:18,
 - 14:85, 14:111, 14:164, 26:63, 1379
 - \GenericInfo ... 03:106, 03:108, 03:119,
 - 03:122, 03:127, 03:162, 03:163,
 - 03:168, 03:172, 03:176, 03:197, 14:4,
 - 14:104, 14:130, 14:182, 21:8, 26:32,
 - 26:35, 26:40, 26:76, 05:68, 50:83,
 - 50:91, 50:121, 50:124, 50:1779, 1379
 - \GenericWarning 14:11, 14:94,
 - 14:120, 14:141, 14:149, 14:173,
 - 14:195, 26:43, 26:48, 26:51, 26:79, 1379
 - \geq 30:426, 30:428
 - \getanddefinefonts 1357
 - \GetDocumentProperty 17:35, 458
 - \GetFileInfo 30:3
 - \gets 30:450, 30:452
 - \gg 30:444
 - \global 1397
 - \globaldefs 24:690, 26:232, 28:61, 28:90,
 - 28:161, 28:192, 28:222, 28:254, 02:247
 - \glossary 1050
 - \glossary 35:104,
 - 44:192, 44:200, 46:23, 46:35, 48:282,
 - 49:40, 49:51, 49:66, 49:74, 49:89,
 - 49:97, 54:906, 54:976, 54:1035, 1436
 - \glossaryentry 46:32
 - \gluestretchorder 54:569
 - \glyphmissingdetails 33:1183, 33:1208, 831
 - \goodbreak 06:975, 06:996, 02:386
 - \grave 30:528
 - \group 1357
 - group commands:
 - \group_align_safe_begin/end: .. 186
 - \group_align_safe_begin:
 - 07:290, 07:302, 07:406, 142
 - \group_align_safe_end:
 - 07:324, 07:429, 07:2108, 139
 - \group_begin: 09:28,
 - 09:217, 09:292, 16:17, 16:23, 16:50,
 - 16:56, 28:298, 28:348, 48:68, 48:250,
 - 48:373, 51:4, 52:210, 52:224, 53:153,
 - 07:1612, 07:1621, 07:1657, 07:1706,
 - 07:1745, 07:1809, 07:1816, 07:2100,
 - 07:2488, 07:2493, 07:2665, 07:2669,
 - 07:2697, 08:381, 08:1172, 08:1185
 - \c_group_begin_token 07:691, 07:702,
 - 07:1680, 07:2106, 07:2187, 07:2777
 - \group_end: . 09:30, 09:221, 09:296,
 - 16:19, 16:28, 16:52, 16:60, 28:301,
 - 28:351, 48:73, 48:269, 48:386, 51:21,
 - 52:220, 52:232, 53:155, 07:1642,
 - 07:1691, 07:1733, 07:1755, 07:1766,
 - 07:1821, 07:1825, 07:1829, 07:2121,
 - 07:2497, 07:2502, 07:2679, 07:2680,
 - 07:2682, 07:2684, 07:2694, 07:2701,
 - 07:2702, 08:384, 08:1176, 08:1195, 334
 - \c_group_end_token
 - 07:694, 07:1722, 07:2176
 - \group_insert_after:N
 - 28:334, 28:335, 28:338, 28:360, 28:361
 - \guillemetleft 21:563, 21:797, 21:1132, 1411
 - \guillemetright
 - 21:564, 21:798, 21:1148, 1411
 - \guillemotleft ... 21:566, 21:800, 21:1134
 - \guillemotright .. 21:567, 21:801, 21:1150
 - \guilsinglleft 21:568, 21:1211
 - \guilsinglright 21:569, 21:1212
- H**
- \H 14:24, 21:250, 21:406, 21:492,
 - 21:624, 21:632, 21:651, 21:659,
 - 21:777, 21:1274, 21:1413, 21:1414,
 - 21:1441, 21:1442, 33:141, 33:165
 - \halign 38:127, 38:210,
 - 38:433, 38:462, 38:625, 02:409, 1357
 - \hang 1388
 - \hangafter 02:249
 - \hangindent 44:139, 02:280
 - \hat 30:534

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \hbadness 24:745, 24:752, 24:788, 24:807,
53:299, 53:301, 53:351, 53:353, 02:188
- \hbar 30:346
- \hbox 1186
- hbox commands:
 - \hbox:n 53:435
 - \hbox_set:Nn
53:231, 53:278, 53:304, 53:329, 53:358
 - \hbox_set_to_wd:Nnn
53:30, 53:54, 53:302, 53:354
 - \hbox_unpack:N
48:425, 48:457, 53:312, 53:356
 - \hbox_unpack_drop:N 53:241
 - \hbox_to_ to_ 1373
 - \headheight . 54:51, 54:913, 54:981, 54:1040
 - \headsep ... 54:52, 54:927, 54:990, 54:1049
 - \heartsuit 30:344
 - \height . 21:896, 21:1250, 40:31, 40:34, 1364
 - \hfil 909
 - \hfill 1360
 - \hfuzz 24:753,
49:133, 49:134, 49:140, 49:141,
53:298, 53:300, 53:350, 53:352, 02:265
 - \hglue 02:376
 - \hideoutput 02:565
 - \hideskip 02:179, 02:400
 - \hidewidth 21:347,
21:349, 21:378, 21:382, 21:411,
21:412, 21:415, 21:418, 21:499,
21:500, 21:504, 21:507, 21:509,
21:512, 21:524, 21:529, 21:545,
21:784, 21:785, 21:788, 21:791,
21:858, 21:861, 21:1281, 21:1283, 02:400
 - \HiTeXversion 21:1060
 - \hline 41:435, 41:438, 972
 - \hoffset 02:281
 - \holdinginserts 02:209
 - \hom 38:29
- hook commands:
 - \hook_activate_generic:n
08:2785, 08:2871,
08:2890, 08:2903, 08:282, 08:282,
08:284, 08:302, 08:320, 08:322, 230
 - \hook_debug_off:
08:2847, 08:7, 08:13, 232
 - \hook_debug_on: 08:2846, 08:7, 08:8, 232
 - \hook_disable:n 08:2859, 08:2859
 - \hook_disable_generic:n
08:2787, 08:2864, 08:2896, 08:258,
08:258, 08:260, 08:276, 08:279, 230
 - \hook_gclear_next_code:n
08:2232, 08:2232, 08:2817, 231
 - \hook_gput_code:nnn
08:2797, 08:491, 08:491, 08:493,
08:601, 08:603, 08:703, 08:724, 231
 - \hook_gput_code_with_args:nnn ...
08:2799, 08:491, 08:499, 08:650, 231
 - \hook_gput_next_code:nn
08:2154, 08:2154, 08:2156, 08:2185,
08:2187, 08:2808, 08:712, 08:730, 231
 - \hook_gput_next_code_with_
args:nn 08:2170, 08:2190, 08:2810, 231
 - \hook_gremove_code:nn .. 08:2819,
08:980, 08:980, 08:982, 08:1017, 231
 - \hook_gset_rule:nnnn ... 08:2849,
08:2851, 08:2854, 08:1325, 08:1325, 284
 - \hook_if_empty:n
08:2474, 08:2476, 08:2491, 08:2493
 - \hook_if_empty:nTF
08:1894, 08:2000, 08:2474, 08:2855,
08:2856, 08:2857, 53:124, 53:225, 232
 - \hook_if_empty_p:n 08:1949, 08:2049,
08:2474, 53:128, 53:174, 53:429, 232
 - \hook_log:n
08:1840, 08:1840, 08:2845, 232
 - \hook_new:n 08:2762, 17:114,
48:288, 53:214, 53:215, 53:216,
53:217, 53:218, 53:219, 53:220,
08:61, 08:63, 08:82, 08:217, 08:883, 229
 - \hook_new_pair:nn 08:2766,
16:6, 16:7, 08:193, 08:195, 08:215, 229
 - \hook_new_pair_with_args:nn ... 230
 - \hook_new_pair_with_args:nnn ...
08:2774, 08:193,
08:193, 08:197, 08:213, 08:220, 1429
 - \hook_new_reversed:n ... 08:2764,
08:164, 08:166, 08:182, 08:218, 229
 - \hook_new_reversed_with_args:nn .
08:2772, 08:164,
08:164, 08:168, 08:180, 08:191, 1429
 - \hook_new_with_args:nn .. 08:2770,
08:61, 08:61, 08:65, 08:80, 08:94, 1429
 - \g_hook_patch_action_list_tl ...
09:6, 09:107, 09:140
 - \hook_provide:n 08:2859, 08:2866
 - \hook_provide_pair:nn 08:2859, 08:2880
 - \hook_provide_reversed:n
08:2859, 08:2873
 - \hook_show:n
08:1840, 08:1847, 08:1854, 08:2844, 232
 - \hook_use:n
08:2244, 08:2244, 08:2246,
08:2266, 08:2268, 08:2286, 08:2288,
08:2343, 08:2834, 16:22, 16:31,
16:55, 16:63, 16:100, 16:107, 17:134,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 17:158, 48:252, 53:123, 53:126,
53:132, 53:136, 53:184, 08:1487, 230
- \hook_use:nnw 08:2284,
08:2314, 08:2316, 08:2316, 08:2318,
08:2334, 08:2336, 08:2344, 08:2836, 231
- \hook_use_once:n
08:2383, 08:2385, 08:2400, 08:2835, 223
- \hook_use_once:nnw
..... 08:2383, 08:2383, 08:2391,
08:2398, 08:2406, 08:2837, 1429
- hook internal commands:
- \c_hook_ 08:1317, 08:1321
- \g_hook_??_code_prop 08:1314
- \c_hook_??_parameter_tl ... 08:1314
- \g_hook_??_reversed_tl ... 08:1314
- \g_hook_{hook}_code_prop 248
- \g_hook_{hook}_labels_clist ... 252
- \g_hook_{hook}_reversed_tl ... 248
- __hook_print_args:nn 1431
- __hook_activate_generic:n .. 08:282
- __hook_activate_generic:nn
..... 08:2888, 08:285, 08:286, 08:304
- __hook_activate_generic_pair:nn
.. 08:2859, 08:2885, 08:2889, 08:2917
- __hook_activate_generic_
reversed:n 08:2859,
08:2878, 08:2887, 08:2890, 08:2910
- \g_hook_all_seq 08:2990,
08:28, 08:102, 08:131, 08:1448, 08:1475
- __hook_apply_rule->:nnn . 08:1815
- __hook_apply_rule<-:nnn . 08:1815
- __hook_apply_rule<:nnn .. 08:1815
- __hook_apply_rule>:nnn .. 08:1815
- __hook_apply_rule_xE:nnn . 08:1815
- __hook_apply_rule_xW:nnn . 08:1815
- __hook_apply_label_pair:nnn ...
..... 08:1595, 08:1596,
08:1677, 08:1678, 08:1742, 08:1742, 304
- __hook_apply_rule:nnn
..... 08:1752, 08:1758, 08:1758, 295
- __hook_apply_rule:nnnN 296
- __hook_apply_rule->:nnn .. 08:1793
- __hook_apply_rule<-:nnn .. 08:1793
- __hook_apply_rule<:nnn ... 08:1765
- __hook_apply_rule>:nnn ... 08:1765
- __hook_apply_rule_xE:nnn .. 08:1779
- __hook_apply_rule_xW:nnn .. 08:1779
- __hook_braced_cs_parameter:n ...
..... 08:2985,
08:1089, 08:1146, 08:1171, 08:1182,
08:1234, 08:1234, 08:1236, 08:1262
- __hook_braced_hidden_loop:w ...
.. 08:1234, 08:1238, 08:1241, 08:1247
- __hook_braced_parameter:n
..... 08:1646, 08:1648, 09:237,
09:539, 08:1265, 08:1265, 08:1267,
08:1288, 08:1290, 08:1509, 08:1511
- __hook_braced_real_loop:w . 08:1265
- __hook_chk_args_allowed:nn
..... 08:2208, 08:507, 08:652,
08:652, 08:654, 08:669, 08:671, 306
- __hook_clear_next:n
08:2212, 08:2228, 08:2233, 08:2234,
08:2234, 08:2236, 08:2239, 08:2241, 303
- __hook_clist_gput:Nn
08:1551, 08:1555, 08:1619, 08:1706,
08:1740, 08:1741, 08:1517, 08:1519
- __hook_cmd_beginndocument_code: .
09:62, 09:70, 09:75, 09:78, 09:640, 352
- __hook_cmd_if_scanable:Nn .. 09:484
- __hook_cmd_if_scanable:NnTF ...
..... 09:442, 09:461, 09:484, 349
- __hook_cmd_patch_xparse:Nnn ...
..... 09:144, 09:165, 09:165
- __hook_cmd_try_patch:nn
..... 09:68, 09:79, 09:79
- __hook_code_gset:nn ... 08:2445,
08:112, 08:1111, 08:1111, 08:1113,
08:1125, 08:1127, 08:1129, 08:1506
- __hook_code_gset_aux:nnn
..... 08:1074, 08:1111, 08:1114,
08:1116, 08:1118, 08:1119, 08:1132
- __hook_code_gset_auxi:nnnn
..... 08:1054, 08:1075,
08:1100, 08:1102, 08:1109, 08:1140, 276
- __hook_cs_end:w 08:1234,
08:1253, 08:1254, 08:1255, 08:1260
- __hook_cs_gput_right:nnn
08:2213, 08:2416, 08:546, 08:1054,
08:1054, 08:1056, 08:1104, 08:1106
- __hook_cs_gput_right_fast:nnn ..
.. 08:1054, 08:1062, 08:1068, 08:1107
- __hook_cs_gput_right_slow:nnn ..
.. 08:1054, 08:1064, 08:1071, 08:1108
- __hook_cs_if_empty:N
.. 08:1214, 08:1216, 08:1230, 08:1232
- __hook_cs_if_empty:NnTF
..... 08:1919, 08:1926, 08:2209,
08:2211, 08:2482, 08:2483, 08:1214
- __hook_cs_if_empty_p:N 08:1214
- __hook_cs_parameter_count:N ...
..... 08:1234, 08:1239, 08:1249
- __hook_cs_parameter_count:w ...
.. 08:1234, 08:1251, 08:1258, 08:1259
- \l_hook_cur_hook_tl ... 08:1581,
08:1663, 08:1799, 08:1810, 08:29, 297

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

`\__hook_curr_name_pop:`
 08:2825, 08:2931, 08:411, 08:449, 261
`\__hook_curr_name_push:n`
 08:2823, 08:2828, 08:411, 08:425, 261
`\__hook_curr_name_push_aux:n`
 08:411, 08:427, 08:433
`\__hook_currname_or_default:`
 08:2673, 08:2721,
 08:327, 08:337, 08:341, 08:357,
 08:358, 08:358, 08:543, 08:633, 258
`\__hook_debug:n`
 08:1561, 08:1600, 08:1620,
 08:1682, 08:1707, 08:1767, 08:1774,
 08:1781, 08:1789, 08:1795, 08:1806,
 08:2160, 08:2174, 09:19, 09:27,
 09:44, 09:55, 09:64, 09:65, 09:81,
 09:85, 08:7, 08:7, 08:20, 08:536,
 08:626, 08:1447, 08:1456, 08:1474,
 08:1477, 08:1499, 08:1523, 08:1533, 245
`\g_hook_debug_bool`
 08:6, 08:10, 08:15, 08:21
`\__hook_debug_gset:`
 08:7, 08:11, 08:16, 08:18
`\__hook_debug_label_data:N`
 08:1600, 08:1642,
 08:1683, 08:1731, 08:1828, 08:1828
`\__hook_debug_print_rules:n`
 08:2138, 08:2138
`\__hook_declare_deprecated_-`
 generic:NNn . 08:778, 08:799, 08:832
`\__hook_declare_deprecated_-`
 generic:NNw . 08:794, 08:800, 08:801
`\__hook_def_cmd:w`
 09:13, 09:14, 09:208,
 09:214, 09:283, 09:289, 09:315, 342
`\g_hook_delayed_patches_prop` ...
 09:17, 09:67, 09:73, 09:74
`\__hook_deprecated_generic_-`
 warn:n 08:1870,
 08:1988, 08:777, 08:784, 08:831,
 08:1009, 08:1045, 08:1336, 08:1367, 269
`\__hook_deprecated_generic_-`
 warn:Nn 08:784
`\__hook_deprecated_generic_-`
 warn:Nw 08:784
`\__hook_deprecated_generic_-`
 warn:w 08:785, 08:786
`\__hook_deprecated_warn:nn`
 08:2861, 08:2868,
 08:2875, 08:2882, 08:2893, 08:2900,
 08:2907, 08:2914, 08:2919, 08:2919
`\__hook_disable:n`
 09:225, 09:535, 08:258, 08:261, 08:262
`\__hook_do_deprecated_generic:Nn`
 08:1871, 08:1989, 08:794, 08:794,
 08:1010, 08:1046, 08:1337, 08:1368
`\__hook_do_deprecated_generic:Nw`
 08:794, 08:795, 08:796
`\__hook_double_hashes:n`
 09:204, 09:279,
 09:341, 09:341, 09:396, 08:558,
 08:1086, 08:1095, 08:1143, 08:1200, 280
`\__hook_double_hashes:w`
 09:341, 09:342,
 09:343, 09:373, 09:396, 09:399, 343
`\__hook_double_hashes_group:n` ...
 09:341, 09:349, 09:395
`\__hook_double_hashes_output:N` ...
 09:341, 09:346, 09:354, 344
`\__hook_double_hashes_output_-`
 aux:N . 09:341, 09:368, 09:375, 09:383
`\__hook_double_hashes_space:w` ...
 09:341, 09:350, 09:398
`\__hook_double_hashes_stop:w` ...
 09:341, 09:357, 09:394
`\c_hook_empty_tl` 08:35, 08:1306
`\__hook_end_document_label_-`
 check: 08:411, 08:456, 08:457, 08:464
`\__hook_exp_not:n`
 09:187, 09:188, 09:192, 09:203,
 09:262, 09:263, 09:267, 09:278, 339
`\__hook_exp_not:NN` 09:13,
 09:13, 09:209, 09:215, 09:235,
 09:240, 09:284, 09:290, 09:306, 09:308
`\__hook_file_hook_normalize:n` ...
 08:2369, 08:738,
 08:921, 08:921, 08:924, 08:926, 272
`\l_hook_front_tl` 08:1570,
 08:1611, 08:1614, 08:1617, 08:1619,
 08:1620, 08:1621, 08:1635, 08:1636,
 08:1696, 08:1700, 08:1704, 08:1706,
 08:1708, 08:1710, 08:1724, 08:1725
`\c_hook_generic_{type}/./{place}_tl`
 269
`\c_hook_generic_class/./after_-`
 tl 08:937
`\c_hook_generic_class/./before_-`
 tl 08:937
`\c_hook_generic_cmd/./after_tl` .
 08:937
`\c_hook_generic_cmd/./before_tl`
 08:937
`\c_hook_generic_env/./after_tl` .
 08:937
`\c_hook_generic_env/./before_tl`
 08:937

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

`\c_hook_generic_env/.begin_tl` .
 08:937
`\c_hook_generic_env/.end_tl` 08:937
`\c_hook_generic_file/.after_tl`
 08:937
`\c_hook_generic_file/.before_-`
`tl` 08:937
`\c_hook_generic_include/.after_-`
`tl` 08:937
`\c_hook_generic_include/.before_-`
`tl` 08:937
`\c_hook_generic_include/.end_-`
`tl` 08:937
`\c_hook_generic_package/.after_-`
`tl` 08:937
`\c_hook_generic_package/.before_-`
`tl` 08:937
`\__hook_generic_parameter:n`
 08:1081, 08:1301, 08:1312
`\__hook_generic_parameter:w`
 08:1302, 08:1303
`\c_hook_generics_file_prop`
 08:913, 08:960
`\c_hook_generics_prop`
 08:849, 08:881, 08:937, 08:955, 08:957
`\c_hook_generics_reversed_ii_-`
`prop` 08:858, 08:885, 08:960
`\c_hook_generics_reversed_iii_-`
`prop` 08:862, 08:889, 08:960
`\__hook_gput_code:nnn`
 08:491, 08:496,
 08:502, 08:505, 08:605, 08:606, 08:684
`\__hook_gput_code_store:nnn`
 08:491, 08:518, 08:521
`\__hook_gput_next_code:nn`
 08:2159, 08:2173,
 08:2189, 08:2192, 08:2192, 08:691
`\__hook_gput_next_do:nn` . 08:2198,
 08:2203, 08:2203, 08:2205, 08:2216,
 08:2218, 08:692, 08:713, 08:730, 267
`\__hook_gput_next_do:Nnn`
 08:2220, 08:2223
`\__hook_gput_undeclared_hook:nnn`
 08:673,
 08:673, 08:685, 08:704, 08:724, 267
`\__hook_gremove_code:nn`
 08:980, 08:983,
 08:984, 08:1011, 08:1019, 08:1047
`\__hook_gset_rule:nnnn`
 08:1325, 08:1327, 08:1330, 08:1332,
 08:1337, 08:1361, 08:1363, 08:1368
`\__hook_guess_arg_count:NN` 09:501,
 09:501, 09:503, 09:526, 09:528, 09:534
`\__hook_guess_arg_count:nw`
 09:501, 09:514, 09:519, 09:522
`\__hook_guess_arg_count:wN`
 09:501, 09:505, 09:509
`\__hook_hash_check:nTF`
 08:491, 08:556, 08:565
`\__hook_hash_check_aux:w`
 08:491, 08:567, 08:573
`\c_hook_hash_tl` 09:11,
 09:195, 09:196, 09:198, 09:270,
 09:271, 09:273, 09:360, 08:1246, 1428
`\c_hook_hashes_tl`
 09:11, 09:361, 08:1173, 344
`\g_hook_hook_curr_name_tl`
 08:32, 08:360, 08:370,
 08:411, 08:423, 08:444, 08:445,
 08:452, 08:462, 08:463, 08:489, 321
`\__hook_hook_gput_code_do:nnn` ...
 08:234, 08:251, 08:491,
 08:525, 08:534, 08:613, 08:624, 08:676
`\__hook_if_cmd_hook:n`
 08:2557, 08:2559, 08:2573
`\__hook_if_cmd_hook:nTF`
 08:2100, 08:2557, 08:2575
`\__hook_if_cmd_hook:w` 08:2560, 08:2561
`\__hook_if_cmd_hook:wTF` 08:2557
`\__hook_if_cmd_hook_p:n` 08:2557
`\__hook_if_cmd_hook_p:w` 08:2557
`\__hook_if_declared:n` 08:2522
`\__hook_if_declared:nTF` . 08:2099,
 08:2522, 09:83, 08:69, 08:87, 08:172,
 08:201, 08:204, 08:291, 08:309,
 08:658, 08:883, 08:1059, 08:1077, 250
`\__hook_if_declared_p:n` 08:2522
`\__hook_if_deprecated_generic:n` .
 08:2545
`\__hook_if_deprecated_generic:nTF`
 08:1868,
 08:1986, 08:2537, 08:775, 08:829,
 08:1007, 08:1043, 08:1334, 08:1365
`\__hook_if_deprecated_generic:w` .
 08:2546, 08:2547
`\__hook_if_deprecated_generic_-`
`p:n` 08:2537
`\__hook_if_disabled:n` 08:267
`\__hook_if_disabled:nTF`
 08:1880, 08:1892, 08:1975,
 08:1998, 08:2078, 08:2194, 08:258,
 08:288, 08:306, 08:529, 08:617, 249
`\__hook_if_disabled_p:n` 08:258
`\__hook_if_execute_immediately:n`
 08:2461

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

__hook_if_execute_immediately:nTF
 08:2387, 08:2393, 08:2402,
 08:2461, 08:508, 08:608, 08:1341
 __hook_if_execute_immediately_
 p:n 08:2461
 __hook_if_file_hook:w
 08:898, 08:901, 08:903
 __hook_if_file_hook:wTF
 08:2366, 08:734, 08:898, 1424
 __hook_if_file_hook_p:w 08:898
 __hook_if_generic:n 08:2537
 __hook_if_generic:nTF
 08:1878, 08:1994,
 08:2537, 08:758, 08:816, 08:1080
 __hook_if_generic:w 08:2538, 08:2539
 __hook_if_generic_p:n 08:2537
 __hook_if_generic_reversed:n 08:2577
 __hook_if_generic_reversed:nTF .
 08:2577, 08:296, 08:314, 08:770, 08:824
 __hook_if_generic_reversed:w ...
 08:2578, 08:2579
 __hook_if_generic_reversed_p:n .
 08:2577
 __hook_if_has_hash:n 09:327
 __hook_if_has_hash:nTF
 09:185, 09:260, 09:327, 343
 __hook_if_has_hash:w
 09:327, 09:328, 09:329, 09:335, 09:337
 __hook_if_has_hash_check:w
 09:327, 09:334, 09:339
 __hook_if_has_hash_p:n 09:327
 __hook_if_label_case:nnnnn
 08:1593,
 08:1675, 08:2115, 08:1434, 08:1434
 __hook_if_public_command:N . 09:124
 __hook_if_public_command:NTF ...
 09:93, 09:103, 336
 __hook_if_public_command:w
 09:93, 09:127, 09:133
 __hook_if_replacing_args: . 08:2591
 __hook_if_replacing_args:TF ...
 08:2587, 08:2593, 08:2598,
 08:2599, 08:2604, 08:2605, 08:2610,
 08:510, 08:553, 08:580, 08:656, 08:1093
 __hook_if_reversed:n 08:2528
 __hook_if_reversed:nTF . 08:1548,
 08:1915, 08:1955, 08:1957, 08:2016,
 08:2055, 08:2058, 08:2528, 08:1515
 __hook_if_reversed_p:n 08:2528
 __hook_if_structure_exist:n 08:2516
 __hook_if_structure_exist:nTF ..
 08:2197, 08:2495, 08:2516, 08:2752,
 08:144, 08:156, 08:986, 08:1021, 249
 __hook_if_structure_exist_p:n ..
 08:2516
 __hook_if_usable:n 08:2510
 __hook_if_usable:nTF
 08:1890, 08:1914, 08:1973, 08:1996,
 08:2014, 08:2076, 08:2161, 08:2175,
 08:2419, 08:2510, 08:2858, 08:100,
 08:523, 08:537, 08:582, 08:611,
 08:627, 08:760, 08:818, 08:851,
 08:1003, 08:1039, 08:1502, 08:1537, 249
 __hook_if_usable_p:n
 08:1948, 08:2048, 08:2510
 __hook_if_usable_use:n
 08:2353, 08:2368, 08:2371, 08:2374, 310
 __hook_include_legacy_code_
 chunk:n 08:120,
 08:136, 08:223, 08:223, 08:225,
 08:243, 08:245, 08:1501, 08:1536, 1430
 __hook_init_structure:n 08:2207,
 08:113, 08:133, 08:140, 08:140,
 08:142, 08:152, 08:154, 08:545,
 08:635, 08:675, 08:1347, 08:1372, 253
 __hook_initialize_all:
 08:2929, 08:1442, 08:1442,
 08:1444, 08:1468, 08:1470, 08:1492
 __hook_initialize_hook_code:n ..
 08:2262, 08:2283,
 08:2312, 08:1446, 08:1473, 08:1495,
 08:1495, 08:1497, 08:1529, 08:1531, 282
 __hook_initialize_single:NNn ...
 08:1559, 08:1575, 08:1575, 08:1577,
 08:1655, 08:1657, 08:1659, 08:1521, 288
 \l__hook_label_0_tl 08:1570
 __hook_label_if_exist_apply:nnnTF
 08:1742, 08:1744, 08:1746, 08:1749, 304
 __hook_label_ordered:nn 08:1426, 287
 __hook_label_ordered:nnTF
 08:1389, 08:1396, 08:1403, 08:1426, 285
 __hook_label_ordered_p:nn . 08:1426
 __hook_label_pair:nn
 08:1824, 08:1825,
 08:1388, 08:1395, 08:1402, 08:1407,
 08:1412, 08:1417, 08:1418, 08:1418, 286
 \l__hook_labels_int 08:1570,
 08:1580, 08:1584, 08:1616, 08:1638,
 08:1662, 08:1666, 08:1702, 08:1727, 293
 \l__hook_labels_seq
 08:1570, 08:1579, 08:1585, 08:1603,
 08:1661, 08:1667, 08:1686, 08:1830
 __hook_list_if_rule_exists:nnnTF
 .. 08:2106, 08:2126, 08:2127, 08:2129
 __hook_list_one_rule:nnn
 .. 08:2106, 08:2117, 08:2118, 08:2124

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

```

\__hook_list_rules:nn .. 08:1935,
    08:2035, 08:2106, 08:2106, 08:2143, 304
\__hook_log:nN .. 08:1840, 08:1843,
    08:1850, 08:1857, 08:1864, 08:1866,
    08:1871, 08:1982, 08:1984, 08:1989
\__hook_log_cmd:n .....
    08:1842, 08:1849, 08:1856,
    08:1861, 08:1863, 08:1875, 08:1993
\__hook_log_line:n 08:1840, 08:1860,
    08:1891, 08:1893, 08:1897, 08:1911,
    08:1923, 08:1933, 08:1951, 08:1970,
    08:1997, 08:1999, 08:2003, 08:2011,
    08:2025, 08:2033, 08:2051, 08:2072
\__hook_log_line_indent:n .....
    08:1840, 08:1862,
    08:1901, 08:1907, 08:1917, 08:1924,
    08:1938, 08:1946, 08:2005, 08:2008,
    08:2018, 08:2026, 08:2038, 08:2046
\__hook_log_next_code:n . 08:2030,
    08:2085, 08:2085, 08:2090, 08:2092
\__hook_log_next_code:w .....
    08:1929, 08:2088
\__hook_make_name:n ..... 08:351,
    08:357, 08:366, 08:372, 08:372, 259
\__hook_make_name:w .....
    08:372, 08:374, 08:378
\__hook_make_prefixes:w .....
    09:170, 09:223, 09:298, 09:319, 09:324
\__hook_make_usable:n .....
    08:91, 08:127, 08:312, 08:822, 08:855
\__hook_make_usable:nn .....
    09:227, 09:537,
    08:75, 08:96, 08:96, 08:98, 08:124,
    08:126, 08:294, 08:765, 08:768, 269
\__hook_misused_if_replacing_
    args:nn .. 08:2587, 08:2588, 08:2594
\__hook_msg_pair_found:nnn .....
    08:1767, 08:1774, 08:1781, 08:1789,
    08:1797, 08:1808, 08:1821, 08:1821
\g_hook_name_stack_seq .....
    08:32, 08:412, 08:413, 08:417,
    08:424, 08:444, 08:451, 08:459, 08:469
\__hook_new:n ... 08:83, 08:85, 08:187
\__hook_new:nn ..... 08:61, 08:64,
    08:66, 08:67, 08:84, 08:175, 08:207
\__hook_new_pair:nnn .....
    08:196, 08:198, 08:199
\__hook_new_reversed:n 08:183, 08:185
\__hook_new_reversed:nn .....
    08:164, 08:167,
    08:169, 08:170, 08:184, 08:190, 08:208
\__hook_next_{hook} ..... 274
\__hook_next_gset:nn .....
    08:2212, 08:2237, 08:2446, 08:148,
    08:992, 08:1111, 08:1117, 08:1131
\c__hook_nine_parameters_tl .....
    08:1906,
    08:35, 08:108, 08:1082, 08:1197
\__hook_normalize_code_pool:n ...
    08:114, 08:1158,
    08:1158, 08:1160, 08:1210, 08:1212, 252
\__hook_normalize_cs_args:nn ...
    08:110, 08:111, 08:1134,
    08:1134, 08:1136, 08:1154, 08:1156, 252
\__hook_normalize_fn:nn .....
    08:586, 08:1164, 08:1183, 08:1208, 279
\__hook_normalize_hook_args:Nn ..
    08:1843, 08:1850, 08:1857,
    08:2159, 08:2173, 08:2188, 08:2233,
    08:2388, 08:2394, 08:2403, 08:2888,
    08:64, 08:66, 08:83, 08:167, 08:169,
    08:183, 08:261, 08:285, 08:379, 08:388
\__hook_normalize_hook_args:Nnn .
    08:196, 08:198, 08:379,
    08:393, 08:496, 08:502, 08:604, 08:983
\__hook_normalize_hook_args_
    aux:Nn .....
    08:379, 08:379, 08:390, 08:395, 08:403
\__hook_normalize_hook_rule_
    args:Nnnnn . 08:379, 08:401, 08:1327
\l__hook_param_text_tl .....
    09:8, 09:191,
    09:212, 09:266, 09:287, 09:316, 339
\__hook_parameter:n .....
    08:1885, 08:1122, 08:1180, 08:1292,
    08:1292, 08:1294, 08:1309, 08:1311
\c__hook_parameter_cmd 08:972, 08:977
\c__hook_parameter_cmd/.after_
    tl ..... 08:972
\c__hook_parameter_cmd/.before_
    tl ..... 08:972
\__hook_parse_dot_label:nN .....
    08:328, 08:330, 08:330, 08:430, 08:478
\__hook_parse_dot_label:w .....
    08:330, 08:342, 08:345
\__hook_parse_dot_label_aux:w ...
    08:330, 08:348, 08:356
\__hook_parse_dot_label_cleanup:w
    ..... 08:330, 08:352, 08:355
\__hook_parse_label_default:nN ..
    08:324, 08:324, 08:391,
    08:397, 08:398, 08:405, 08:406, 08:408
\__hook_patch_check:NNnn .....
    09:93, 09:97, 09:100, 09:103, 09:113
\__hook_patch_cmd_or_delay:Nnn ..
    09:32,
    09:45, 09:58, 09:62, 09:62, 09:72, 334

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx


```

\__hook_patch_command:Nnn ..... 09:200, 09:201, 09:206, 09:213,
    ..... 09:72, 09:90, 09:93, 09:93, 336
    09:240, 09:267, 09:274, 09:275,
\__hook_patch_debug:n ..... 09:276, 09:281, 09:288, 09:308, 339
    . 09:18, 09:18, 09:95, 09:96, 09:99,
    09:102, 09:105, 09:174, 09:249,
    09:405, 09:441, 09:444, 09:445,
    09:450, 09:460, 09:463, 09:464, 09:469
\__hook_patch_DeclareRobustCommand:Nnn
    ..... 09:142, 09:146, 09:146, 337
\__hook_patch_DeclareRobustCommand_
aux:Nnn ..... 09:148, 09:151
\__hook_patch_expand_redefine:NNnn
    ..... 09:157,
    09:162, 09:167, 09:170, 09:170,
    09:172, 09:245, 09:247, 09:403, 345
\__hook_patch_newcommand:Nnn ...
    .. 09:143, 09:156, 09:160, 09:160, 337
\l__hook_patch_num_args_int ....
    ..... 09:7, 09:175,
    09:180, 09:183, 09:197, 09:227,
    09:236, 09:250, 09:255, 09:258,
    09:272, 09:534, 09:537, 09:557, 09:564
\l__hook_patch_prefixes_tl .....
    .... 09:8, 09:222, 09:297, 09:314, 342
\__hook_patch_required_catcodes:
    ..... 09:475,
    09:475, 09:496, 09:572, 09:600, 350
\__hook_patch_retokenize:Nnnn ...
    ..... 09:446, 09:465, 09:530,
    09:530, 09:532, 09:577, 09:579, 346
\__hook_post_initialization_
defs: . 08:2339, 08:2339, 08:2341,
    08:2346, 08:2349, 08:2351, 08:1465
\__hook_preamble_hook:n .....
    ..... 08:1874, 08:1992, 08:2244,
    08:2248, 08:2259, 08:2272, 08:2282,
    08:2292, 08:2311, 08:2320, 08:2345,
    08:2378, 08:2413, 08:2432, 08:1488, 307
\__hook_print_args:n ..... 08:2095
\__hook_print_args:nn 08:1882, 08:2095
\__hook_prop_gput_labeled_
cleanup:nnn . 08:491, 08:551, 08:577
\__hook_prop_gput_labeled_do:Nnn
    ..... 08:591, 08:594
\__hook_prop_gput_labeled_
do:Nnnn ..... 08:491
\l__hook_rear_tl .....
    ..... 08:1570, 08:1601, 08:1607,
    08:1608, 08:1631, 08:1632, 08:1684,
    08:1692, 08:1693, 08:1720, 08:1721
\__hook_redefine_with_hooks:Nnnn
    .. 09:170, 09:230, 09:302, 09:312, 341
\l__hook_replace_text_tl .....
    ..... 09:8, 09:192, 09:199,
    09:200, 09:201, 09:206, 09:213,
    09:240, 09:267, 09:274, 09:275,
    09:276, 09:281, 09:288, 09:308, 339
\__hook_replacement_spec:N .....
    ..... 08:1218, 08:1224
\__hook_replacing_args_false: ...
    08:2158, 08:2415, 08:2587, 08:2601,
    08:231, 08:495, 08:663, 08:1451, 1430
\__hook_replacing_args_reset: ...
    08:2168, 08:2182, 08:2418, 08:2587,
    08:2607, 08:237, 08:497, 08:503, 08:1454
\__hook_replacing_args_true: ...
    ..... 08:2172,
    08:2587, 08:2595, 08:501, 08:1452
\g__hook_replacing_stack_seq 08:2587
\__hook_retokenize_patch:Nnn ...
    ..... 09:108, 09:400, 09:400
\l__hook_return_tl .....
    ..... 08:1617, 08:1618, 08:1704,
    08:1705, 08:2609, 08:2610, 08:25,
    08:451, 08:452, 08:459, 08:463,
    08:579, 08:587, 08:592, 08:596,
    08:597, 08:642, 08:645, 08:999, 08:1034
\__hook_rollback_tidying: .. 08:2969
\__hook_rule_<_gset:nnn .... 08:1385
\__hook_rule_>_gset:nnn .... 08:1385
\__hook_rule_after_gset:nnn ....
    ..... 08:1385, 08:1392, 08:1398
\__hook_rule_before_gset:nnn ...
    ..... 08:1385, 08:1385, 08:1391, 292
\__hook_rule_gclear:nnn .....
    08:1348, 08:1373, 08:1415, 08:1416, 286
\__hook_rule_incompatible-error_
gset:nnn ..... 08:1405
\__hook_rule_incompatible-warning_
gset:nnn ..... 08:1405
\__hook_rule_unrelated_gset:nnn .
    ..... 08:1415, 08:1415, 286
\__hook_rule_voids_gset:nnn ....
    ..... 08:1399, 08:1399
\__hook_seq_csname:n ... 08:1568,
    08:1569, 08:1587, 08:1621, 08:1669,
    08:1710, 08:1770, 08:1777, 08:1835
\__hook_set_default_hook_label:n
    ..... 08:2821, 08:467, 08:467
\__hook_set_default_label:n ....
    ..... 08:467, 08:475, 08:482
\__hook_set_normalize_fn:nn ....
    08:584, 08:1158, 08:1162, 08:1167, 279
\__hook_str_compare:nn .....
    08:23, 08:23, 08:1420, 08:1428, 08:1437
\__hook_strip_double_slash:n ...
    ..... 08:921, 08:927, 08:928

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

```

\__hook_strip_double_slash:w ...
..... 08:921, 08:929, 08:930, 08:934
\__hook_tl_csname:n .....
08:1568, 08:1568, 08:1574, 08:1586,
08:1602, 08:1605, 08:1607, 08:1611,
08:1623, 08:1625, 08:1628, 08:1631,
08:1636, 08:1668, 08:1685, 08:1689,
08:1691, 08:1696, 08:1712, 08:1714,
08:1717, 08:1719, 08:1725, 08:1768,
08:1769, 08:1775, 08:1776, 08:1834
\__hook_tl_gclear:N .....
08:1612, 08:1697, 08:2455, 08:2456,
08:2457, 08:58, 08:58, 08:60, 08:238,
08:253, 08:1026, 08:1027, 08:1031
\__hook_tl_gput:Nn 08:1549, 08:1553,
08:1618, 08:1645, 08:1705, 08:1734,
08:1740, 08:1740, 08:1516, 08:1518, 293
\__hook_tl_gput_left:Nn .....
..... 08:1550, 08:52, 08:52, 08:1516
\__hook_tl_gput_right:Nn 08:1554,
08:1647, 08:1736, 08:1738, 08:2229,
08:49, 08:49, 08:51, 08:636, 08:1518
\__hook_tl_gset:Nn .....
..... 08:1541, 08:2228, 08:2442,
08:2980, 08:43, 08:43, 08:45, 08:47,
08:48, 08:50, 08:54, 08:1387,
08:1394, 08:1401, 08:1406, 08:1411
\__hook_tl_gset_eq:NN .....
..... 08:57, 08:57, 08:59
\__hook_tl_set:Nn ..... 08:1586,
08:1668, 08:41, 08:41, 08:1173, 247
\__hook_tmp:w .....
..... 08:1905, 08:1908, 08:2110,
08:2132, 08:2141, 08:2152, 08:2978,
08:2995, 08:2996, 09:189, 09:195,
09:196, 09:198, 09:264, 09:270,
09:271, 09:273, 09:487, 09:490,
09:494, 09:546, 09:549, 09:570,
09:582, 09:585, 09:598, 08:34, 08:34,
08:414, 08:418, 08:420, 08:1169,
08:1180, 08:1196, 08:1202, 08:1205, 350
\l__hook_tmpa_bool ..... 08:1934,
08:1937, 08:1945, 08:1954, 08:2034,
08:2037, 08:2045, 08:2054, 08:24, 300
\l__hook_tmpa_tl .. 09:182, 09:184,
09:186, 09:257, 09:259, 09:261,
09:406, 09:422, 09:425, 09:493,
09:496, 09:538, 09:542, 09:558,
09:565, 09:569, 09:572, 09:597,
09:600, 08:25, 08:424, 08:1181, 08:1203
\l__hook_tmppb_tl .. 09:411, 09:423,
09:426, 08:25, 08:1170, 08:1177, 08:1205
\__hook_toplevel_l{hook} ..... 274
\__hook_toplevel_gset:nn .....
..... 08:2447, 08:147, 08:991,
08:996, 08:1111, 08:1115, 08:1130
\__hook_try_declaring_generic_-
hook:nnn ..... 08:531,
08:619, 08:678, 08:679, 08:681,
08:697, 08:699, 08:718, 08:721, 267
\__hook_try_declaring_generic_-
hook:nNNnn .....
.. 08:723, 08:729, 08:732, 08:732, 267
\__hook_try_declaring_generic_-
hook:wn .. 08:752, 08:755, 08:810,
08:813, 08:840, 08:843, 08:872, 08:875
\__hook_try_declaring_generic_-
hook:wnTF 08:683, 08:690, 08:701,
08:710, 08:745, 08:751, 08:804, 269
\__hook_try_declaring_generic_-
hook_split:nNNnn .....
..... 08:732, 08:737, 08:740, 08:743
\__hook_try_declaring_generic_-
next_hook:nn ..... 08:2199,
08:678, 08:688, 08:708, 08:727, 267
\__hook_try_file_hook:n .....
..... 08:2353, 08:2361, 08:2364, 310
\__hook_try_patch_with_catcodes:Nnnnw
... 09:419, 09:436, 09:436, 09:438,
09:451, 09:455, 09:457, 09:470, 345
\__hook_try_put_cmd_hook:n .....
. 09:20, 09:20, 09:22, 09:37, 09:39,
09:48, 09:50, 08:764, 08:821, 08:854
\__hook_try_put_cmd_hook:w 09:20,
09:23, 09:24, 09:40, 09:41, 09:51, 09:52
\__hook_unpatchable_cases:n .....
..... 09:625, 09:627
\__hook_update_hook_code:n .....
..... 08:2210, 08:2226, 09:242,
09:574, 08:76, 08:297, 08:315,
08:526, 08:614, 08:1004, 08:1040,
08:1352, 08:1377, 08:1441, 08:1441,
08:1446, 08:1453, 08:1472, 08:1476, 274
\__hook_use:wn .. 08:2295, 08:2309,
08:2353, 08:2353, 08:2356, 08:2358, 310
\__hook_use_end: .....
..... 08:2303, 08:2306, 08:2313
\__hook_use_i_delimit_by_s_-
mark:nw ..... 08:39, 08:40, 08:1244
\__hook_use_initialized:n 08:2244,
08:2249, 08:2251, 08:2276, 08:2297,
08:2343, 08:2420, 08:2434, 08:1487, 307
\__hook_use_initialized:nnw .....
.. 08:2316, 08:2321, 08:2323, 08:2344
\__hook_use_none_delimit_by_s_-
mark:w ..... 08:2463, 08:2530,
09:523, 08:39, 08:39, 08:1339, 08:1345
\__hook_use_once:n .. 08:2403, 08:2430

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- _hook_use_once:nn 08:2388, 08:2394, 08:2409, 08:2409, 08:2411, 08:2428, 08:2437
- _hook_use_once_clear:n 08:2417, 08:2435, 08:2439, 08:2439, 08:2443, 08:2451, 08:2453, 312
- _hook_use_once_set:n 08:2414, 08:2433, 08:2439, 08:2441, 312
- _hook_use_undefined:w 08:2301, 08:2305
- \g_hook_used_prop 08:1562, 08:31, 08:1447, 08:1459, 08:1474, 08:1480, 08:1524
- \l_hook_work_prop 08:1557, 08:1582, 08:1589, 08:1591, 08:1600, 08:1617, 08:1642, 08:1664, 08:1671, 08:1673, 08:1683, 08:1703, 08:1731, 08:1802, 08:1813, 08:30, 08:587, 08:1163, 08:1165, 08:1199, 08:1520, 292
- hook_?? internal commands:
 - _hook_?? 283
- hook_⟨hook⟩ internal commands:
 - _hook_⟨hook⟩ 288
- \hookleftarrow 30:491
- \hookrightarrow 30:489
- Hooks:
 - .../after 1169
 - /after 1160, 1160, 1161, 1169
 - /before 1160, 1160, 1169
 - /end 1169
 - A/foo 227, 227
 - after 241
 - babel/⟨language⟩/afterextras ... 236
 - begindocument . 327, 354, 354, 489, 489, 1204, 1208, 239, 239, 239, 242, 250
 - begindocument/before 489, 242
 - begindocument/end 242
 - bfseries 244, 244
 - bfseries/defaults 244, 244
 - build/column/after 54:731, 1232
 - build/column/before 54:731, 1232
 - build/page/after 54:729, 1232
 - build/page/before 54:729, 1232
 - build/page/reset 54:730, 1232
 - class 236
 - class/ 1162, 1169
 - class/⟨name⟩/after 1161
 - class/⟨name⟩/before 1161
 - class/after 1161
 - class/before 1161
 - cmd 315, 330, 331, 230, 236, 236, 269, 274
 - cmd/#1/after 342
 - cmd/#1/before 342
 - cmd/⟨name⟩/after 327, 330, 241
 - cmd/⟨name⟩/before 327, 327, 241
 - cmd/foo/before 329
 - cmd/include/after 1163
 - cmd/include/before 1163
 - cmd/section/after 330, 330
 - cmd/section/before 330
 - enddocument 1206, 238, 239, 243
 - enddocument/afteraux 243
 - enddocument/afterlastpage . 1206, 243
 - enddocument/end 243
 - enddocument/info 1182, 243
 - env 227, 236
 - env/⟨env⟩/after 240, 240, 241
 - env/⟨env⟩/before 240, 240, 241
 - env/⟨env⟩/begin 487, 240, 241
 - env/⟨env⟩/end 240, 241
 - env/document/after 244
 - env/document/before 242
 - env/document/begin 489, 242
 - env/document/end 244
 - env/quote/after 234, 234
 - expand@font@defaults 244
 - file 227, 236
 - file/ 1169
 - file/.../after 1169
 - file/⟨file name⟩/after ... 1160, 1161
 - file/⟨file name⟩/before .. 1160, 1161
 - file/⟨file: name⟩/after 1160
 - file/⟨file: name⟩/before 1160
 - file/⟨package name⟩.sty/after . 1162
 - file/⟨package name⟩.sty/before 1162
 - file/after 1160, 1160, 1162, 1169, 1182
 - file/array.sty/after 1161
 - file/before 315, 1160, 1160, 1162, 1169
 - foo 313
 - include 496, 1163, 236
 - include/ 1169, 1169
 - include/.../end 1169
 - include/⟨name⟩/after 1163
 - include/⟨name⟩/before 1163
 - include/⟨name⟩/end 1163
 - include/⟨name⟩/excluded .. 1163, 1163
 - include/after 1163
 - include/before 1163
 - include/end 1163
 - include/excluded 1163, 1163
 - insertmark 1059, 1078, 245
 - mdseries 244
 - mdseries/defaults 244
 - myhook 221
 - normalfont 244
 - package 236
 - package/ 1162, 1169
 - package/⟨name⟩/after 1161

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- package/<name>/before 1161
- package/<package name>/after .. 1162
- package/<package name>/before . 1162
- package/after 1161, 1162
- package/before 1161, 1162
- para/after 443, 454
- para/before 443, 449
- para/begin 443, 445, 452
- para/end 443, 444, 454
- rmfamily 244, 244, 244
- selectfont 245
- sffamily 244
- shipout .. 1186, 1186, 1187, 1188, 1188
- shipout/... 1184, 1188
- shipout/after .. 1184, 1184, 1185, 1186, 1186, 1187, 1189, 1189, 1198
- shipout/afterpage 1201
- shipout/background 1185, 1185, 1187, 1188, 1190, 1190, 1191, 1195, 1197, 1200, 1200
- shipout/before .. 354, 1184, 1184, 1185, 1185, 1185, 1186, 1186, 1186, 1186, 1186, 1188, 1188, 1188, 1188, 1189, 1189, 1191, 1191, 1196, 1197
- shipout/firstpage 1184, 1185, 1185, 1186, 1187, 1187, 1187, 1197, 1197, 1200, 1200, 1201, 1201, 1212
- shipout/foreground ... 1185, 1186, 1187, 1188, 1190, 1190, 1191, 1197
- shipout/lastpage 1184, 1185, 1186, 1187, 1187, 1187, 1187, 1196, 1198, 1198, 1201, 1206, 1206, 1207, 1207, 243
- tbl/cell/begin 41:469
- tbl/cell/end 41:469
- tbl/cell/init 41:468
- tbl/init 41:465
- tbl/row/begin 41:467
- tbl/row/end 41:467
- tbl/row/init 41:466
- test 237, 237
- top:level 220, 220, 239
- ttfamily 244
- hook ?? internal commands:
 - __hook-?? 08:1314
- \hphantom 38:75
- \hrule .. 18:419, 18:427, 18:453, 18:461, 21:309, 21:314, 30:351, 30:629, 40:207, 40:212, 40:285, 40:295, 40:637, 41:436, 41:453, 42:600, 42:610, 45:395, 02:377, 02:421, 945
- \hrulefill 06:976, 06:997, 02:421
- \hsize 1401
- \hskip 1360
- \hskip... 1397
- \hspace 18:534, 18:538, 18:544, 482
- \hss 1192
- \Hwithstroke 21:517, 21:1241, 529
- \hwithstroke 21:533, 21:1242, 529
- \hyphenation 21:225, 1384
- \hyphenchar 37:566, 06:934, 06:937, 06:946, 06:949, 06:952, 06:957, 1410
- \hyphenpenalty 24:760, 24:792, 02:191
- I**
- \I 50:1378, 50:1510, 50:1600, 50:1620, 57:255, 57:599, 02:312
- \i 21:267, 21:423, 21:473, 21:474, 21:475, 21:476, 21:477, 21:478, 21:479, 21:570, 21:610, 21:611, 21:703, 21:705, 21:707, 21:709, 21:737, 21:802, 21:1168, 21:1323, 21:1325, 21:1327, 21:1329, 21:1380, 21:1383, 21:1386, 21:1389, 21:1459, 57:259, 57:603, 1403
- \ialign 30:348, 30:470, 30:541, 30:544, 30:548, 30:551, 38:168, 38:170, 38:189, 41:243, 41:266, 42:152, 02:409, 02:411, 1362
- identity (plug) 356, 359, 359, 359
- \IeC 57:406, 57:410, 57:517
- \if 871
- if commands:
 - \if:w 08:2464, 08:2478, 08:2531, 08:2563, 09:359, 08:1058, 08:1218, 08:1243, 312
 - \if_case:w 07:1504, 08:1269, 08:1420, 08:1437
 - \if_catcode:w 09:377, 09:385, 07:1840, 07:1841, 07:1842, 07:1843, 344
 - \if_charcode:w 08:2581, 07:986
 - \if_cs_exist:N 48:101
 - \if_cs_exist:w 08:1751, 08:2131, 08:2253, 08:2261, 08:2278, 08:2299, 08:2326, 08:2465, 08:789
 - \if_false: 07:905, 07:908, 07:2656, 07:2660
 - \if_int_compare:w 16:103, 08:1428
 - \if_meaning:w 09:356, 09:360, 09:361, 09:388, 52:241, 07:1343
 - \if_mode... 453
 - \if_mode_horizontal: ... 16:32, 16:64
 - \if... 1150
 - \IfBlankF 07:3302, 120
 - \IfBlankT 07:3302, 120
 - \IfBlankTF 47:19, 07:3302, 120
 - \IfBold 753
 - \IfBooleanF 22:210, 22:225, 22:263, 22:275, 07:3261, 121

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \IfBooleanT 07:3261, 121
- \IfBooleanTF 17:126, 17:150,
35:67, 35:69, 35:207, 07:3261, 121
- \IfClassAtLeastF 50:318
- \IfClassAtLeastT 50:318
- \IfClassAtLeastTF 1097
- \IfClassAtLeastTF
50:200, 50:328, 50:329, 50:357, 50:358
- \IfClassLoadedF 50:318
- \IfClassLoadedT 50:318
- \IfClassLoadedTF 1097
- \IfClassLoadedTF
50:298, 50:324, 50:325, 50:353, 50:354
- \IfClassLoadedWithOptionsF 50:318
- \IfClassLoadedWithOptionsT 50:336, 50:365
- \IfClassLoadedWithOptionsTF 1097
- \IfClassLoadedWithOptionsTF
50:298, 50:318
- \ifcoremisses 33:1186, 33:1377
- \ifcscname 21:11, 21:210, 24:403, 25:2828,
25:2831, 25:3173, 25:3176, 26:129,
28:154, 05:137, 29:43, 29:54, 29:76,
29:87, 33:1212, 06:832, 06:849,
50:1263, 50:1398, 53:486, 54:2480, 1434
- \ifdefined 05:26,
05:76, 05:77, 05:78, 05:79, 05:122,
05:123, 05:124, 29:734, 40:350,
40:644, 45:418, 45:477, 57:318,
57:329, 57:332, 57:356, 57:373,
57:725, 57:730, 02:217, 02:444, 02:508
- \IfDocumentMetadataF
17:7, 17:13, 50:807, 50:813
- \IfDocumentMetadataT 17:6, 17:12
- \IfDocumentMetadataTF .. 17:5, 17:11, 1437
- \IfExplAtLeastF 78
- \IfExplAtLeastF 05:217, 05:225
- \IfExplAtLeastT 78
- \IfExplAtLeastT 05:216, 05:224
- \IfExplAtLeastTF 78
- \IfExplAtLeastTF 05:212,
05:221, 05:223, 05:224, 05:225, 78
- \iff 30:511
- \IfFileAtLeastF 50:318
- \IfFileAtLeastT 50:318
- \IfFileAtLeastTF 1097
- \IfFileAtLeastTF 50:200,
50:330, 50:331, 50:359, 50:360, 1428
- \IfFileExists 485, 1098
- \IfFileExists 20:488,
20:500, 20:636, 20:692, 20:717,
01:162, 05:42, 05:73, 05:119, 50:968,
52:149, 52:185, 52:195, 57:733, 505
- \IfFileLoadedF 50:338
- \IfFileLoadedT 50:338
- \IfFileLoadedTF 50:338, 1431
- \iffontchar
21:890, 21:1088, 21:1193, 21:1195,
21:1197, 21:1244, 33:1182, 33:1183
- \IfFontSeriesContextF ... 29:595, 29:617
- \IfFontSeriesContextT ... 29:594, 29:616
- \IfFontSeriesContextTF
29:575, 29:613, 29:615, 754
- \IfFormatAtLeastF 50:318
- \IfFormatAtLeastT 50:318
- \IfFormatAtLeastTF 1097
- \IfFormatAtLeastTF 50:200,
50:332, 50:333, 50:361, 50:362, 1171
- \IfHookEmptyF 08:2855, 1434
- \IfHookEmptyT 08:2855, 228
- \IfHookEmptyTF
08:2855, 08:2960, 37:351, 37:368, 228
- \IfHookExistsTF ... 08:2858, 08:2959, 323
- \ifincsname .. 18:497, 06:297, 06:318, 1156
- \ifinner 18:261,
18:312, 38:277, 38:285, 38:324,
38:350, 38:376, 45:57, 45:126, 45:315
- \IfInstanceExistsF 11:1610, 376
- \IfInstanceExistsT 11:1610, 376
- \IfInstanceExistsTF 11:1610, 376
- \IfLabelExistsF 36:203, 853
- \IfLabelExistsT 36:203, 1434
- \IfLabelExistsTF ... 36:203, 36:298, 1432
- \IfLabelExistTF 1432
- \IfMarksEqualF 48:408, 1060
- \IfMarksEqualT 48:408, 1060
- \IfMarksEqualTF 48:408, 48:525, 1060
- \ifmmode 1357
- \IfNoValue(TF) 119
- \IfNoValueF 07:3296, 120
- \IfNoValueT 07:3296, 120
- \IfNoValueTF 11:1571, 36:142,
57:684, 57:691, 57:698, 07:3296, 119
- \ifnum 1189
- \ifodd 28:1228,
40:344, 42:331, 42:355, 42:389,
42:411, 45:68, 45:137, 54:17, 54:117,
54:891, 54:965, 54:1023, 54:1333,
54:1336, 54:1369, 54:1372, 54:1484,
54:1487, 54:1633, 54:1636, 54:1790,
54:1793, 54:2151, 54:2154, 54:2272,
54:2275, 54:2395, 54:2660, 54:2668
- \IfPackageAtLeastF 50:318
- \IfPackageAtLeastT 50:318
- \IfPackageAtLeastTF 1097
- \IfPackageAtLeastTF
50:200, 50:326, 50:327, 50:355, 50:356
- \IfPackageLoadedF 50:318
- \IfPackageLoadedT 50:318

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\IfPackageLoadedTF</code>	1097	41:380, 41:382, 41:409, 42:47, 42:57,
<code>\IfPackageLoadedTF</code>		42:77, 42:86, 42:120, 42:131, 42:142,
50:298, 50:322, 50:323, 50:351, 50:352		42:154, 42:159, 42:165, 43:57, 43:59,
<code>\IfPackageLoadedWithOptionsF</code> ...	50:318	44:110, 45:17, 45:24, 45:539, 45:557,
<code>\IfPackageLoadedWithOptionsT</code> ...	50:318	45:575, 47:7, 47:9, 53:33, 53:55, 715
<code>\IfPackageLoadedWithOptionsTF</code> ...	1097	<code>\ignorespacesafterend</code>
<code>\IfPackageLoadedWithOptionsTF</code>		37:7
50:298, 50:334, 50:335, 50:363, 50:364		<code>\IJ</code>
<code>\IfPDFManagementActiveF</code>	57:804	21:270,
<code>\IfPDFManagementActiveT</code> ..	53:11, 57:803	21:457, 21:573, 21:1169, 57:720, 1423
<code>\IfPDFManagementActiveTF</code> ...	57:802, 1352	<code>\ij</code>
<code>\IfPropertyExistsF</code>	36:188, 853	21:269,
<code>\IfPropertyExistsT</code>	36:188, 853	21:455, 21:572, 21:1170, 57:720, 1423
<code>\IfPropertyExistsTF</code> ..	36:188, 36:297, 853	<code>\Im</code>
<code>\IfPropertyExistTF</code>	1432	30:325
<code>\IfPropertyRecordedF</code>	36:218, 1434	<code>\imath</code>
<code>\IfPropertyRecordedT</code>	36:218, 1434	30:320
<code>\IfPropertyRecordedTF</code>	36:218, 36:299, 853	<code>\immediate</code>
<code>\ifsafesubencodingfound</code>		1360
.. 33:1185, 33:1199, 33:1204, 33:1373		<code>\in</code>
<code>\IfSocketExistsF</code>	10:200, 10:226	30:440, 30:472
<code>\IfSocketExistsT</code>	10:200, 10:225	in commands:
<code>\IfSocketExistsTF</code> ...	10:200, 10:224, 360	<code>in_callback</code>
<code>\IfSocketPlugAssignedF</code> ..	10:200, 10:232	04:825
<code>\IfSocketPlugAssignedT</code> ..	10:200, 10:231	<code>\in_callback</code>
<code>\IfSocketPlugAssignedTF</code>		44
10:200, 10:230, 360		<code>\include</code>
<code>\IfSocketPlugExistsF</code>	10:200, 10:229	485
<code>\IfSocketPlugExistsT</code>	10:200, 10:228	<code>include (hook)</code>
<code>\IfSocketPlugExistsTF</code>	10:200, 10:227, 360	496, 1163, 236
<code>\IfTargetDateBefore</code>	50:1838	<code>\include</code>
<code>\ifthenelse</code>	843	20:237, 20:285, 20:287,
<code>\IfValue(TF)</code>	120	20:303, 20:305, 20:364, 20:416, 1163
<code>\IfValueF</code>	07:3299, 120	<code>include/ (hook)</code>
<code>\IfValueT</code>	07:3299, 120	1169, 1169
<code>\IfValueTF</code>	07:3299, 120	<code>include/.../after</code>
<code>\ifvbox</code>	54:324,	1163
54:381, 54:428, 54:499, 54:756, 54:821		<code>include/.../before</code>
<code>\ifvoid</code>	1399	1169
<code>\ifx</code>	1415	<code>include/.../end (hook)</code>
<code>\ignoreprimitiverror</code>	1072	1163
<code>\ignorespaces</code>	18:50,	<code>include/.../end</code>
18:146, 18:167, 18:185, 18:197,		1163
18:208, 18:224, 18:237, 18:581,		<code>include/<name>/after (hook)</code>
20:71, 20:139, 20:194, 21:90, 24:355,		1163
24:365, 24:375, 24:428, 28:317,		<code>include/<name>/before (hook)</code>
28:356, 37:341, 37:358, 37:374,		1163
37:383, 37:432, 37:437, 37:443,		<code>include/<name>/end (hook)</code>
38:332, 38:358, 38:384, 38:538,		1163
38:539, 39:55, 39:248, 40:198,		<code>include/<name>/excluded (hook)</code> 1163, 1163
40:553, 40:570, 40:587, 41:57,		<code>include/after (hook)</code>
41:62, 41:68, 41:83, 41:92, 41:105,		1163
41:109, 41:116, 41:123, 41:125,		<code>include/after</code>
41:134, 41:154, 41:314, 41:378,		1163
		<code>include/before (hook)</code>
		1163
		<code>include/before</code>
		1163
		<code>include/end (hook)</code>
		1163
		<code>include/end</code>
		1163
		<code>include/excluded (hook)</code>
		1163, 1163
		<code>\IncludeInRelease</code>
		02:561,
		08:1575, 02:566, 08:1657, 02:576,
		08:1864, 08:1982, 08:2085, 08:2090,
		08:2154, 08:2185, 08:2203, 08:2216,
		08:2234, 08:2239, 08:2244, 08:2266,
		08:2286, 08:2316, 08:2334, 08:2339,
		08:2349, 08:2353, 08:2356, 08:2383,
		08:2398, 08:2409, 08:2428, 08:2439,
		08:2451, 03:71, 08:2474, 08:2491,
		08:2557, 08:2573, 08:2630, 08:2659,
		08:2767, 08:2776, 08:2782, 08:2789,
		08:2794, 08:2801, 08:2805, 08:2812,
		08:2832, 08:2839, 08:2933, 09:20,
		09:37, 09:48, 09:170, 09:245, 03:180,
		03:184, 03:188, 09:436, 03:192,
		09:455, 09:501, 09:526, 09:530,
		09:577, 09:604, 09:607, 09:637,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

10:210, 04:3, 11:1634, 14:138,
 14:158, 14:192, 14:205, 16:12,
 16:45, 16:169, 04:234, 17:59, 17:139,
 17:163, 17:178, 18:5, 18:22, 04:257,
 18:55, 18:66, 18:82, 18:87, 18:99,
 18:105, 18:133, 18:156, 18:175,
 18:189, 18:201, 18:214, 18:229,
 04:280, 18:258, 18:273, 18:289,
 18:308, 18:342, 18:376, 18:402,
 18:435, 18:493, 18:504, 18:511,
 18:517, 18:523, 18:529, 18:537,
 18:543, 18:557, 18:563, 20:10, 20:84,
 20:142, 20:209, 20:222, 20:239,
 20:276, 20:297, 20:312, 20:373,
 20:422, 20:483, 20:493, 20:519,
 20:551, 20:573, 20:591, 20:601,
 20:617, 20:642, 20:647, 20:655,
 20:669, 20:680, 20:696, 20:729,
 21:95, 21:122, 21:166, 21:187,
 21:344, 21:352, 21:373, 21:389,
 22:41, 22:47, 22:54, 22:77, 22:96,
 22:118, 22:134, 22:142, 22:148,
 22:171, 22:197, 22:205, 22:219,
 22:233, 22:249, 22:258, 22:270,
 22:281, 22:296, 22:328, 22:344,
 22:352, 22:370, 23:5, 23:11, 23:19,
 23:26, 24:25, 24:54, 24:206, 24:228,
 24:273, 24:295, 24:347, 24:358,
 24:368, 24:424, 24:439, 24:517,
 24:528, 24:536, 24:568, 24:576,
 24:590, 24:611, 24:647, 24:734,
 24:798, 25:3, 25:1432, 25:2773,
 25:2780, 25:2792, 25:2804, 25:2812,
 25:2894, 25:2910, 25:2931, 25:2946,
 25:3048, 25:3129, 25:3133, 25:3143,
 25:3150, 25:3158, 25:3230, 25:3249,
 25:3255, 25:3259, 26:114, 05:2,
 26:159, 26:162, 05:14, 05:24, 26:558,
 26:567, 27:2, 27:22, 05:63, 28:50,
 28:79, 28:139, 28:181, 28:211,
 28:242, 28:276, 28:323, 28:369,
 28:418, 05:117, 28:470, 28:483,
 28:489, 28:529, 28:577, 28:604,
 28:629, 05:147, 28:862, 28:910,
 05:162, 28:956, 28:965, 28:1149,
 28:1160, 05:173, 05:183, 29:34,
 29:69, 29:102, 05:192, 29:127,
 29:144, 29:160, 05:206, 29:233,
 29:286, 29:297, 29:321, 29:331,
 05:213, 29:380, 05:220, 29:427,
 29:456, 29:474, 05:228, 29:514,
 05:235, 29:548, 29:577, 29:612,
 06:5, 29:626, 06:13, 29:660, 29:673,
 29:732, 29:741, 29:766, 29:798,
 29:814, 29:830, 29:847, 29:864,
 29:870, 30:75, 30:93, 30:112, 30:121,
 06:56, 30:633, 30:645, 06:62, 32:27,
 32:34, 06:74, 06:83, 33:571, 06:136,
 06:149, 35:12, 35:29, 35:46, 35:62,
 35:76, 35:96, 35:110, 01:274, 35:125,
 35:132, 35:164, 35:174, 35:183,
 35:191, 35:198, 35:210, 35:216,
 35:222, 06:259, 06:266, 36:287,
 37:10, 37:78, 06:291, 37:142, 37:184,
 37:189, 37:201, 37:228, 37:247,
 06:307, 37:269, 37:297, 37:314,
 37:336, 37:347, 37:363, 37:377,
 37:385, 37:396, 37:404, 37:417,
 37:423, 37:435, 37:440, 37:450,
 37:469, 37:486, 37:502, 37:513,
 37:527, 37:549, 01:21, 37:576,
 37:592, 37:605, 37:619, 37:631,
 37:637, 37:643, 37:650, 37:668,
 37:681, 37:697, 37:720, 37:730,
 38:79, 38:87, 38:109, 38:118, 38:137,
 38:144, 38:152, 38:157, 38:165,
 38:176, 38:216, 06:376, 38:233,
 01:291, 38:272, 38:280, 38:291,
 38:297, 38:307, 38:336, 38:361,
 38:396, 38:403, 38:422, 38:450,
 38:492, 38:501, 38:534, 38:542,
 38:555, 38:567, 38:579, 38:588,
 06:418, 39:125, 39:135, 39:143,
 39:162, 06:446, 40:4, 40:14, 40:42,
 40:58, 40:71, 40:91, 40:112, 40:122,
 40:136, 40:142, 40:147, 40:152,
 40:161, 40:169, 06:474, 40:225,
 40:233, 40:246, 06:481, 40:266,
 40:304, 40:313, 40:331, 40:357,
 40:367, 40:392, 40:418, 06:498,
 40:441, 40:463, 40:540, 40:558,
 06:514, 40:574, 40:594, 40:600,
 40:616, 40:655, 40:668, 40:676,
 40:706, 40:717, 41:60, 41:65, 41:137,
 41:157, 41:169, 41:177, 41:186,
 41:192, 41:203, 41:211, 41:219,
 41:225, 41:231, 41:255, 41:299,
 41:304, 06:552, 06:564, 42:10,
 42:16, 42:23, 42:28, 42:37, 42:50,
 42:66, 42:80, 42:91, 42:100, 42:111,
 42:123, 42:157, 42:162, 42:171,
 42:183, 42:246, 42:261, 42:322,
 06:594, 42:380, 42:466, 42:474,
 06:605, 42:483, 42:511, 42:541,
 42:568, 42:582, 42:595, 42:606,
 42:618, 42:635, 42:655, 42:662,
 42:696, 06:630, 42:756, 42:814,
 42:830, 06:638, 43:22, 43:31, 43:42,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

43:48, 43:64, 43:71, 44:5, 44:20,	07:1382, 07:1391, 07:1399, 07:1402,
06:650, 06:657, 44:161, 44:168,	07:1569, 02:226, 02:236, 07:2195,
44:174, 44:182, 44:196, 44:211,	07:2208, 07:2239, 07:2251, 07:2267,
44:218, 44:227, 44:250, 45:35,	02:329, 07:3139, 07:3192, 02:347,
45:105, 45:206, 45:232, 45:280,	07:3302, 07:3308, 07:3324, 08:61,
45:294, 45:335, 45:352, 45:405,	08:80, 08:96, 08:124, 08:140, 08:152,
45:438, 45:443, 45:451, 45:458,	08:164, 08:180, 08:193, 08:213,
45:464, 45:497, 45:502, 45:508,	08:223, 08:243, 08:258, 08:276,
45:525, 45:544, 45:561, 45:611,	08:282, 08:302, 08:320, 02:432,
45:621, 47:14, 47:25, 47:61, 47:77,	08:491, 02:455, 08:601, 08:652,
01:26, 06:756, 06:773, 48:517, 49:20,	08:669, 08:678, 08:696, 08:717,
49:35, 49:61, 49:82, 49:108, 49:116,	08:751, 08:809, 08:839, 08:871,
50:18, 50:23, 50:36, 50:51, 06:822,	08:898, 08:901, 02:489, 08:921,
50:68, 50:80, 50:104, 50:112, 50:118,	08:924, 08:937, 08:955, 08:960,
06:828, 50:134, 50:144, 50:161,	08:963, 08:972, 02:496, 08:977,
50:179, 50:202, 50:210, 50:222,	08:980, 08:1017, 08:1054, 08:1104,
50:235, 50:261, 50:280, 50:300,	08:1111, 08:1127, 08:1134, 08:1154,
50:308, 50:320, 50:348, 50:380,	08:1158, 02:521, 08:1210, 08:1214,
50:396, 06:858, 50:410, 50:428,	08:1230, 08:1234, 08:1262, 08:1265,
50:441, 50:457, 50:477, 50:500,	08:1288, 08:1292, 08:1309, 08:1317,
50:516, 50:545, 50:559, 50:591,	08:1321, 08:1330, 08:1361, 08:1442,
50:609, 50:629, 50:647, 50:666,	08:1468, 08:1495, 08:1529, <i>1421</i>
50:676, 50:686, 50:698, 50:730,	<code>\includeonly</code> <i>485</i>
50:741, 50:755, 50:765, 50:819,	<code>\includeonly</code> <u>20:237</u> ,
50:833, 50:848, 50:878, 50:905,	20:277, 20:279, 20:298, 20:299, <i>1162</i>
50:937, 06:899, 50:1089, 06:918,	<code>\indent</code> 18:560, 39:192, 41:81, <i>454</i>
50:1169, 50:1176, 50:1187, 50:1201,	<code>\IndentBox</code> <u>16:93</u> , 16:173, <i>445</i>
50:1251, 06:929, 50:1386, 06:946,	<code>\index</code> <i>1050</i>
50:1517, 06:957, 06:968, 06:988,	<code>\index</code> 35:103,
52:5, 52:15, 52:27, 52:89, 52:144,	44:191, 44:200, 46:6, <u>46:18</u> , 48:281,
52:181, 52:192, 52:205, 52:249,	49:40, 49:51, 49:66, 49:74, 49:89,
52:257, 52:268, 52:299, 52:322,	49:97, 54:905, 54:975, 54:1034, <i>1022</i>
52:343, 52:357, 52:365, 52:389,	<code>\indexentry</code> 46:15
52:408, 52:433, 52:479, 52:495,	<code>\inf</code> 38:25
52:506, 52:533, 53:3, 53:45, 53:64,	<code>\infy</code> 30:327
53:483, 53:523, 53:535, 53:544,	<code>\initcatcodetable</code> 04:91
53:558, 54:130, 54:158, 54:184,	<code>inline/begin</code> (tag socket) <u>55:84</u>
54:350, 54:371, 54:376, 54:424,	<code>inline/end</code> (tag socket) <u>55:84</u>
54:488, 54:734, 54:842, 54:947,	<code>\input</code> <i>485</i> , <i>1098</i>
54:1006, 54:1145, 54:1163, 54:1224,	<code>\input</code> ... 04:18, <u>20:652</u> , 01:158, 01:161,
54:1245, 54:1281, 54:1305, 54:1417,	26:16, 01:218, 27:106, 05:113,
54:1569, 54:1713, 07:228, 54:1882,	05:135, 06:22, 29:908, 29:918,
07:242, 54:1965, 54:2045, 54:2139,	29:928, 30:10, 30:11, 30:12, 30:13,
54:2261, 54:2475, 54:2498, 54:2512,	30:14, 30:23, 30:42, 30:43, 30:44,
54:2540, 54:2602, 54:2607, 54:2725,	30:48, 30:49, 30:50, 30:51, 30:52,
54:2775, 54:2836, 54:2854, 54:2873,	30:53, 30:58, 30:147, 30:148, 30:149,
54:2903, 54:2944, 55:464, 57:8,	30:150, 30:665, 30:666, 30:667,
57:16, 57:23, 57:30, 57:37, 57:52,	33:1090, 50:715, 57:164, 57:178,
57:71, 57:80, 57:87, 57:106, 57:143,	57:203, 57:283, 57:339, 57:340,
57:166, 57:199, 57:290, 57:295,	57:378, 57:454, 57:738, 01:52, <i>1160</i>
57:316, 57:368, 57:390, 57:397,	<code>\input@path</code> <i>6</i> , <i>1</i>
57:490, 57:723, 57:728, 07:1131,	<code>input@path</code> commands:
07:1183, 07:1186, 07:1201, 07:1214,	<code>\input@path:</code> 01:223
07:1229, 07:1347, 07:1350, 07:1379,	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\inputencodingname</code>	<code>\int_if_zero:nTF</code>
..... 57:431, 57:453, 57:535, 1411	10:33
<code>\InputIfFileExists</code>	<code>\int_incr:N</code>
485, 1098	08:1584, 08:1666,
<code>\InputIfExists</code> 17:9, 20:635, 20:658,	07:363, 07:484, 07:916, 07:997,
20:673, 20:683, 20:693, 20:747,	07:1004, 07:1259, 07:1565, 07:2188, 166
21:1574, 24:488, 29:900, 29:910,	<code>\int_new:N</code> ... 08:1571, 09:7, 11:94,
29:920, 33:806, 33:1173, 50:1054,	17:115, 48:278, 52:544, 07:16, 07:17,
50:1120, 52:143, 56:8, 57:277, 1169	07:18, 07:30, 53:393, 53:395, 07:39,
<code>\inputlineno</code>	55:100, 55:322, 55:323, 55:338, 07:2158
14:214, 01:311, 1371	<code>\int_set:Nn</code> . 09:175, 09:218, 09:250,
<code>\insert</code>	09:293, 09:481, 09:482, 09:512,
45:527, 45:546, 45:564,	11:283, 52:212, 52:226, 07:1137,
54:820, 54:821, 54:2460, 02:129,	07:1148, 07:1356, 07:1366, 07:1490,
02:165, 02:167, 02:170, 02:202, 441	07:2162, 07:2672, 07:2698, 08:1186
<code>\InsertMark</code>	<code>\int_set_eq:NN</code> .. 48:70, 07:792, 07:797
48:399, 48:521, 1059	<code>\int_step_function:nN</code>
<code>insertmark</code> (hook)	07:1439
1059, 1078, 245	<code>\int_step_inline:nn</code>
<code>insertmark</code>	48:367
48:278, 1059	<code>\int_step_inline:nnn</code>
<code>\InstanceValue</code>	 09:183, 09:197, 09:258, 09:272
11:1617, 376	<code>\int_use:N</code> .. 09:236, 09:557, 09:564,
<code>\int</code>	10:63, 10:89, 10:106, 11:1576,
30:359	11:1577, 11:1578, 17:132, 17:156,
int commands:	28:282, 28:283, 28:285, 28:290,
<code>\int_add:Nn</code>	28:330, 28:331, 28:333, 28:342,
07:607, 07:1292	36:256, 36:266, 36:267, 48:105,
<code>\int_case:nnTF</code>	48:118, 48:120, 48:255, 52:553,
55:9, 55:33, 55:43	53:165, 53:177, 07:28, 53:415,
<code>\int_compare:nNnTF</code>	53:423, 55:34, 55:44, 55:371, 55:378,
..... 08:1605, 08:1627, 08:1638,	55:379, 55:431, 55:432, 07:947,
08:1688, 08:1716, 08:1727, 08:2097,	07:1119, 07:1148, 07:1277, 07:1281,
08:2103, 08:2422, 08:2619, 09:180,	07:1282, 07:1366, 07:1556, 1189
09:255, 09:408, 09:521, 10:39, 11:23,	<code>\int_value:w</code>
11:25, 11:27, 11:31, 11:36, 11:121,	53:109, 53:198
28:288, 28:290, 28:342, 48:81,	<code>\int_zero:N</code>
48:283, 53:116, 53:172, 53:203,	08:1580,
53:400, 53:411, 53:419, 55:358,	08:1662, 07:355, 55:113, 07:448,
55:367, 55:388, 55:407, 55:410,	07:449, 07:798, 07:802, 07:951, 07:1251
55:435, 55:442, 55:453, 07:461,	<code>\c_max_int</code>
07:602, 07:902, 07:944, 07:2179,	. 48:70, 53:275, 53:301, 53:326, 53:353
07:2245, 07:2260, 07:2512, 07:2514	<code>\l_tmpa_int</code>
<code>\int_compare:nTF</code>	373
11:284	<code>\c_zero_int</code>
<code>\int_compare_p:nNn</code>	09:180, 09:255, 53:152
..... 08:2966, 08:2967, 07:971	int internal commands:
<code>\int_const:Nn</code>	<code>\g_mark_int</code>
10:37	48:255, 48:278, 48:283, 48:284, 48:285
<code>\int_decr:N</code>	<code>\interdisplaylinepenalty</code>
08:1702, 07:536, 07:582, 07:2178	 18:14, 38:55, 38:207, 38:486
<code>\int_div_truncate:nn</code>	<code>\interfootlinepenalty</code>
07:1441	02:302
<code>\int_eval:n</code>	<code>\interfootnotelinepenalty</code>
08:1624,	. 18:19, 45:529, 45:548, 45:566, 02:302
08:1713, 08:1769, 08:1776, 08:1884,	<code>\interlinepenalty</code>
05:177, 53:402, 07:2526, 08:1270	18:11,
<code>\int_gadd:Nn</code>	24:762, 37:538, 37:541, 37:559,
55:347, 55:448	37:562, 44:67, 44:118, 44:234,
<code>\int_gdecr:N</code>	44:257, 45:529, 45:548, 45:566,
52:559, 55:419	54:343, 54:1505, 54:1509, 54:1654,
<code>\int_gincr:N</code>	54:1658, 54:1815, 54:1819, 02:201, 1359
... 17:131, 17:155, 48:284, 52:549,	<code>\interval</code>
53:148, 53:163, 55:399, 55:416, 07:1115	77
<code>\int_gset:Nn</code>	<code>\interval</code>
... 24:415, 28:295, 28:347, 52:545,	05:172, 05:187, 77
55:360, 55:403, 55:425, 55:426, 55:445	
<code>\int_gzero:N</code>	
48:285,	
55:394, 55:395, 55:400, 55:413, 55:422	
<code>\int_if_odd:nTF</code>	
48:439, 48:499	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltaloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\intertextsep` 29:707, 32:21, 33:598, 43:67, 43:69,
 54:1488, 54:1492, 54:1507, 54:1510,
 54:1517, 54:1637, 54:1641, 54:1656,
 54:1659, 54:1666, 54:1794, 54:1800,
 54:1817, 54:1820, 54:1829, 54:3004
`\intop` 30:358, 30:359
`\iota` 30:287
 iow commands:
 `\iow_char:N`
 08:1799, 08:1810, 08:2642,
 08:2644, 08:2646, 08:2651, 08:2654,
 08:2656, 08:2666, 08:2667, 08:2707,
 08:2715, 08:2727, 08:2732, 08:2737,
 08:2923, 08:2925, 09:615, 16:155,
 16:162, 16:163, 16:164, 16:165,
 07:472, 07:532, 07:563, 07:635,
 07:713, 07:725, 07:756, 07:1158,
 07:1164, 07:2163, 07:3021, 07:3099
 `\iow_log:n` 08:1842
 `\iow_newline:`
 52:557, 07:1409, 07:1429, 07:1438,
 07:1455, 07:1473, 07:1477, 07:1563
 `\iow_now:Nn` 53:414, 53:527
 `\iow_shipout:Nn` 53:527
 `\iow_show:n`
 .. 08:1845, 08:1849, 07:1470, 07:1473
 `\iow_term:n` 08:1620,
 08:1640, 08:1641, 08:1643, 08:1707,
 08:1729, 08:1730, 08:1732, 08:1798,
 08:1809, 08:1823, 08:1829, 08:1830,
 08:1831, 08:1834, 08:1838, 08:1856,
 08:2140, 08:2145, 08:2160, 08:2174,
 09:19, 09:27, 09:44, 09:55, 09:64,
 09:66, 09:82, 09:86, 10:25, 11:109,
 48:20, 48:92, 48:179, 48:185, 48:191,
 48:258, 52:550, 52:583, 07:1477,
 08:536, 08:626, 08:1458, 08:1461,
 08:1479, 08:1482, 08:1500, 08:1534
`\ishortstack` 42:143
`\itdefault` 25:3217, 29:31, 30:106
`\item` ... 14:282, 37:446, 37:496, 37:498,
 37:529, 37:551, 38:562, 38:574,
 38:601, 39:172, 39:250, 41:78, 43:67,
 43:69, 43:74, 43:76, 47:4, 47:8, 440
`\itemindent`
 39:9, 39:42, 39:95, 39:218, 39:239, 909
`\itemitem` 1359
`itemize (env.)` 39:273
`\itemize` 39:273
`\itemsep` 39:1, 39:207, 909
`\iterate` 01:65, 01:66, 02:365
`\itshape` 21:468, 21:831, 25:3215, 25:3216,
 29:29, 29:30, 29:637, 29:670, 29:676,
- J**
- `\J` 57:257, 57:601
`\j` 21:268, 21:424, 21:571,
 21:803, 21:1178, 21:1393, 21:1479
`\jmath` 30:321
`\jobname` 1170
`\Join` 29:881
`\joinrel` 30:482, 30:489,
 30:491, 30:493, 30:495, 30:497,
 30:499, 30:501, 30:503, 30:507, 30:509
`\jot` 38:53, 38:204, 38:497, 38:507
- K**
- `\k` 21:508, 21:613, 21:618,
 21:640, 21:645, 21:721, 21:722,
 21:781, 21:782, 21:836, 21:838,
 21:843, 21:845, 21:1279, 21:1347,
 21:1348, 21:1365, 21:1366, 21:1388,
 21:1389, 21:1390, 21:1443, 21:1444,
 21:1477, 21:1478, 33:145, 33:168, 535
`\kanjiskip` 05:79
`\kappa` 30:288
`\ker` 38:27
`\kern` 1374
`\kern...` 1397
`kernel (para/begin) (tag plug)` ... 55:170
`kernel (para/end) (tag plug)` 55:190
`kernel (para/semantic/begin) (tag
 plug)` 55:126
`kernel (para/semantic/end) (tag plug)`
 55:137
`kernel (para/textblock/begin) (tag
 plug)` 55:147
`kernel (para/textblock/end) (tag plug)`
 55:159
`kernel (recordtarget) (tag plug)` . 55:207
`kernel (refstepcounter/target) (plug)`
 35:136
 kernel internal commands:
 `\__kernel_chk_if_free_cs:N`
 07:3321, 216
 `\__kernel_cmd_if_xparse:NTF`
 09:144, 07:1168, 07:1485,
 07:1487, 07:2730, 07:2751, 07:2807, 171
 `\__kernel_cs_parameter_spec:N` ...
 09:177,
 09:252, 09:402, 05:156, 07:2734, 1432
 `__kernel_cs_parm_from_arg_-
 count:nnTF` 08:103
 `\g__kernel_documentproperties_-
 prop`
 . 17:26, 17:29, 17:37, 17:48, 17:55, 459

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- _kernel_exp_not:w 08:42, 08:44, 08:50, 08:55
 - _kernel_expl_bool 09:413
 - _kernel_file_name_sanitiz:n 52:98
 - _kernel_msg... 1421
 - _kernel_msg... 1423
 - _kernel_quark_new_conditional:Nn 11:104
 - _kernel_quark_new_test:N ... 07:52
 - \g_kernel_target_int 17:115, 17:131, 17:132, 17:155, 17:156
 - \kerneltmpDoNotUse ... 09:474, 09:486, 09:492, 09:497, 09:545, 09:552, 09:573, 09:581, 09:588, 09:601, 348
 - keys commands:
 - \keys_define:nn 20:763, 51:215, 51:219, 51:236, 51:271, 57:621
 - \keys_if_exist:nnTF 51:119, 51:125, 51:149, 51:183, 51:199
 - \l_keys_key_str . 51:48, 51:49, 51:224
 - \l_keys_key_tl 51:48
 - \l_keys_path_str 51:47
 - \keys_set:nn 51:71, 51:286, 57:643
 - \keys_set_known:nnN 11:1257
 - \l_keys_usage_load_prop 51:232
 - \l_keys_usage_preamble_prop . 51:267
 - keys internal commands:
 - \l_keys_class_only_clist 51:53, 51:122, 51:151
 - _keys_find_key_module:wNN .. 51:46
 - \l_keys_forced_global_clist ... 51:49, 51:54, 51:153
 - \l_keys_local_clist 51:56, 51:166, 51:171, 51:175, 51:185, 51:189
 - \l_keys_no_value_bool 51:24
 - _keys_options:n 51:58, 51:58, 51:228
 - _keys_options_aux:n 51:58, 51:59, 51:60
 - _keys_options_class:n 51:101, 51:105, 51:105
 - _keys_options_class:nn 51:105, 51:121, 51:126, 51:133
 - _keys_options_class:nnn 51:105, 51:111, 51:117
 - \l_keys_options_clist 51:55, 51:65, 51:70, 51:136, 51:161, 51:185, 51:201, 1153
 - _keys_options_expand_module:Nn 51:59, 51:208, 51:208, 51:215, 51:219, 51:286
 - _keys_options_expand_module:nN 51:208, 51:212
 - _keys_options_global:n 51:67, 51:96, 51:96
 - _keys_options_loaded:n 51:73, 51:230, 51:230
 - _keys_options_loaded:nn 51:230, 51:239, 51:244
 - \l_keys_options_loading_bool ... 51:57, 51:69, 51:72, 51:246, 1152
 - _keys_options_local: 51:62, 51:164, 51:164
 - _keys_options_local:n 51:68, 51:181, 51:181
 - _keys_options_local:nnn 51:181, 51:191, 51:197
 - _keys_options_package:n 51:102, 51:138, 51:138
 - _keys_options_package:nn 51:138, 51:154, 51:156, 51:159
 - _keys_options_package:nnn ... 51:138, 51:142, 51:147
 - \c_keys_props_root_str 51:9, 51:10, 51:22
 - _keys_remove_equals:n .. 51:112, 51:143, 51:178, 51:192, 51:205, 51:205
 - _keys_remove_equals:w 51:205, 51:206, 51:207
 - _keys_scope:N 51:22, 51:33, 51:35, 51:44
 - _keys_scope:n 51:22, 51:25, 51:26, 51:28
 - _keys_tmp:nn 51:5, 51:11, 51:13
 - \l_keys_tmpa_tl 51:232, 51:234
 - \l_keys_unused_clist 51:75, 51:77, 51:177, 51:186, 51:202
 - keyval commands:
 - \keyval_map_break:n 11:508
 - \keyval_map_inline:nnn 11:498
 - \keyval_parse:NNn 11:306
 - \keyval_parse:nnn 11:451, 11:562, 11:682, 11:768, 11:871
 - \KeyValue 11:162, 11:1196, 11:1459, 11:1616, 372
 - \kill 41:154, 41:162
- L**
- \L 21:262, 21:445, 21:553, 21:795, 21:1171, 50:1375, 50:1507, 50:1597, 50:1619, 57:720
 - \l 21:271, 21:447, 21:574, 21:804, 21:1172, 57:720
 - \label ... 35:94, 44:190, 44:200, 48:280, 49:40, 49:51, 49:66, 49:74, 49:89, 49:97, 54:904, 54:974, 54:1033, 1059
 - label 36:259, 854
 - \labelenumi 922
 - \labelenumiv 922

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\labelformat</code>	841	57:266, 57:585, 57:593, 57:600,
<code>\labelformat</code>	35:184, 35:200,	57:602, 57:603, 57:605, 57:607, 1407
	35:212, 35:217, 35:218, 35:223, 35:225	
<code>\labelitemi</code>	922	<code>\lceil</code>
<code>\labelitemii</code>	922	30:613
<code>\labelitemiii</code>	922	<code>\lcode</code>
<code>\labelitemiv</code>	922	1398
<code>\labelsep</code>	39:9, 39:241,	<code>\ldotp</code>
	39:247, 43:67, 43:69, 43:74, 43:76, 922	30:512, 30:515, 30:630
<code>\labelwidth</code>		<code>\ldots</code>
	39:9, 39:93, 39:240, 39:242, 39:245, 909	21:342, 30:516, 1386
<code>\Lambda</code>	30:311	<code>\le</code>
<code>\lambda</code>	30:289	30:427, 30:429
<code>\land</code>	30:379, 30:381	<code>\leaders</code>
<code>\angle</code>	30:605	30:351, 30:569, 30:570, 30:572,
<code>\language</code>	20:52, 20:122,	30:573, 41:453, 42:575, 42:588,
	37:534, 37:726, 02:35, 54:850,	42:600, 42:610, 44:239, 44:262, 02:421
	54:952, 02:78, 02:80, 56:10, 1410	<code>\leadsto</code>
<code>\lastbox</code>	24:780,	29:884
	38:193, 38:194, 39:132, 39:138,	<code>\leavevmode</code>
	39:216, 44:99, 44:132, 54:310, 448	18:476,
<code>\LastDeclaredEncoding</code>		18:490, 21:93, 21:202, 21:309,
	24:138, 24:141, 57:531, 1404	21:310, 21:414, 21:446, 21:450,
<code>\LastMark</code>	48:402, 48:523, 1064	21:453, 21:502, 21:787, 21:820,
<code>\lastnamedcs</code>	06:850	32:124, 33:62, 33:888, 37:538,
<code>\lastnodetype</code>		37:559, 37:572, 37:583, 37:613,
	17:72, 24:773, 24:774, 24:775, 24:779	37:627, 37:722, 37:732, 37:745,
<code>\lastpenalty</code>	24:776, 32:113, 32:116	38:562, 38:574, 38:601, 39:58,
<code>\lastskip</code>	17:71,	39:103, 40:8, 40:17, 40:24, 40:200,
	17:79, 17:86, 17:94, 17:98, 17:105,	40:202, 40:218, 40:250, 40:269,
	18:45, 18:129, 18:141, 18:164,	40:371, 40:395, 40:421, 40:500,
	18:182, 18:243, 18:244, 18:248,	40:607, 40:629, 40:672, 40:679,
	18:250, 18:251, 18:263, 18:278,	41:206, 41:213, 42:145, 42:325,
	18:295, 18:317, 18:320, 18:350,	42:384, 44:40, 44:235, 44:247,
	18:353, 18:384, 18:387, 18:388,	44:258, 45:410, 45:469, 45:592,
	32:103, 32:106, 39:115, 39:116,	47:34, 54:136, 54:141, 54:163,
	39:181, 39:182, 42:134, 54:568,	54:168, 54:189, 54:194, 02:380,
	02:389, 02:390, 02:392, 02:394, 471	02:407, 02:410, 02:421, 02:423, 1381
<code>\latelua</code>	17:65, 1442	<code>\left</code>
<code>\LaTeX</code>	19:3, 19:15, 06:929, 50:1337,	30:636, 30:638,
	50:1470, 50:1559, 53:450, 53:456, 02:256	30:640, 30:642, 30:647, 30:648,
<code>\LaTeXe</code>	19:13	30:649, 30:650, 38:167, 38:173, 38:195
<code>\latexrelease</code>	1181	<code>\Leftarrow</code>
<code>\LaTeXReleaseInfo</code>	03:36, 03:37,	30:421, 30:503, 30:509
	03:40, 03:45, 03:50, 37:49, 37:114, 36	<code>\leftarrow</code>
<code>\latexreleaseversion</code>	03:1	30:448,
<code>\lbrace</code>	21:328, 30:609	30:450, 30:491, 30:501, 30:507, 30:561
<code>\lbrack</code>	02:316	<code>\leftarrowfill</code>
<code>\lccode</code>	14:19, 14:20,	30:545, 30:561
	14:21, 14:22, 14:23, 14:24, 21:158,	<code>\lefteqn</code>
	21:1093, 37:655, 37:672, 37:686,	38:521
	37:701, 37:744, 57:224, 57:241,	<code>\leftharpoondown</code>
	57:249, 57:256, 57:258, 57:259,	30:464, 30:478
	57:261, 57:263, 57:264, 57:265,	<code>\leftharpoonup</code>
		30:463
		<code>\lefthyphenmin</code>
		56:11, 02:246
		<code>\leftline</code>
		40:725
		<code>\leftmargin</code>
		39:9,
		39:52, 39:53, 39:94, 39:177, 39:179, 907
		<code>\leftmargini</code>
		38:554, 39:17, 909
		<code>\leftmarginii</code>
		39:17
		<code>\leftmarginiii</code>
		39:17
		<code>\leftmarginiv</code>
		39:17
		<code>\leftmarginv</code>
		39:17
		<code>\leftmarginvi</code>
		39:17, 909
		<code>\leftmark</code>
		49:106, 1064
		<code>\Leftrightarrow</code>
		30:420
		<code>\leftrightharpoonup</code>
		30:447
		<code>\leftskip</code>
		24:757,
		37:454, 37:460, 37:464, 37:474,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 37:478, 37:482, 37:531, 37:553,
39:74, 40:455, 40:476, 44:232,
44:237, 44:255, 44:260, 02:291, 02:402
- legacy commands:
- \legacy_if:nTF ... 48:437, 48:459,
48:497, 48:504, 55:174, 55:177, 55:178
 - \legacyoldstylenums 33:4, 33:581
 - \leq 30:425, 30:427
 - \leqno 38:532
 - \let 1377
 - \LetLtxMacro 103
 - \lfloor 30:617
 - \lg 38:4
 - \lggroup 30:619
 - \lhd 29:887
 - \lhook 30:488, 30:489
 - \lim 38:6
 - \liminf 38:8
 - \limits 30:550, 30:554, 38:162, 38:411
 - \limsup 38:7
 - \line . 14:269, 42:169, 42:461, 42:819, 42:836
 - \linebreak 463
 - \linebreak 18:9, 18:27
 - \linepenalty 02:190
 - \lineskip 30:469, 38:200,
40:457, 40:477, 41:71, 41:250,
41:273, 42:147, 42:326, 42:385,
53:284, 53:335, 54:908, 54:977,
54:1036, 02:285, 02:310, 02:375, 02:410
 - \lineskiplimit 30:469,
30:521, 38:202, 38:206, 40:443,
40:458, 40:465, 53:285, 53:336,
54:909, 54:977, 54:1036, 02:271,
02:311, 02:375, 02:412, 02:413, 1410
 - \linespread 24:379
 - \linethickness 42:141, 42:820, 42:837
 - \linewidth 20:30, 20:100, 20:158,
38:315, 38:343, 38:368, 38:563,
38:575, 38:602, 38:606, 38:625, 39:9,
39:51, 39:52, 39:54, 40:453, 40:474,
41:36, 45:266, 54:125, 54:210, 908
 - \LinkTargetOff 17:59
 - \LinkTargetOn 17:59
 - list (env.) 39:34
 - \list 39:34, 39:267, 39:278
 - \listfiles 1098
 - \listfiles 20:769, 1376
 - \listparindent 39:9, 39:41, 39:50, 909
 - \literal 1366
 - \ll 30:445
 - \llap 39:269, 39:280, 40:729
 - \lmoustache 30:574
 - \ln 38:5
 - \lnot 30:337, 30:338
 - \LoadClass 1096
 - \LoadClass
. 50:721, 50:751, 50:1002, 50:1137,
50:1232, 50:1240, 50:1241, 1362
 - \LoadClassWithOptions 1096
 - \LoadClassWithOptions 50:750
 - \LoadFontDefinitionFile
..... 24:210, 24:515, 24:541,
24:542, 30:21, 30:27, 30:28, 30:29, 30:33
 - \LoadPackageWithOptions 1161
 - \loccount 04:17
 - \log 38:3
 - \LogDocumentProperties 17:39, 458
 - \loggingall 02:432
 - \loggingoutput 02:428,
02:445, 02:463, 02:479, 02:493, 1403
 - \LogHook 08:2844, 228
 - \LogSocket 10:192, 10:215, 360
 - \long 1417
 - \Longleftarrow 30:503
 - \longleftarrow 30:500
 - \Longleftrightarrow 30:509, 30:511
 - \longleftrightarrow 30:507
 - \longmapsto 30:505
 - \Longrightarrow 30:497
 - \longrightarrow 30:498, 30:505
 - \loop 04:150, 04:159, 24:771,
41:459, 50:1285, 50:1346, 50:1416,
50:1479, 50:1530, 50:1568, 57:414,
57:425, 57:435, 57:446, 57:476,
57:502, 57:512, 01:65, 02:365, 1384
 - \looseness 02:207
 - \lor 30:380, 30:382
 - \lower 19:2, 30:469, 40:283,
42:46, 42:56, 42:207, 42:316, 42:317,
42:364, 42:365, 42:420, 42:421, 45:473
 - \lowercase
.. 14:26, 21:159, 21:1094, 21:1572,
22:186, 24:395, 24:487, 37:659,
37:676, 37:690, 37:705, 37:745, 1396
 - \lq 02:314
 - lrbox (env.) 925
 - \lrbox 40:188
 - \ltfilehookdate 52:542
 - \ltfilehookversion 52:542
 - lua commands:
 - \lua_now:n 53:92, 53:528
 - \lua_shipout:n 53:528
 - \luabytecode 04:202
 - \luachunk 04:210
 - \luadef 04:182, 04:186, 40
 - \luafunction ... 04:178, 04:182, 04:186, 40
 - luamml commands:
 - \luamml_save:... 1321

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- luatexbase 04:287
 \luatexluafunction 01:21, 01:26
 \luatexversion 04:5, 21:1018,
 01:14, 05:78, 05:124, 02:217, 02:218
- M**
- \M 02:312
 \Macro 1359
 \mag 02:251
 \magstep 02:303
 \magstephalf 02:303
 \makeat... 1378
 \makeatletter
 ... 09:409, 20:32, 20:102, 20:160,
 24:493, 37:29, 37:94, 37:156, 44:151,
 50:715, 50:966, 50:1098, 06:925, 1375
 \makeatother
 . 09:409, 50:715, 06:925, 57:806, 1375
 \makebox 925
 \makebox 38:315, 38:343, 38:368, 40:3, 1322
 \makecol 1250
 \makeglossary 1050
 \makeglossary 20:205, 46:20, 1383
 \makeindex 1050
 \makeindex 20:204, 46:3, 1383
 \makelabel 39:45, 39:97,
 39:236, 39:249, 39:269, 39:280, 908
 \MakeLinkTarget .. 17:59, 43:67, 43:69, 462
 \MakeLowercase 57:620, 1390
 \MakeRobust ... 28:960, 28:1155, 06:375,
 42:816, 42:817, 42:818, 42:819,
 42:820, 42:821, 42:822, 42:823,
 42:824, 42:825, 42:826, 42:827,
 06:970, 06:971, 06:972, 06:973,
 06:974, 06:975, 06:976, 06:977,
 06:978, 06:979, 06:980, 06:981,
 06:982, 06:983, 06:984, 06:985, 1421
 \maketitle 1012
 \MakeTitlecase 57:620, 1426
 \MakeUppercase 35:203,
 35:205, 35:214, 35:220, 57:620, 1390
 \mapsto 30:455
 \mapstochar 30:454, 30:455, 30:505
 \marginpar 45:308, 443
 marginpar/begin (tag socket) 55:235
 marginpar/end (tag socket) 55:235
 \marginparpush 54:61, 54:2411
 \marginparsep 54:60, 54:2422, 54:2424
 \marginparwidth
 45:341, 45:359, 54:59, 54:2424
 \mark 49:46, 49:56, 49:69,
 49:77, 49:92, 49:100, 49:121, 1366
- mark commands:
- \mark_clear_structure:n
 48:224, 48:224, 1067
 \mark_copy_structure:nn
 48:138, 48:213,
 48:213, 48:427, 48:428, 48:429,
 48:430, 48:461, 48:465, 48:466, 1066
 \mark_debug_off:
 48:340, 48:346, 48:357, 1063
 \mark_debug_on:
 48:340, 48:341, 48:356, 1063
 \mark_get_marks_for_reinsertion:nnn
 48:162, 48:162, 1067
 \mark_if_eq:nnnn 48:299
 \mark_if_eq:nnnnnn 48:306
 \mark_if_eq:nnnnnnTF ... 48:299, 1060
 \mark_if_eq:nnnnTF
 . 48:299, 48:409, 48:412, 48:415, 1060
 \mark_insert:nn 48:189,
 48:200, 48:246, 48:246, 48:401,
 49:24, 49:25, 49:26, 49:29, 49:30,
 49:43, 49:44, 49:45, 49:54, 49:55, 1067
 \mark_new_class:n
 48:7, 48:7, 48:16, 48:399, 1059
 \mark_set_structure_to_err:n ...
 48:235, 48:235, 48:462, 1066
 \mark_update_structure_from_-
 material:nn 48:132, 48:132,
 48:420, 48:424, 48:452, 48:456, 1080
 \mark_use_first:nn
 . 48:289, 48:289, 48:403, 49:112, 1060
 \mark_use_last:nn
 . 48:289, 48:290, 48:405, 49:111, 1060
 \mark_use_top:nn
 48:289, 48:291, 48:407, 1060
- mark internal commands:
- __mark_update_dblcol_structures:
 1066
 __mark_class_status:nnn
 48:358, 48:359, 48:391
 __mark_debug:n
 48:20, 48:92, 48:179, 48:185,
 48:191, 48:258, 48:340, 48:340,
 48:353, 48:396, 48:433, 48:493, 1082
 __mark_debug_gset:
 48:340, 48:344, 48:349, 48:351
 __mark_drop_id:n
 48:189, 48:202, 48:295, 48:297, 48:298
 __mark_error:n 48:235
 __mark_error:nn
 48:238, 48:239, 48:240, 48:243, 48:294
 __mark_extract_and_handle_-
 marks:nn
 ... 48:67, 48:67, 48:133, 48:165, 1071

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>__mark_get_from_splitmarks:</code> 48:166, 48:171, 48:171	<code>math/luamml/mtable/aligncol</code> (tag socket) 55:290
<code>__mark_init_region:nn</code> 48:24, 48:25, 48:26, 48:27, 48:28, 48:29, 48:30, 48:31, 48:32, 48:33, 48:34, 48:35, 48:36, 48:37, 48:38, 48:39, 48:40, 48:41, 48:42, 48:43, 48:44, 48:45, 48:46, 48:47, 48:48, 48:49, 48:50, 48:51, 48:52, 48:54, 48:54	<code>math/luamml/mtable/finalize</code> (tag socket) 55:289
<code>__mark_new_class:nn</code> 48:7, 48:14, 48:17	<code>math/luamml/mtable/finalizecol</code> (tag socket) 55:288
<code>__mark_prepare_and_extract:nn</code> 48:72, 48:75, 48:75, 48:96, 1071	<code>math/luamml/mtable/innertable/finalize</code> (tag socket) 55:293
<code>__mark_region_status:nnn</code> 48:361, 48:362, 48:363, 48:364, 48:365, 48:366, 48:369, 48:372, 48:372	<code>math/luamml/mtable/innertable/save</code> (tag socket) 55:291
<code>__mark_status:nn</code> 48:388, 48:388, 48:396, 48:435, 48:495	<code>math/luamml/mtable/smallmatrix/save</code> (tag socket) 55:292
<code>__mark_update_dblcol_structures:</code> 48:449, 48:449, 48:516, 1067	<code>math/luamml/mtable/tag/save</code> (tag socket) 55:294
<code>__mark_update_singlecol_-</code> structures: 48:417, 48:417, 48:514, 1067	<code>math/luamml/mtable/tag/set</code> (tag socket) 55:295
<code>__mark_update_structure_from_-</code> splitmarks:n 48:134, 48:137, 48:137, 1074	<code>math/luamml/save/nn</code> (tag socket) .. 55:281
<code>__mark_use_check:nnn</code> 48:289, 48:290, 48:291, 48:292	<code>math/luamml/save/nNn</code> (tag socket) . 55:282
<code>__mark_value:nn</code> 48:62, 48:238, 48:239, 48:240, 48:255, 48:277, 48:277, 48:374	<code>\mathaccent</code> 1409
<code>__mark_vbox_set_split_to_-</code> maxdimen:NN 48:87, 48:101, 48:102, 48:114, 1071	<code>\mathalpha</code> 28:1047, 28:1226, 30:180, 30:181, 30:182, 30:183, 30:184, 30:185, 30:186, 30:187, 30:188, 30:189, 30:190, 30:191, 30:192, 30:193, 30:194, 30:195, 30:196, 30:197, 30:198, 30:199, 30:200, 30:201, 30:202, 30:203, 30:204, 30:205, 30:206, 30:207, 30:208, 30:209, 30:210, 30:211, 30:212, 30:213, 30:214, 30:215, 30:216, 30:217, 30:218, 30:219, 30:220, 30:221, 30:222, 30:223, 30:224, 30:225, 30:226, 30:227, 30:228, 30:229, 30:230, 30:231, 30:232, 30:233, 30:234, 30:235, 30:236, 30:237, 30:238, 30:239, 30:240, 30:241, 30:308, 30:309, 30:310, 30:311, 30:312, 30:313, 30:314, 30:315, 30:316, 30:317, 30:318, 30:527, 30:528, 30:529, 30:530, 30:531, 30:532, 30:533, 30:534, 30:536, 30:539
<code>\markboth</code> .. 49:21, 49:22, 49:36, 49:38, 49:62, 49:64, 49:83, 49:85, 49:87, 1058	<code>\mathbf</code> 29:15, 29:334, 29:387, 29:432, 29:460, 29:554, 30:162
<code>\markright</code> 49:22, 49:49, 49:72, 49:86, 49:95, 1064	<code>\mathbin</code> 28:1231, 30:243, 30:244, 30:246, 30:369, 30:370, 30:371, 30:372, 30:375, 30:376, 30:377, 30:378, 30:381, 30:382, 30:383, 30:384, 30:385, 30:386, 30:387, 30:388, 30:389, 30:390, 30:391, 30:392, 30:393, 30:394, 30:395, 30:396, 30:397, 30:398, 30:399, 30:400, 30:401, 30:402, 30:403, 30:404, 30:405, 30:406, 30:407, 30:408, 38:37
<code>\marks</code> 04:37, 57:10, 57:12	
<code>math</code> (env.) 38:389	
<code>\math</code> 38:389	
<code>math/luamml/annotate/false</code> (tag socket) 55:283	
<code>math/luamml/array/finalize</code> (tag socket) 55:285	
<code>math/luamml/array/finalizecol</code> (tag socket) 55:287	
<code>math/luamml/array/initcol</code> (tag socket) 55:286	
<code>math/luamml/array/save</code> (tag socket) 55:284	
<code>math/luamml/artifact</code> (tag socket) . 55:299	
<code>math/luamml/finph0nt</code> (tag socket) 55:297, 55:298	
<code>math/luamml/hbox</code> (tag socket) 55:296	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\mathcal</code>	30:161	30:284, 30:285, 30:286, 30:287,
<code>\mathchar</code>	28:988, 28:1032,	30:288, 30:289, 30:290, 30:291,
	30:346, 30:347, 30:628, 02:408, 1409	30:292, 30:293, 30:294, 30:295,
<code>\mathchardef</code>	12:3, 12:4, 12:5,	30:296, 30:297, 30:298, 30:299,
	12:6, 04:226, 21:88, 28:1023, 02:21,	30:300, 30:301, 30:302, 30:303,
	02:22, 02:23, 02:24, 02:83, 02:84, 1382	30:304, 30:305, 30:306, 30:307,
<code>\mathcharzero</code>	04:226	30:319, 30:320, 30:321, 30:322,
<code>\mathchoice</code>	38:61	30:323, 30:324, 30:325, 30:326,
<code>\mathclose</code>	28:1234, 30:242,	30:327, 30:328, 30:329, 30:330,
	30:251, 30:253, 30:256, 30:261,	30:331, 30:332, 30:333, 30:334,
	30:267, 30:269, 30:271, 30:577,	30:335, 30:336, 30:338, 30:339,
	30:604, 30:608, 30:612, 30:616,	30:340, 30:341, 30:342, 30:343,
	30:622, 38:43, 38:46, 38:49, 38:52	30:344, 30:345, 30:535, 30:537,
<code>\mathcode</code> .	28:1020, 30:263, 30:264, 30:265	30:538, 30:560, 30:561, 30:564,
<code>\MathCollectFalse</code>	1310	30:565, 30:566, 30:567, 30:579,
<code>\MathCollectFalse</code>	55:52	30:581, 30:583, 30:586, 30:588,
<code>\MathCollectTrue</code>	1310	30:602, 30:624, 30:625, 30:626, 30:627
<code>\MathCollectTrue</code>	55:52	<code>\mathpalette</code>
<code>\mathdefaultsmode</code>	38:529, 38:530	30:468, 30:472,
<code>\mathdollar</code>	21:327, 30:625, 1382	30:475, 38:60, 38:69, 38:99, 38:129
<code>\mathellipsis</code>	21:341, 30:630, 1382	<code>\mathparagraph</code>
<code>\mathfontset</code>	1357	. 21:330, 22:335, 22:347, 30:625, 1399
<code>\mathgroup</code>	24:15, 26:304, 26:310,	<code>\mathpunct</code>
	26:316, 26:317, 26:328, 30:654, 33:8,	28:1235,
	33:14, 33:578, 33:1100, 02:75, 1357	30:245, 30:249, 30:512, 30:513, 30:514
<code>\mathhexbox</code>	29:763, 02:408, 1389	<code>\mathrel</code>
<code>\mathindent</code>	38:552,	28:1232, 30:248, 30:250,
	38:564, 38:576, 38:604, 38:615, 1415	30:258, 30:259, 30:272, 30:273,
<code>\mathinner</code> .	30:515, 30:519, 30:524, 30:630	30:349, 30:409, 30:410, 30:411,
<code>\mathit</code> 25:3216, 29:30, 30:164, 30:167, 30:628		30:412, 30:413, 30:414, 30:415,
<code>\MathMLarg</code>	1310	30:416, 30:417, 30:418, 30:419,
<code>\MathMLarg</code>	55:300	30:420, 30:421, 30:422, 30:425,
<code>\MathMLintend</code>	1310	30:426, 30:429, 30:430, 30:431,
<code>\MathMLintend</code>	55:300	30:432, 30:433, 30:434, 30:435,
<code>\mathnormal</code>	30:160	30:436, 30:437, 30:438, 30:439,
<code>\mathop</code> 28:1230, 30:352, 30:353, 30:354,		30:440, 30:441, 30:443, 30:444,
	30:355, 30:356, 30:357, 30:358,	30:445, 30:446, 30:447, 30:448,
	30:360, 30:361, 30:362, 30:363,	30:449, 30:452, 30:453, 30:454,
	30:364, 30:365, 30:367, 30:368,	30:456, 30:457, 30:458, 30:459,
	30:548, 30:551, 38:3, 38:4, 38:5,	30:460, 30:461, 30:462, 30:463,
	38:6, 38:7, 38:8, 38:9, 38:10, 38:11,	30:464, 30:465, 30:466, 30:468,
	38:12, 38:13, 38:14, 38:15, 38:16,	30:472, 30:475, 30:482, 30:484,
	38:17, 38:18, 38:19, 38:20, 38:21,	30:487, 30:488, 30:490, 30:493,
	38:22, 38:23, 38:24, 38:25, 38:26,	30:495, 30:590, 30:592, 30:594,
	38:27, 38:28, 38:29, 38:30, 38:31,	30:596, 30:598, 30:600, 38:42,
	38:32, 38:33, 38:34, 38:162, 38:411	38:45, 38:48, 38:51, 38:162, 38:411
<code>\mathopen</code> 28:1233, 30:252, 30:255, 30:260,		<code>\mathring</code>
	30:266, 30:268, 30:270, 30:575,	30:539, 1401
	30:606, 30:610, 30:614, 30:618,	<code>\mathrm</code>
	30:620, 38:41, 38:44, 38:47, 38:50	29:6,
<code>\mathord</code>	28:1047, 28:1229, 30:247,	29:485, 29:528, 29:560, 30:159, 1358
	30:254, 30:257, 30:262, 30:274,	<code>\mathsection</code>
	30:275, 30:276, 30:278, 30:279,	. 21:331, 22:334, 22:346, 30:625, 1382
	30:280, 30:281, 30:282, 30:283,	<code>\mathsf</code>
		29:9,
		29:490, 29:533, 29:563, 30:163, 30:166
		<code>\mathsterling</code>
		21:339, 30:625, 1382
		<code>\mathstrut</code> ...
		38:84, 38:93, 38:171, 38:172
		<code>\mathsurround</code>
		02:275, 02:396, 1399
		<code>\mathsymbol</code>
		28:1025

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\mathtt</code>	29:12, 29:495, 29:538, 29:566, 30:165	<code>\MessageBreak</code>	03:101,
<code>\mathunderscore</code>	30:625, 1399		14:3, 14:6, 14:13, 14:33, 14:46,
<code>\mathversion</code>	24:399, 29:725, 29:727, 1358		14:60, 14:73, 14:220, 14:222, 14:228,
<code>\mathversion.</code>	1357		14:235, 17:19, 21:179, 21:1010,
<code>\matrix</code>	38:169, 38:173, 38:180		21:1011, 21:1577, 21:1579, 21:1581,
<code>\max</code>	38:22		21:1584, 22:128, 22:130, 22:165,
<code>\maxdeadcycles</code>	54:3, 02:248		22:167, 24:35, 24:36, 24:633, 24:667,
<code>\maxdepth</code>	18:324,		25:2844, 26:21, 26:22, 26:68, 26:89,
	18:357, 20:60, 20:129, 20:184, 54:68,		26:328, 26:495, 26:515, 26:547,
	54:148, 54:149, 54:175, 54:176,		26:563, 26:578, 26:591, 27:31, 27:33,
	54:504, 54:593, 54:594, 54:767,		05:83, 05:86, 05:87, 05:88, 05:89,
	54:1075, 54:1342, 57:152, 02:268, 1437		05:90, 05:91, 28:281, 05:104, 28:329,
<code>\maxdimen</code>	24:743, 24:753,		05:105, 05:106, 05:107, 05:108,
	24:789, 24:804, 26:401, 26:454,		28:654, 05:139, 28:663, 05:141,
	30:469, 42:485, 42:513, 42:543,		05:142, 28:801, 29:62, 29:95, 32:145,
	42:621, 42:638, 50:1640, 50:1689,		33:23, 33:43, 33:45, 33:64, 33:812,
	50:1699, 53:398, 53:400, 53:462,		33:814, 33:815, 33:816, 33:818,
	54:296, 54:2430, 54:2450, 54:2455,		33:820, 33:821, 33:822, 33:823,
	54:2841, 54:2909, 54:2910, 54:2912,		33:824, 33:874, 33:876, 33:883,
	57:156, 02:179, 02:269, 02:270,		33:890, 33:1105, 06:245, 37:56,
	02:375, 02:413, 02:429, 02:443,		37:121, 37:160, 06:383, 06:422,
	02:444, 02:462, 02:478, 02:493, 1208		06:450, 50:388, 50:402, 50:777,
<code>\mbox</code>	925		50:788, 50:790, 50:792, 50:803,
<code>\mbox</code>	19:13, 21:313, 21:430, 21:591,		50:810, 50:812, 50:997, 50:998,
	21:1062, 21:1193, 29:759, 30:517,		50:999, 50:1000, 50:1010, 50:1011,
	40:11, 40:20, 40:24, 42:63, 45:441,		50:1013, 50:1014, 50:1015, 50:1017,
	45:447, 45:500, 45:506, 02:408, 1322		50:1019, 50:1105, 50:1106, 50:1108,
<code>mc (tag socket)</code>	55:54		50:1109, 50:1110, 50:1112, 50:1114,
<code>\mddefault</code>	29:19, 29:363,		50:1132, 50:1133, 50:1134, 50:1135,
	29:369, 29:370, 29:371, 29:410,		50:1217, 50:1234, 50:1235, 50:1307,
	29:411, 29:412, 29:421, 29:448,		50:1324, 50:1363, 50:1438, 50:1457,
	29:464, 29:481, 29:522, 29:558,		50:1496, 50:1551, 50:1585, 50:1703,
	30:104, 30:116, 30:118, 30:132, 1423		50:1705, 50:1788, 50:1791, 50:1804,
<code>mdseries (hook)</code>	244		50:1806, 52:488, 53:160, 53:236,
<code>\mdseries</code>	29:17, 29:18,		53:423, 53:551, 53:552, 53:553,
	29:314, 29:356, 29:407, 29:408,		53:554, 54:829, 54:2571, 54:2618,
	29:442, 29:443, 29:462, 29:463,		57:302, 57:303, 57:304, 57:306, 1379
	29:556, 29:557, 29:762, 32:20, 753		
<code>mdseries</code>	29:499	<code>\mho</code>	29:880
<code>mdseries/defaults (hook)</code>	244	<code>\mid</code>	30:413
<code>mdseries/defaults</code>	29:499	<code>\min</code>	38:23
<code>\meaning</code>	01:203,	<code>minipage (env.)</code>	926
	01:212, 28:699, 28:712, 28:813,	<code>\minipage</code>	40:487
	28:878, 28:925, 28:989, 28:1083,	<code>\mit</code>	29:933
	28:1179, 28:1283, 06:332, 06:394,	<code>\mkern</code>	30:346,
	06:432, 06:460, 01:307, 06:543, 06:896		30:349, 30:351, 30:473, 30:482,
<code>\medbreak</code>	06:977, 06:998, 02:390		30:524, 30:525, 30:526, 30:556,
<code>\mediumseries</code>	745		30:557, 30:558, 30:559, 30:560,
<code>\medmuskip</code>	30:656,		30:561, 30:562, 30:563, 38:36, 38:37,
	38:36, 38:38, 38:224, 38:227, 38:241		38:40, 38:73, 38:74, 44:240, 44:263
<code>\medskip</code>	18:469, 02:393	<code>mlist commands:</code>	
<code>\medskipamount</code>	18:470, 18:472, 02:392	<code>mlist_to_hlist</code>	04:1023
<code>\medspace</code>	38:214	<code>mode commands:</code>	
		<code>\mode_if_horizontal:TF</code>	16:97, 16:102
		<code>\mode_if_inner:TF</code>	16:98

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\mode_if_math:TF</code>	28:360	<code>\msg_expandable_error:nnn</code>	
<code>\mode_if_vertical:TF</code>		... 08:2590, 05:199, 36:77, 36:113,	
.....	16:89, 16:116, 16:128	36:136, 07:419, 07:3274, 08:364, 08:1283	
<code>\models</code>	30:495, 1403	<code>\msg_expandable_error:nnnn</code>	
module commands:			52:464, 07:2423, 07:2446
<code>module_error</code>	04:345	<code>\msg_info:nnnn</code>	
<code>module_info</code>	04:345		11:192, 11:286, 11:552,
<code>module_warning</code>	04:345	11:950, 11:973, 11:991, 07:73,	
<code>\module_error</code>	43	07:84, 07:234, 07:235, 07:251, 07:255	
<code>\module_info</code>	43	<code>\msg_line_context:</code>	
<code>\module_warning</code>	43	08:2672, 08:2677, 08:2681, 08:2728,	
<code>modules</code>	04:298	11:1075, 11:1233, 11:1365, 11:1371,	
<code>\month</code>	03:11, 03:17, 01:169,	11:1379, 11:1386, 11:1393, 11:1400,	
50:1341, 50:1474, 50:1563, 02:260		11:1407, 11:1414, 11:1422, 11:1528,	
<code>\moveright</code>	54:912, 54:980, 54:1039	11:1530, 11:1534, 11:1537, 48:20,	
<code>\mp</code>	30:400	48:94, 48:180, 48:259, 07:3066,	
<code>\mscount</code>	41:456	07:3071, 07:3076, 07:3081, 08:585	
msg commands:		<code>\msg_log:nnn</code>	17:53
<code>\msg...</code>	1421	<code>\msg_module_name:n</code>	
<code>\msg_error:nn</code>	10:49, 11:358,		51:94, 51:253, 51:259, 51:263
16:124, 16:134, 36:14, 08:436, 08:453		<code>\g_msg_module_name_prop</code>	08:2614
<code>\msg_error:nnn</code> 08:2195, 10:30, 10:96,		<code>\g_msg_module_type_prop</code> .	08:2612,
10:115, 10:136, 10:151, 11:138,		08:2613, 10:191, 11:1548, 48:313, 07:58	
11:145, 11:150, 11:278, 11:338,		<code>\msg_new:nnn</code>	08:2670,
11:382, 11:404, 11:461, 11:473,		08:2675, 08:2679, 08:2724, 08:2730,	
11:486, 11:507, 11:577, 11:590,		08:2735, 08:2740, 08:2745, 08:2749,	
11:608, 11:635, 11:647, 11:692,		08:2756, 08:2921, 11:1527, 11:1529,	
11:782, 11:804, 11:872, 11:907,		11:1531, 11:1536, 11:1538, 11:1543,	
11:936, 11:1044, 11:1205, 11:1244,		17:39, 05:201, 51:251, 52:470,	
48:11, 48:273, 51:274, 52:75, 07:349,		07:1156, 07:3020, 07:3063, 07:3068,	
07:755, 07:3161, 07:3167, 07:3178,		07:3073, 07:3078, 07:3083, 07:3095	
07:3197, 07:3200, 07:3208, 08:70,		<code>\msg_new:nnnn</code>	
08:88, 08:173, 08:202, 08:205,			08:2615, 08:2625, 08:2632,
08:461, 08:530, 08:548, 08:618, 08:638		08:2639, 08:2648, 08:2662, 08:2684,	
<code>\msg_error:nnnn</code>	09:59,	08:2700, 08:2713, 09:609, 09:619,	
09:117, 09:431, 10:83, 10:112,		10:168, 10:173, 10:177, 10:181,	
10:133, 11:132, 11:158, 11:292,		10:187, 11:1316, 11:1324, 11:1329,	
11:710, 11:738, 11:748, 11:995,		11:1335, 11:1344, 11:1350, 11:1357,	
11:1007, 11:1118, 11:1222, 11:1228,		11:1364, 11:1370, 11:1376, 11:1385,	
11:1314, 16:20, 16:33, 16:53, 16:65,		11:1392, 11:1399, 11:1406, 11:1413,	
16:110, 36:35, 48:244, 51:38, 51:79,		11:1421, 11:1428, 11:1437, 11:1443,	
51:248, 07:463, 07:471, 07:495,		11:1449, 11:1455, 11:1462, 11:1468,	
07:499, 07:531, 07:562, 07:577,		11:1474, 11:1481, 11:1487, 11:1493,	
07:667, 07:684, 07:712, 07:724,		11:1499, 11:1515, 11:1521, 16:146,	
07:743, 07:782, 07:1155, 07:2065,		16:157, 36:268, 36:274, 36:280,	
07:2686, 07:2690, 07:3109, 07:3123,		48:314, 48:322, 48:329, 51:90,	
07:3228, 07:3242, 08:106, 08:512, 08:662		51:256, 51:279, 07:2844, 07:2851,	
<code>\msg_error:nnnnn</code> ..	11:124, 11:571,	07:2858, 07:2865, 07:2872, 07:2878,	
11:1585, 07:674, 07:699, 07:2225,		07:2884, 07:2891, 07:2898, 07:2905,	
07:2232, 07:2525, 08:440, 08:471, 08:486		07:2912, 07:2919, 07:2927, 07:2934,	
<code>\msg_error:nnnnnn</code>		07:2941, 07:2948, 07:2956, 07:2964,	
..	08:1782, 08:1343, 08:1355, 08:1380	07:2973, 07:2980, 07:2987, 07:2994,	
<code>\msg_expandable_error:nn</code>		07:3001, 07:3007, 07:3013, 07:3022,	
.....	08:335, 08:336	07:3028, 07:3035, 07:3042, 07:3053	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\msg_redirect_module:nnn</code>	07:57	<code>\negthickspace</code>	38:214, 1418
<code>\msg_show:nnn</code>	17:46	<code>\negthinspace</code>	18:566, 38:214, 1415
<code>\msg_show:nnnnn</code>	11:1306	<code>\neq</code>	30:423, 779
<code>\msg_show:nnnnnn</code>	11:1295	new commands:	
<code>\msg_show_item:nn</code>	17:48, 17:55	<code>new_attribute</code>	04:411
<code>\msg_show_item_unbraced:nn</code>	11:1299, 11:1311	<code>new_bytecode</code>	04:445
<code>\msg_warning:nnn</code>	07:81, 08:289, 08:307	<code>new_chunkname</code>	04:458
<code>\msg_warning:nnnn</code>	08:2920,	<code>new_luacommand</code>	04:491
51:247, 07:736, 07:1630, 07:1751,		<code>new_luafunction</code>	04:474
08:1000, 08:1013, 08:1035, 08:1049		<code>new_whatsit</code>	04:433
<code>\msg_warning:nnnnn</code>	11:1593, 08:790	<code>\New...</code>	117
<code>\msg_warning:nnnnnn</code>	08:1790	<code>\new_attribute</code>	41
msg internal commands:		<code>\new_bytecode</code>	41
<code>\c__msg_coding_error_text_tl</code>	48:317, 48:325, 48:332	<code>\new_chunkname</code>	41
<code>\c__msg_return_text_tl</code>	48:320, 48:337	<code>\new_luacommand</code>	42
<code>\mskip</code>	38:36, 38:38,	<code>\new_luafunction</code>	41
38:219, 38:238, 38:241, 38:243, 38:244		<code>\new_whatsit</code>	41
<code>\mu</code>	30:290	<code>\newattribute</code>	40
<code>\mubyte</code>	57:401	<code>\newattribute</code>	04:74, 04:237
<code>\multicolumn</code>	41:310, 126	<code>\newbox</code>	12:13, 37:585, 38:66, 39:27,
<code>\multiput</code>	42:89, 42:821, 42:838	40:159, 41:16, 41:17, 41:18, 41:420,	
<code>\multispan</code>	41:310, 41:456, 1392	42:6, 42:680, 42:685, 02:47, 54:62,	
<code>\muskip</code>	04:34, 30:556, 30:557, 02:29, 02:51	54:98, 54:99, 54:100, 02:184, 02:398	
muskip commands:		<code>\newcatcodetable</code>	40
<code>\muskip_new:N</code>	11:95	<code>\newcatcodetable</code>	04:84,
<code>\l_tmpa_muskip</code>	373	04:93, 04:94, 04:120, 04:121, 04:241	
<code>\muskipdef</code>	04:227, 02:51	<code>\newcommand</code>	81
<code>\muskipzero</code>	04:227	<code>\newcommand</code>	21:4, 22:14,
<code>myhook (hook)</code>	221	25:2918, 25:2923, 25:2928, 29:37,	
N		29:73, 30:63, 30:64, 30:65, 30:66,	
<code>\n</code>	04:332, 04:334,	30:68, 30:69, 30:71, 30:72, 30:104,	
04:341, 04:343, 04:470, 04:696,		30:105, 30:106, 30:107, 30:108,	
04:721, 04:745, 04:786, 04:808,		30:109, 30:131, 30:132, 30:133,	
04:843, 04:851, 04:852, 04:858,		06:95, 36:263, 37:398, 37:399,	
04:883, 04:900, 04:939, 04:946,		37:400, 37:401, 40:351, 40:353,	
04:947, 04:954, 04:966, 57:121, 57:126		40:645, 40:646, 40:647, 40:649,	
<code>\nabla</code>	30:330	40:650, 40:651, 42:25, 42:692,	
<code>\NAME</code>	81	52:573, 52:574, 52:575, 52:576,	
<code>\narrower</code>	02:401	52:578, 54:2992, 54:2995, 54:2998,	
<code>\natural</code>	30:340	54:2999, 54:3002, 54:3003, 57:610, 118	
<code>\ncallback</code>	04:791	<code>\newcommand(*)</code>	128
<code>\ndefault</code>	04:796, 04:800	<code>\NewCommandCopy</code>	06:565,
<code>\ne</code>	30:424, 779	06:567, 07:1132, 07:1184, 07:1202,	
<code>\narrow</code>	30:416	07:1215, 07:1230, 07:1348, 1417	
<code>\NeedsDocumentMetadata</code>	1097	<code>\newcount</code>	12:7, 12:8, 18:125, 20:7,
<code>\NeedsDocumentMetadata</code>	50:806	22:58, 22:79, 22:98, 24:422, 26:26,	
<code>\NeedsFormat</code>	1362	28:28, 28:143, 28:501, 38:55, 38:415,	
<code>\NeedsTeXFormat</code>	26:12, 33:781, 50:782, 50:1850, 1366	38:416, 39:23, 39:24, 39:25, 39:26,	
<code>\neg</code>	30:336, 30:337	39:56, 39:257, 39:272, 40:536, 41:11,	
<code>\negmedspace</code>	38:214, 1418	41:12, 41:13, 41:14, 41:15, 41:412,	
		41:413, 41:414, 42:674, 42:675,	
		42:676, 42:677, 42:686, 44:36,	
		44:140, 44:141, 45:3, 45:267, 45:268,	
		45:269, 45:270, 50:1637, 02:47,	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 54:81, 54:83, 54:85, 54:87, 54:89,
54:97, 54:2597, 54:2990, 54:2993,
54:2996, 54:3000, 57:3, 57:4, 57:5,
57:91, 57:364, 02:231, 02:302, 559
- \newcounter 556
- \newcounter 22:10
- \newcounteralias 556
- \newcounteralias 22:14, 43:27, 1440
- \newdimen 12:10,
12:11, 12:12, 18:124, 26:415, 26:416,
38:53, 39:9, 39:10, 39:11, 39:12,
39:13, 39:14, 39:15, 39:16, 39:17,
39:18, 39:19, 39:20, 39:21, 39:22,
40:215, 40:216, 41:3, 41:5, 41:6,
41:7, 41:8, 41:166, 41:415, 41:416,
41:417, 41:418, 42:3, 42:4, 42:5,
42:7, 42:442, 42:443, 42:444, 42:445,
42:446, 42:447, 42:678, 42:679,
42:681, 42:682, 42:683, 42:684,
45:513, 02:47, 54:47, 54:48, 54:49,
54:51, 54:52, 54:53, 54:54, 54:55,
54:56, 54:57, 54:58, 54:59, 54:60,
54:61, 54:67, 54:69, 54:70, 54:71,
54:72, 54:84, 54:86, 54:88, 54:90,
54:91, 54:92, 54:93, 54:94, 54:95,
54:96, 54:507, 54:2598, 54:2599,
54:2604, 02:179, 02:181, 02:182, 02:301
- \NewDocumentCommand . 08:2761, 08:2763,
08:2765, 08:2769, 08:2771, 08:2773,
08:2784, 08:2786, 08:2796, 08:2798,
08:2807, 08:2809, 08:2816, 08:2818,
08:2820, 08:2822, 08:2824, 08:2848,
08:2850, 08:2853, 11:1569, 17:27,
17:44, 17:51, 17:116, 17:142, 17:165,
17:174, 17:175, 17:176, 20:769,
22:207, 22:222, 22:260, 22:272,
35:66, 35:68, 35:206, 36:80, 48:394,
51:214, 51:216, 51:227, 51:285,
57:682, 57:689, 57:696, 07:3103, 117
- \NewDocumentEnvironment
..... 07:2887, 07:2894, 07:3139, 117
- \newenvironment 82
- \newenvironment
06:187, 50:1339, 50:1472, 50:1561, 240
- \NewEnvironmentCopy . 06:606, 06:608, 1428
- \NewExpandableDocumentCommand 36:140,
48:402, 48:404, 48:406, 48:408,
48:411, 48:414, 57:650, 07:3222, 126
- \newfam 04:38, 24:17, 02:47, 1389
- \newfont 29:729
- \newgroup 28:48
- \newhelp 02:176
- \NewHook 08:2761, 08:2936,
20:73, 20:74, 20:75, 20:367, 20:370,
20:419, 26:151, 29:499, 29:500,
29:501, 29:502, 29:503, 29:504,
29:505, 29:506, 29:507, 37:36,
37:37, 37:38, 37:39, 37:40, 37:101,
37:102, 37:103, 37:104, 37:105,
50:1083, 50:1084, 52:177, 54:730, 219
- \NewHookPair 329
- \NewHookWithArguments 08:2767,
35:115, 41:465, 41:466, 41:468, 331
- \newif 03:73, 12:9, 20:5, 20:6,
24:267, 25:2789, 25:2801, 28:16,
29:609, 32:83, 33:832, 33:1185,
33:1186, 06:209, 35:3, 38:75, 38:76,
38:203, 38:417, 39:28, 39:29, 39:30,
39:31, 39:32, 39:33, 39:140, 40:486,
40:592, 41:19, 41:328, 42:168,
42:438, 42:439, 42:440, 42:441,
42:470, 42:471, 44:38, 44:124, 50:2,
06:964, 54:73, 54:74, 54:75, 54:76,
54:77, 54:78, 54:79, 54:80, 1357
- \newinsert 40:537, 45:390,
54:19, 54:2429, 02:116, 02:142, 1408
- \newlabel 35:84, 35:105, 35:120, 35:129, 1391
- \newlanguage 02:47, 57:286
- \newlength 569
- \newlength 23:3
- \newline 18:93, 18:100, 18:106
- \newlinechar 06:20, 02:256, 01:56
- \newluabytecode 41
- \newluabytecode 04:197, 04:251
- \newluachunkname 41
- \newluachunkname 04:205, 04:253
- \newluacmd 40
- \newluacmd 04:181, 41
- \newluafunction 40
- \newluafunction
..... 04:4, 04:173, 04:235, 04:247, 40
- \NewMarkClass 48:399,
48:520, 57:25, 57:26, 57:27, 1059
- \NewMarkRegion 1069
- \newmarks 48:22, 57:6, 1332
- \newmathalphabet 27:13, 27:109
- \NewMirroredHookPair
. 08:2761, 08:2938, 54:729, 54:731, 218
- \NewMirroredHookPairWithArguments ..
..... 08:2767, 41:467, 41:469, 218
- \NewModuleRelease
09:4, 03:152, 10:4, 11:7, 16:4, 17:3,
33:2, 36:4, 48:4, 07:9, 55:462, 08:4, 1421
- \newmuskip 02:47
- \newpage 54:111, 54:118, 54:129, 1244
- \NewProperty 36:38, 36:290, 850
- \newprotectedluacmd 41
- \newprotectedluacmd 04:181, 21:1034, 53:89

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \newread [02:47](#), [02:177](#), [1360](#)
- \NewReversedHook [08:2761](#),
[08:2937](#), [20:368](#), [20:369](#), [20:420](#),
[20:421](#), [50:1085](#), [50:1086](#), [52:178](#), [225](#)
- \NewReversedHookWithArguments
..... [08:2767](#), [218](#)
- \newrobustcmd [98](#)
- \newsavebox [925](#)
- \newsavebox [40:159](#)
- \newskip [12:14](#),
[12:15](#), [12:17](#), [18:472](#), [18:473](#), [18:474](#),
[18:539](#), [18:552](#), [23:3](#), [37:495](#), [38:418](#),
[38:553](#), [39:2](#), [39:3](#), [39:4](#), [39:5](#), [39:6](#),
[39:7](#), [39:8](#), [02:47](#), [54:3004](#), [54:3005](#),
[54:3006](#), [54:3010](#), [54:3011](#), [54:3014](#),
[54:3015](#), [54:3016](#), [54:3020](#), [54:3021](#),
[54:3022](#), [02:180](#), [02:183](#), [02:299](#), [02:300](#)
- \NewSocket [10:192](#), [10:213](#), [35:134](#),
[35:135](#), [54:663](#), [54:722](#), [54:723](#), [359](#)
- \NewSocketPlug [10:183](#), [10:192](#),
[10:217](#), [35:136](#), [54:664](#), [54:674](#),
[54:681](#), [54:688](#), [54:697](#), [54:702](#),
[54:707](#), [54:724](#), [54:727](#), [55:21](#), [355](#)
- \NewStructureName [1310](#)
- \NewStructureName [55:304](#)
- \NewTaggingSocket [1309](#)
- \NewTaggingSocket [55:7](#), [55:54](#),
[55:62](#), [55:72](#), [55:73](#), [55:84](#), [55:92](#),
[55:101](#), [55:103](#), [55:107](#), [55:117](#),
[55:118](#), [55:124](#), [55:125](#), [55:145](#),
[55:146](#), [55:206](#), [55:214](#), [55:215](#),
[55:216](#), [55:217](#), [55:218](#), [55:219](#),
[55:220](#), [55:221](#), [55:222](#), [55:223](#),
[55:224](#), [55:225](#), [55:226](#), [55:227](#),
[55:228](#), [55:229](#), [55:230](#), [55:231](#),
[55:232](#), [55:233](#), [55:234](#), [55:235](#),
[55:236](#), [55:237](#), [55:238](#), [55:239](#),
[55:240](#), [55:241](#), [55:242](#), [55:243](#),
[55:244](#), [55:245](#), [55:246](#), [55:247](#),
[55:248](#), [55:249](#), [55:250](#), [55:251](#),
[55:252](#), [55:253](#), [55:254](#), [55:255](#),
[55:256](#), [55:257](#), [55:258](#), [55:259](#),
[55:260](#), [55:261](#), [55:262](#), [55:263](#),
[55:264](#), [55:265](#), [55:266](#), [55:267](#),
[55:268](#), [55:269](#), [55:270](#), [55:271](#),
[55:272](#), [55:273](#), [55:274](#), [55:275](#),
[55:276](#), [55:277](#), [55:278](#), [55:279](#),
[55:280](#), [55:281](#), [55:282](#), [55:283](#),
[55:284](#), [55:285](#), [55:286](#), [55:287](#),
[55:288](#), [55:289](#), [55:290](#), [55:291](#),
[55:292](#), [55:293](#), [55:294](#), [55:295](#),
[55:296](#), [55:297](#), [55:298](#), [55:299](#), [1309](#)
- \NewTaggingSocketPlug [1309](#)
- \NewTaggingSocketPlug
- [55:20](#), [55:55](#), [55:63](#), [55:74](#),
[55:78](#), [55:85](#), [55:93](#), [55:102](#), [55:104](#),
[55:108](#), [55:119](#), [55:120](#), [55:121](#),
[55:122](#), [55:127](#), [55:137](#), [55:147](#),
[55:159](#), [55:170](#), [55:190](#), [55:208](#), [1309](#)
- \NewTemplateType
..... [11:1440](#), [11:1518](#), [11:1549](#), [370](#)
- \newtheorem [43:1](#), [1392](#)
- \newtie . [21:875](#), [33:152](#), [33:153](#), [33:172](#),
[33:669](#), [33:976](#), [33:977](#), [33:1284](#), [803](#)
- \newtoks
..... [03:36](#), [12:16](#), [16:77](#), [24:420](#), [24:421](#),
[25:2887](#), [26:248](#), [02:47](#), [02:176](#), [1408](#)
- \newwhatsit [41](#)
- \newwhatsit [04:189](#), [04:249](#)
- \newwrite [20:3](#), [20:4](#),
[44:154](#), [46:4](#), [46:21](#), [02:47](#), [02:178](#), [1382](#)
- \newXeTeXintercharclass [57:35](#)
- \next [1385](#)
- \NextLinkTarget [17:59](#), [1441](#)
- nfss internal commands:
 __nfss_init_mv_freeze:N
 [28:334](#), [28:359](#), [28:360](#)
 __nfss_transform_scriptfont:nw .
 [26:346](#), [26:350](#)
- \NG [21:554](#), [21:1173](#), [57:720](#), [1377](#)
- \ng [21:575](#), [21:1174](#), [57:720](#), [1377](#)
- \ni [30:441](#), [30:442](#)
- \noalign [30:350](#), [30:542](#), [30:545](#), [30:548](#),
[30:549](#), [30:553](#), [30:554](#), [38:171](#),
[38:172](#), [38:188](#), [38:191](#), [38:205](#),
[38:497](#), [38:507](#), [41:301](#), [41:307](#),
[41:436](#), [41:455](#), [42:159](#), [42:165](#), [1357](#)
- \nobreak [17:73](#), [17:84](#), [17:95](#),
[17:103](#), [18:60](#), [18:72](#), [18:116](#), [18:142](#),
[18:150](#), [18:165](#), [18:171](#), [18:183](#),
[18:196](#), [18:222](#), [18:420](#), [18:428](#),
[18:454](#), [18:462](#), [18:483](#), [18:490](#),
[18:550](#), [20:203](#), [20:218](#), [20:232](#),
[21:430](#), [21:456](#), [21:458](#), [21:591](#),
[21:1193](#), [37:420](#), [37:427](#), [40:710](#),
[40:720](#), [44:90](#), [44:237](#), [44:238](#),
[44:242](#), [44:260](#), [44:261](#), [44:265](#),
[45:435](#), [45:494](#), [45:593](#), [48:270](#),
[49:48](#), [49:58](#), [49:71](#), [49:79](#), [49:94](#),
[49:102](#), [06:978](#), [06:999](#), [54:341](#),
[54:1501](#), [54:1650](#), [54:1811](#), [57:208](#),
[57:210](#), [57:214](#), [57:215](#), [57:216](#),
[57:220](#), [02:378](#), [02:381](#), [02:383](#), [1398](#)
- \nobreakdashes [18:475](#)
- \nobreakspace [18:489](#), [480](#)
- \nobreakspace_ [480](#)
- \NoCaseChange [57:620](#), [1426](#)
- \nocite [1053](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\nocite` 47:59, 1384
`\nocorr` ... 32:44, 32:59, 32:63, 32:66, 1372
`\nocorrlist` 32:90, 32:122
`\noexpand` 1426
`\nofiles` 485
`\nofiles` 20:199, 870
`\noindent` 16:90, 24:767, 24:793, 44:139, 445
`\nointerlineskip` 30:350, 30:542,
30:545, 30:549, 30:553, 38:314,
38:342, 38:367, 42:573, 42:576,
42:586, 42:588, 54:2419, 54:2427, 02:373
`\nolimits` 30:359, 30:366,
38:3, 38:4, 38:5, 38:9, 38:10, 38:11,
38:12, 38:13, 38:14, 38:15, 38:16,
38:17, 38:18, 38:19, 38:20, 38:21,
38:26, 38:27, 38:28, 38:29, 38:31, 38:34
`\nolinebreak` 463
`\nolinebreak` 18:9, 18:28
`\nonfrenchspacing` 02:582, 20:48,
20:118, 20:176, 06:979, 06:1000, 02:306
`\nonscript` 38:36, 38:38
`\nonstopmode` 1073
`\nonumber` 38:480, 38:519, 38:520
`noop (plug)` 356, 358, 359, 1234, 1251
`\nopagebreak` 463
`\nopagebreak` 18:7, 18:26
`\noprotrusion` 44:247, 44:270
`\normalbaselines` 38:167,
38:169, 02:271, 02:284, 02:285, 02:310
`\normalbaselineskip`
26:189, 40:459, 40:478, 02:299, 02:311
`\normalcolor` 38:410,
38:549, 40:110, 40:528, 44:243,
44:266, 45:97, 45:166, 54:221,
54:620, 54:746, 54:919, 54:934,
54:984, 54:994, 54:1043, 54:1053,
54:2887, 54:2930, 54:2960, 1383
`normalfont (hook)` 244
`\normalfont` 24:744, 24:805,
29:764, 29:799, 29:801, 29:810,
29:815, 29:817, 29:826, 29:831,
29:833, 29:841, 29:848, 29:850,
29:856, 32:18, 37:573, 38:410,
38:549, 44:243, 44:266, 45:401, 1389
`normalfont` 29:499
`\normalize` 1370
`\normallineskip`
..... 40:457, 40:477, 02:299, 02:310
`\normallineskiplimit` 38:206,
40:442, 40:458, 40:464, 02:299, 02:311
`\normalmarginpar` 45:387
`\normalsfcodes`
20:44, 20:46, 20:48, 20:114, 20:116,
20:118, 20:172, 20:174, 20:176,
20:198, 54:903, 54:973, 54:1032, 1399
`\normalshape` 25:3202, 25:3243, 661
`\normalsize` 20:42, 20:112, 20:170,
32:143, 45:23, 45:176, 45:372,
50:5, 54:902, 54:972, 54:1031, 1368
`\not` 30:349, 30:423, 30:424, 30:446
`\notexpanded` 343
`\notin` 30:472
`\NoValue` 07:508,
07:516, 07:522, 07:938, 07:1005,
07:1066, 07:1548, 07:1826, 07:1963,
07:2068, 07:2229, 07:2236, 07:2302,
07:2350, 07:2426, 07:2449, 07:2533,
07:3280, 07:3282, 07:3288, 07:3294, 117
`\noexpand` 1362
`\nu` 30:291
`\null` 21:347,
21:383, 21:509, 21:512, 21:858,
21:861, 21:1281, 35:21, 35:38, 35:55,
37:538, 37:559, 37:722, 37:732,
38:112, 38:121, 38:169, 38:198,
44:237, 44:260, 53:438, 02:324, 1386
`\nulldelimiterspace` 30:653, 02:273
`\nullfont` 37:244
`\number` 03:52, 03:61,
04:105, 22:309, 24:693, 24:696,
26:456, 28:65, 28:94, 28:114, 28:129,
28:165, 28:196, 28:226, 28:258,
06:2, 29:756, 06:132, 50:1248,
50:1341, 50:1474, 50:1563, 53:398, 01:70
`\numberline` 44:72, 44:82, 44:273, 45:17, 1361
`\numexpr` 04:82, 04:105, 04:157, 21:1058,
28:151, 06:839, 02:112, 02:128, 02:138
`\unknown` 04:813
`\narrow` 30:418
`\nxt` 1385

O

- `\O` 21:264,
21:421, 21:556, 21:794, 21:1158, 57:719
`\o` 21:273,
21:426, 21:577, 21:805, 21:1164, 57:719
`\oalign` 02:410
`\obeycr` 18:578
`\obeyedline`
07:1652, 07:1697, 07:1770, 07:1772,
07:1774, 07:2246, 07:2252, 07:2261,
07:2268, 02:334, 02:338, 02:361, 126
`\obeyedspace` ... 02:339, 02:342, 02:362, 26
`\obeylines` 37:544, 37:565, 37:713, 37:714,
06:980, 06:1001, 54:834, 02:327, 26
`\obeyspaces`
.. 06:981, 06:1002, 54:834, 02:327, 26

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\oddsidemargin</code>	<code>\overbrace</code>	30:547
54:48, 54:50, 54:894, 54:966, 54:1025	<code>\overfullrule</code>	49:142, 02:267
<code>\odot</code>	<code>\overleftarrow</code>	30:544
<code>\OE</code>	<code>\overrightarrow</code>	30:541
21:263,	<code>\owns</code>	30:442, 30:443
21:420, 21:555, 21:793, 21:1175,		
57:662, 57:667, 57:672, 57:719, 1348		
<code>\oe</code>		
21:272,		
21:425, 21:576, 21:806, 21:1176,		
57:662, 57:667, 57:672, 57:719, 1348		
<code>\of</code>		
38:67, 38:414		
<code>off (plug)</code>		
54:727, 1234, 1251, 1251		
<code>\offinterlineskip</code>		
02:373		
<code>\oint</code>		
30:366		
<code>\ointop</code>		
30:365, 30:366		
<code>\oldstylenums</code>		
33:4,		
33:340, 33:341, 33:342, 33:343,		
33:344, 33:345, 33:346, 33:347,		
33:348, 33:349, 33:574, 33:1097, 799		
<code>\Omega</code>		
30:318		
<code>\omega</code>		
30:301		
<code>\ominus</code>		
30:398		
<code>\omit</code>		
38:191, 38:192,		
41:446, 41:449, 41:456, 41:460, 125		
<code>\OmitIndent</code>		
16:93, 16:172, 443		
<code>on (plug)</code>		
54:724, 1234		
<code>\onecolumn</code>		
54:120		
<code>\OnlyDescription</code>		
26:5, 31:3		
<code>\oalign</code>		
21:347, 21:377,		
21:415, 21:503, 21:509, 21:511,		
21:522, 21:538, 21:755, 21:788,		
21:858, 21:861, 21:918, 21:1281,		
29:761, 30:473, 30:476, 02:410, 1404		
<code>\openin</code>		
1414		
<code>\openout</code>		
1360		
<code>\openup</code>		
38:199, 38:204		
<code>\oplus</code>		
30:399		
<code>\OptionNotUsed</code> ...		
50:544, 50:571, 50:1151		
or commands:		
<code>\or:</code>		
07:1505,		
07:1506, 08:1273, 08:1274, 08:1275,		
08:1276, 08:1277, 08:1278, 08:1279,		
08:1280, 08:1281, 08:1422, 08:1438		
<code>\oslash</code>		
30:396		
<code>\OT</code>		
21:392		
<code>\otimes</code>		
30:397		
<code>\outer</code>		
04:21, 04:38, 1410		
<code>\output</code>		
54:261, 02:203		
<code>\outputmode</code>		
57:320		
<code>\outputpenalty</code>		
54:263,		
54:277, 54:300, 54:303, 54:304,		
54:339, 54:1511, 54:1512, 54:1660,		
54:1661, 54:1821, 54:1824, 02:203		
<code>\oval</code>		
42:461, 42:464, 42:822, 42:839		
<code>\over</code>		
30:480, 38:162, 38:412		
	P	
<code>\P</code>	21:330, 1382	
<code>package (hook)</code>	236	
<code>package/ (hook)</code>	1162, 1169	
<code>package/.../after</code>	1161	
<code>package/.../before</code>	1161	
<code>package/<name>/after (hook)</code>	1161	
<code>package/<name>/before (hook)</code>	1161	
<code>package/<package name>/after (hook)</code>	1162	
<code>package/<package name>/before (hook)</code>	1162	
<code>package/after (hook)</code>	1161, 1162	
<code>package/after</code>	1161	
<code>package/before (hook)</code>	1161, 1162	
<code>package/before</code>	1161	
<code>\PackageError</code> ..	03:77, 03:135, 03:147,	
14:84, 21:1575, 33:787, 33:838, 33:882		
<code>\PackageInfo</code> ...	14:84, 33:791, 33:807,	
33:812, 33:828, 33:829, 33:889, 33:1174		
<code>\PackageNote</code>	14:136, 1422	
<code>\PackageNoteNoLine</code>	14:136	
<code>\PackageWarning</code>		
14:84, 33:789, 33:839, 33:1103, 53:550		
<code>\PackageWarningNoLine</code>		
14:84, 21:1008, 54:2570		
<code>page</code>	36:257, 854	
<code>page@sofar (tag socket)</code>	55:270	
<code>\pagebreak</code>	463	
<code>\pagebreak</code> ..	18:6, 18:7, 18:23, 18:25, 1244	
<code>\pagegoal</code>	54:2457, 54:2464	
<code>pagenum</code>	36:258, 854	
<code>\pagenumbering</code>	839	
<code>\pagenumbering</code>	34:5	
<code>\pageref</code>	35:10, 854	
<code>\pageshrink</code> ..	54:529, 54:533, 54:548, 54:780	
<code>\pagestyle</code>	49:2	
<code>pagetarget</code>	36:263, 854	
<code>\pagetotal</code>	54:106	
<code>\paperheight</code>	54:69	
<code>\paperwidth</code>	54:69	
<code>\par</code>	01:104, 04:156, 15:3, 15:4,	
15:5, 16:142, 16:178, 18:261, 18:312,		
24:770, 06:9, 06:21, 37:186, 37:242,		
37:420, 37:427, 37:536, 37:557,		
39:63, 39:110, 39:127, 39:130,		
39:137, 39:192, 39:195, 40:448,		
40:469, 40:524, 40:554, 40:571,		
41:247, 41:270, 41:462, 44:41,		

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 44:90, 44:245, 44:267, 45:15, 45:24,
45:249, 45:344, 45:540, 45:558,
02:11, 49:135, 49:136, 53:454,
54:145, 54:172, 54:198, 54:262,
54:2463, 02:320, 02:336, 02:338,
02:355, 02:377, 02:386, 02:387,
02:388, 02:390, 02:392, 02:394, 1350
- para (plug) 1234
- para commands:
- \para_end: 16:95,
16:95, 16:142, 16:143, 16:144, 454
 - \g_para_indent_box .. 16:16, 16:49,
16:84, 16:86, 16:91, 16:93, 16:119, 448
 - \para_omit_indent: .. 16:88, 16:94, 445
 - \para_raw_end:
..... 16:115, 16:137, 16:140, 445
 - \para_raw_indent:
..... 16:115, 16:115, 16:138, 445
 - \para_raw_noindent:
..... 16:115, 16:127, 16:139, 445
- para internal commands:
- _para_handle_indent:
... 16:34, 16:66, 16:85, 16:85, 16:121
 - \g__para_standard_everypar_tl ...
16:12, 16:76, 16:78, 16:120, 16:131, 451
 - _para_tmp:w 16:37, 16:41, 16:68, 16:72
- para/after (hook) 443, 454
- para/after 16:6, 443
- para/before (hook) 443, 449
- para/before 16:6, 443
- para/begin (hook) 443, 445, 452
- para/begin (tag socket) 55:117
- para/begin 16:6, 443
- para/end (hook) 443, 444, 454
- para/end (tag socket) 55:117
- para/end 16:6, 443
- para/off (tag socket) 55:101
- para/on (tag socket) 55:101
- para/restore (tag socket) 55:107
- para/semantic/begin (tag socket) .. 55:124
- para/semantic/end (tag socket) ... 55:124
- para/textblock/begin (tag socket) . 55:145
- para/textblock/end (tag socket) .. 55:145
- \paracntvalue 446
- \paragraphmark 44:143
- \parallel 30:412
- \parbox 925
- \parbox 40:303, 441
- \parfillskip ... 24:743, 24:759, 24:804,
37:456, 37:466, 37:475, 37:483,
37:532, 37:554, 39:76, 40:456,
40:477, 44:232, 44:255, 02:298, 453
- \parindent 37:456, 37:461, 37:466, 37:475,
37:479, 37:483, 37:532, 37:554,
39:50, 40:451, 40:472, 44:233,
44:256, 02:279, 02:402, 02:403, 440
- \parsep 39:1, 39:49, 39:90, 909
- \parseunicodedataI 04:123, 04:162
- \parseunicodedataII 04:124, 04:126
- \parseunicodedataIII 04:128, 04:134
- \parseunicodedataIV 04:130, 04:142
- \parseunicodedataV 04:146, 04:149
- \parshape 39:54, 448
- \parskip 37:431, 37:530, 37:532,
37:552, 37:554, 38:621, 39:49, 39:73,
39:88, 39:90, 39:117, 39:184, 39:203,
39:254, 40:451, 40:472, 41:79,
54:1511, 54:1660, 54:1823, 02:286, 449
- \partial 30:326
- \partokencontext
... 57:724, 57:725, 57:729, 57:730, 1350
- \partopsep 38:619, 39:1, 39:61, 909
- \PassOptionsToClass 1096
- \PassOptionsToClass 50:476
- \PassOptionsToPackage 1096
- \PassOptionsToPackage 50:476
- \patterns 21:225, 1386
- \pausing 02:208
- pdfannot commands:
- \pdfannot_link_off: 53:19
 - \pdfannot_link_on: 53:18
- \pdfextension 53:15, 57:325
- \pdffilesize 05:76, 05:122
- \pdfgentounicode 57:317, 57:327, 57:329,
57:333, 57:359, 57:362, 57:364,
57:369, 57:374, 57:386, 57:391, 57:392
- \pdfglyphtounicode ... 57:319, 57:363,
57:820, 57:821, 57:822, 57:823,
57:824, 57:825, 57:826, 57:827,
57:828, 57:829, 57:830, 57:831,
57:832, 57:833, 57:834, 57:835,
57:836, 57:837, 57:838, 57:839,
57:840, 57:841, 57:842, 57:843,
57:844, 57:845, 57:846, 57:847,
57:848, 57:849, 57:850, 57:851,
57:852, 57:853, 57:854, 57:855,
57:856, 57:857, 57:858, 57:859,
57:860, 57:861, 57:862, 57:863,
57:864, 57:865, 57:866, 57:867,
57:868, 57:869, 57:870, 57:871,
57:872, 57:873, 57:874, 57:875,
57:876, 57:877, 57:878, 57:879,
57:880, 57:881, 57:882, 57:883,
57:884, 57:885, 57:886, 57:887,
57:888, 57:889, 57:890, 57:891,
57:892, 57:893, 57:894, 57:895,
57:896, 57:897, 57:898, 57:899,
57:900, 57:901, 57:902, 57:903,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- 57:904, 57:905, 57:906, 57:907,
57:908, 57:909, 57:910, 57:911,
57:912, 57:913, 57:914, 57:915,
57:916, 57:917, 57:918, 57:919,
57:920, 57:921, 57:922, 57:923,
57:924, 57:925, 57:926, 57:927,
57:928, 57:929, 57:930, 57:931,
57:932, 57:933, 57:934, 57:935,
57:936, 57:937, 57:938, 57:939,
57:940, 57:941, 57:942, 57:943,
57:944, 57:945, 57:946, 57:947,
57:948, 57:949, 57:950, 57:951, 1441
\pdfhorigin 53:375
\pdfsavepos 854
\pdftexrevision 57:334, 57:376
\pdftexversion
57:332, 57:333, 57:334, 57:373, 57:376
\pdfvariable
53:374, 53:379, 57:327, 57:357, 57:371
\pdfvorigin 53:380
peek commands:
 \peek_meaning:NTF 07:2775, 204
 \peek_meaning_remove:NTF
 07:2106, 07:2223, 07:2763, 188
 \peek_N_type:TF
 07:2112, 07:2136, 07:2168
 \peek_remove_spaces:n 07:2104
\penalty .. 18:35, 18:38, 18:47, 18:318,
18:328, 18:351, 18:361, 18:385,
18:389, 32:119, 37:538, 37:541,
37:559, 37:562, 38:37, 38:207,
38:497, 38:507, 39:221, 41:56,
45:195, 45:199, 45:201, 45:217,
45:221, 45:223, 47:37, 54:114,
54:155, 54:182, 54:200, 54:203,
54:1509, 54:1658, 54:1819, 02:382,
02:383, 02:384, 02:385, 02:386,
02:387, 02:391, 02:393, 02:395, 1359
\perp 30:458
\phantom 38:75
\Phi 30:316
\phi 30:298
\Pi 30:313
\pi 30:293
picture (env.) 42:21
\picture 42:21
Plugs:
 default 1234
 floats:footnotes 54:697, 1234
 floats:space:footnotes .. 54:681, 1233
 footnotes:floats 54:702, 1234
 footnotes:floats:legacy . 54:707, 1234
 footnotes:space:floats .. 54:674, 1233
 identity 356, 359, 359, 359
 kernel (refstepcounter/target) 35:136
 noop 356, 358, 359, 1234, 1251
 off 54:727, 1234, 1251, 1251
 on 54:724, 1234
 para 1234
 space:floats:footnotes .. 54:688, 1233
 space:footnotes:floats .. 54:664, 1233
 tagpdf 359
\pm 30:401
\pmatrix 38:173, 38:181
\pmod 38:39
\PopDefaultHookLabel 08:2820, 224
\poptabs 14:256, 41:142, 41:161
\poptracing 26:149, 26:341
\postdisplaypenalty
 18:13, 38:561, 38:573, 38:599, 02:200
\pounds 21:338, 1377
\Pr 38:32
pre commands:
 pre_shipout_filter 1188
\prec 30:432
\preceq 30:435
\predisplaypenalty 18:12,
 38:560, 38:572, 38:598, 02:199, 1428
\predisplaysize 02:276
\pretolerance 24:745, 24:761, 24:806, 02:186
\prevdepth 18:324, 18:325,
 18:357, 18:358, 18:418, 18:423,
 18:452, 18:457, 38:205, 45:196,
 45:198, 45:218, 45:220, 54:146,
 54:148, 54:151, 54:173, 54:175,
 54:178, 02:373, 02:377, 02:378, 476
\PreviousTotalPages 53:460, 1189
prg commands:
 \prg_break:n 07:3264,
 07:3266, 07:3268, 07:3270, 07:3271
 \prg_break_point: 07:3272
 \prg_do_nothing: 08:2346,
 09:209, 09:284, 09:421, 09:640,
 48:265, 52:132, 52:134, 53:227,
 53:228, 53:233, 53:242, 53:246,
 53:382, 53:538, 55:12, 55:36, 55:46,
 07:410, 07:999, 07:1005, 07:2048, 158
 \prg_generate_conditional_-
 variant:Nnn
 11:166, 36:187, 36:202, 36:217
 \prg_gset_conditional:Npnn . 07:3284
 \prg_new_conditional:Npnn
 08:2461, 08:2476, 08:2493,
 08:2510, 08:2516, 08:2522, 08:2528,
 08:2537, 08:2545, 08:2559, 08:2577,
 09:327, 10:53, 10:117, 10:138, 11:9,
 11:160, 11:167, 11:173, 36:176,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- 36:191, 36:206, 48:299, 48:306,
55:26, 08:267, 08:903, 08:1216, 08:1426
- \prg_new_protected_conditional:Npnn
..... 09:123,
09:484, 08:754, 08:812, 08:842, 08:874
- \prg_replicate:nn 08:2329,
08:2337, 08:2407, 08:2423, 52:552,
07:906, 07:949, 07:1010, 07:1065,
07:2532, 07:2539, 07:2824, 08:1190
- \prg_return_false: 08:2471, 08:2487,
08:2505, 08:2514, 08:2520, 08:2526,
08:2534, 08:2543, 08:2552, 08:2555,
08:2569, 08:2584, 09:138, 09:340,
09:499, 10:56, 10:120, 10:142,
11:28, 11:32, 11:37, 11:164, 11:171,
11:177, 36:184, 36:199, 36:214,
48:304, 48:311, 55:27, 07:3291,
08:273, 08:781, 08:835, 08:847,
08:867, 08:879, 08:894, 08:911,
08:915, 08:918, 08:1221, 08:1431, 269
- \prg_return_true:
08:2469, 08:2485, 08:2504, 08:2507,
08:2513, 08:2519, 08:2525, 08:2532,
08:2542, 08:2553, 08:2567, 08:2582,
09:137, 09:340, 09:498, 10:55,
10:119, 10:141, 11:28, 11:32, 11:37,
11:163, 11:170, 11:176, 36:181,
36:196, 36:211, 48:303, 48:310,
07:3289, 08:272, 08:772, 08:826,
08:865, 08:892, 08:914, 08:1219, 08:1429
- \prime 30:264, 30:328, 38:256
- \ProcessedArgument ... 07:391, 07:395,
07:402, 07:2465, 07:2466, 07:2483,
07:2485, 07:2507, 07:2516, 07:2522,
07:2530, 07:2547, 07:2558, 07:2561,
07:2586, 07:2652, 07:2654, 07:3315, 198
- \ProcessKeyOptions ... 51:86, 51:227, 1425
- \ProcessKeyPackageOptions 1425
- \ProcessList 07:3320, 124
- \ProcessOption* 1362
- \ProcessOptions 21:1600, 26:72, 33:805,
33:842, 50:572, 50:663, 50:1236, 1368
- \ProcessOptions* 50:572
- \ProcessorA 123
- \ProcessorB 123
- \prod 30:360
- prop commands:
- \prop_clear:N 11:230,
11:240, 11:256, 11:266, 11:273,
11:302, 11:303, 11:304, 07:1956, 08:1163
- \prop_clear_new:N 11:206
- \prop_const_from_keyval:Nn
..... 08:957, 08:965, 08:967, 08:969
- \prop_gclear:N 09:74, 11:193,
08:990, 08:1025, 08:1447, 08:1474
- \prop_gclear_new:N 08:2448,
08:2458, 11:182, 11:214, 11:217
- \prop_get:NnN ... 08:1617, 08:1703,
11:120, 11:705, 11:708, 11:735,
11:775, 11:855, 11:862, 11:869,
11:1002, 11:1023, 11:1115, 08:587
- \prop_get:NnNTF 11:467,
11:582, 11:686, 11:797, 11:823,
11:842, 11:895, 11:1030, 11:1126,
11:1199, 51:232, 07:1962, 08:596, 08:641
- \prop_gpop:NnNTF 08:998, 08:1033
- \prop_gput:Nnn
..... 08:1562, 08:2612, 08:2613,
08:2614, 09:67, 10:191, 11:288,
11:1548, 17:29, 48:313, 07:58,
08:597, 08:598, 08:644, 08:647, 08:1524
- \prop_gset_eq:NN
11:183, 11:196, 11:215, 11:218, 08:1165
- \prop_if_empty:NTF
.. 08:2004, 08:2481, 08:1504, 08:1539
- \prop_if_empty_p:N .. 08:1900, 08:2498
- \prop_if_exist:NTF
08:2108, 08:2480, 08:2518, 11:190,
11:224, 11:234, 11:250, 11:260, 11:267
- \prop_if_exist_p:N 08:1899
- \prop_if_in:NnTF .. 11:143, 11:277,
11:699, 11:703, 08:849, 08:857,
08:861, 08:881, 08:884, 08:888, 08:913
- \prop_item:Nn 11:1064, 17:37
- \prop_map_break:
..... 08:1594, 08:1676, 08:2116
- \prop_map_function:NN
..... 09:73, 11:1299,
11:1310, 17:48, 17:55, 08:1164, 279
- \prop_map_inline:Nn
..... 08:1582, 08:1589, 08:1591,
08:1664, 08:1671, 08:1673, 08:1832,
08:1903, 08:2007, 08:2111, 08:2113,
11:565, 51:267, 08:1459, 08:1480
- \prop_new:N
.. 09:17, 11:75, 11:86, 11:88, 11:90,
11:91, 11:195, 17:26, 07:53, 08:30,
08:31, 08:146, 08:158, 08:1314, 08:1315
- \prop_pop:NnNTF 11:1028
- \prop_put:Nnn ... 08:1802, 08:1813,
11:352, 11:364, 11:513, 11:516,
11:519, 11:644, 11:656, 11:661,
11:681, 11:744, 11:887, 11:921,
11:928, 11:933, 11:969, 07:1953, 08:1199
- \prop_remove:Nn 11:588
- \prop_set_eq:NN
..... 08:1557, 11:207, 11:227,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- 11:237, 11:253, 11:263, 11:270, 08:1520
 \prop_show:N 304
 property commands:
 \property_gset:nnnn
 36:6, 36:17, 36:46, 850
 \property_if_exist:n . 36:176, 36:187
 \property_if_exist:nTF
 .. 36:176, 36:188, 36:189, 36:190, 852
 \property_if_exist_p:n ... 36:176, 852
 \property_if_recorded:n 36:191, 36:202
 \property_if_recorded:nn
 36:206, 36:217
 \property_if_recorded:nnTF 36:206,
 36:218, 36:219, 36:220, 36:241, 852
 \property_if_recorded:nTF
 36:191, 36:203,
 36:204, 36:205, 36:230, 36:239, 852
 \property_if_recorded_p:n 36:191, 852
 \property_if_recorded_p:nn 36:206, 852
 \property_item:nn
 36:118, 36:118, 36:126, 858
 \property_item:nnn
 36:127, 36:127, 36:139, 851
 \property_new:nnnn
 . 36:6, 36:6, 36:41, 36:255, 36:257,
 36:258, 36:259, 36:260, 36:262,
 36:264, 36:265, 36:266, 36:267, 850
 \property_record:nN . 36:48, 36:48, 851
 \property_record:nn
 . 36:48, 36:49, 36:50, 36:52, 36:85, 851
 \property_ref:nn
 36:89, 36:89, 36:96, 36:144, 851
 \property_ref:nnn
 36:97, 36:97, 36:117, 36:147, 851
 \property_ref_undefined_warn: ...
 36:221, 36:221, 851
 \property_ref_undefined_warn:n ..
 36:228, 36:228, 36:236, 852
 \property_ref_undefined_warn:nn .
 .. 36:237, 36:237, 36:253, 36:254, 850
 property internal commands:
 __property_data:nnn
 36:150, 36:162, 36:164, 36:169
 __property_gset:nnnn
 36:6, 36:10, 36:19, 36:22, 36:37
 __property_record:nn
 36:48, 36:51, 36:53, 36:62
 __property_record_value:n
 36:48, 36:59, 36:63
 __property_record_value_aux:n ..
 36:48, 36:64, 36:65, 36:79
 __property_ref:nnn
 ... 36:91, 36:97, 36:99, 36:104, 36:116
 l__property_ref_flag 36:88
 \propto 30:409
 \protect 14:246, 14:248, 14:250,
 14:256, 14:262, 14:269, 14:279,
 14:282, 14:288, 20:215, 20:228,
 21:44, 21:50, 21:69, 21:73, 21:229,
 21:237, 28:762, 28:1310, 29:749,
 32:144, 06:120, 35:16, 35:33, 35:50,
 06:273, 37:282, 37:285, 37:319,
 37:329, 06:338, 06:347, 06:352,
 06:355, 06:356, 06:358, 06:359,
 06:364, 06:365, 06:370, 06:373,
 06:374, 06:399, 06:437, 06:465,
 41:341, 44:12, 44:72, 44:82, 44:164,
 44:171, 44:177, 45:17, 06:672,
 06:692, 47:5, 52:281, 52:304, 53:144,
 53:156, 53:186, 53:193, 53:213,
 54:881, 54:951, 54:1010, 57:407, 1439
 \protected 04:186,
 17:8, 18:57, 18:260, 18:311, 18:496,
 18:560, 21:328, 21:329, 22:30,
 22:361, 25:3202, 25:3206, 25:3209,
 25:3212, 25:3215, 25:3218, 25:3221,
 25:3224, 28:1152, 06:7, 29:656,
 29:702, 06:286, 06:293, 06:294,
 37:203, 37:249, 06:310, 06:312,
 06:314, 37:419, 06:329, 38:199,
 38:481, 38:519, 38:538, 38:539,
 40:334, 40:338, 40:341, 40:628,
 41:56, 41:278, 41:285, 06:547,
 42:153, 45:408, 45:467, 50:806,
 53:570, 53:571, 53:572, 54:845,
 54:846, 57:319, 57:327, 02:338, 669
 \Provide 1381
 \Provide... 117
 \providecommand 08:2945,
 08:2948, 21:6, 21:13, 21:1005,
 06:219, 35:112, 35:113, 35:114, 54:2581
 \ProvideCommandCopy 100
 \ProvideDocumentCommand ... 07:3103, 117
 \ProvideDocumentEnvironment 07:3139, 117
 \ProvideExpandableDocumentCommand ..
 07:3222, 125
 \ProvideHook 08:2891, 1421
 \ProvideMirroredHookPair 08:2891
 \ProvideReversedHook 08:2891
 provides commands:
 provides_module 04:299
 \provides_module 43
 \ProvidesClass 1096
 \ProvidesClass 50:456
 \ProvidesExplPackage 52:541
 \ProvidesFile .. 30:676, 30:678, 30:679,
 30:680, 33:1178, 50:465, 01:73, 1384
 \ProvidesPackage 1096

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- \ProvidesPackage 26:13,
33:779, 33:810, 50:379, 50:458,
50:460, 50:1851, 52:570, 53:565, 1366
- \ProvideTextCommand 21:3, 21:78, 1381
- \ProvideTextCommandDefault .. 21:75, 1390
- \Psi 30:317
- \psi 30:300
- \PushDefaultHookLabel 08:2820, 225
- \pushtabs
14:256, 41:139, 41:138, 41:158, 41:160
- \pushtracing 26:118, 26:322
- \put 42:67, 42:69, 42:81, 42:83,
42:336, 42:337, 42:338, 42:339,
42:347, 42:349, 42:362, 42:363,
42:364, 42:365, 42:372, 42:375,
42:395, 42:396, 42:397, 42:398,
42:404, 42:406, 42:418, 42:419,
42:420, 42:421, 42:426, 42:431,
42:739, 42:795, 42:823, 42:840, 1185
- Q**
- \qbezier 974
- \qbezier 42:692, 42:824, 42:841
- \qbeziermax . 42:691, 42:717, 42:718, 42:778
- \quad 18:571
- \quad 18:571,
38:168, 38:170, 38:190, 44:111, 1388
- quark commands:
- \q_mark 07:1340,
07:1343, 07:2374, 07:2378, 07:2390,
07:2712, 07:2720, 07:2723, 07:2728
- \q_nil 11:162, 11:175,
11:834, 11:836, 07:601, 07:1332,
07:1343, 07:1516, 07:1769, 07:1770,
07:1772, 07:1773, 07:1774, 07:1775,
07:1776, 07:1862, 07:1883, 07:1887,
07:1893, 07:1902, 07:1904, 07:2300,
07:2322, 07:2331, 07:2359, 07:2374,
07:2378, 07:2390, 07:2396, 07:2420, 192
- \quark_if_nil:NTF 07:612, 07:2380, 193
- \quark_if_nil:NTF 07:1513
- \quark_if_recursion_tail_stop:N .
..... 07:828, 07:1497
- \quark_if_recursion_tail_stop:n .
.... 36:166, 51:7, 07:483, 07:673,
07:1258, 07:2725, 07:2726, 08:416
- \quark_if_recursion_tail_stop_-
do:Nn 07:620
- \quark_if_recursion_tail_stop_-
do:nn 09:440,
09:459, 07:507, 07:512, 07:521,
07:528, 07:574, 07:587, 07:597, 07:641
- \quark_new:N 09:15,
09:16, 11:103, 07:50, 07:51, 07:1871
- \q_recursion_stop
... 09:429, 36:162, 51:20, 07:336,
07:347, 07:460, 07:811, 07:1254,
07:1493, 07:1954, 07:1969, 07:1971,
07:1978, 07:2715, 07:2721, 08:422
- \q_recursion_tail
..... 09:429, 36:162, 51:19,
07:460, 07:811, 07:1254, 07:1493,
07:1969, 07:1973, 07:2711, 07:2714,
07:2720, 07:2721, 08:421, 08:422, 146
- \q_stop 11:16,
11:18, 11:40, 11:43, 11:46, 11:162,
11:175, 11:793, 11:811, 11:834,
11:836, 07:423, 07:601, 07:610,
07:615, 07:817, 07:818, 07:1340,
07:1342, 07:1770, 07:1776, 07:1887,
07:1904, 07:2523, 08:569, 08:574
- quark internal commands:
- \q__cmd ... 07:407, 07:411, 07:1034,
07:1370, 07:1371, 07:1452, 07:1460,
07:1869, 07:2286, 07:2287, 07:2288,
07:2293, 07:2299, 07:2300, 07:2302,
07:2304, 07:2309, 07:2312, 07:2315,
07:2326, 07:2330, 07:2331, 07:2334,
07:2335, 07:2336, 07:2347, 07:2350,
07:2352, 07:2363, 07:2365, 07:2366,
07:2367, 07:2368, 07:2369, 07:2370,
07:2371, 07:2375, 07:2394, 07:2395,
07:2396, 07:2397, 07:2398, 07:2399,
07:2400, 07:2401, 07:2402, 07:2403,
07:2404, 07:2405, 07:2406, 07:2407,
07:2408, 07:2413, 07:2419, 07:2420,
07:2426, 07:2429, 07:2430, 07:2431,
07:2442, 07:2449, 07:2452, 07:2453,
07:2454, 07:2457, 07:2458, 07:2460, 193
- \q__cmd_recursion_stop
..... 07:51, 07:2592,
07:2594, 07:2603, 07:2626, 07:2635, 133
- \q__cmd_recursion_tail
..... 07:50, 07:2592, 133
- \q__hook_recursion_stop
... 09:15, 09:342, 09:343, 09:352, 09:394
- \q__hook_recursion_tail
..... 09:15, 09:342, 09:356
- \q__template_nil . 11:103, 11:1203, 383
- \quotedblbase 21:578, 21:807, 21:1204
- \quotesinglbase 21:579, 21:1201
- R**
- \r 21:256, 21:409, 21:452,
21:493, 21:633, 21:660, 21:670,
21:696, 21:780, 21:819, 21:1273,
21:1291, 21:1317, 21:1439, 21:1440,
33:147, 33:169, 02:318, 02:319, 1377

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \radical .. 28:1176, 28:1179, 28:1209, 1409
- \raggedbottom 49:126, 1244
- \raggedleft . 37:462, 37:480, 37:491, 37:498
- \raggedright 37:457, 37:476, 37:490, 37:496
- \raise 21:347,
 - 21:379, 21:451, 21:454, 21:756,
 - 21:821, 21:919, 21:1281, 29:762,
 - 30:476, 30:524, 30:526, 38:73,
 - 40:690, 40:699, 42:72, 42:84, 42:116,
 - 42:128, 42:206, 42:316, 42:463,
 - 42:505, 42:536, 42:561, 42:629,
 - 42:646, 42:647, 42:750, 42:806, 45:414
- \raisebox 926
- \raisebox 21:896, 21:1250, 40:667
- \rangle 30:603
- \RawIndent 16:138, 16:174, 445
- \RawNoIndent 16:138
- \RawNoindent 16:139, 16:175, 445
- \RawParEnd ... 16:138, 16:176, 02:357, 445
- \RawShipout 53:212, 53:491, 1186
- \rbrace 21:329, 30:607
- \rbrack 02:316
- \rceil 30:611
- \Re 30:324
- \read 505
- \ReadOnlyShipoutCounter
 - 53:393, 53:493, 1189
- \RecordProperties 36:80, 36:293, 851
- \Ref 841
- \Ref 35:199, 35:201, 35:211,
 - 35:213, 35:217, 35:219, 35:223, 35:226
- \ref 35:10, 35:213, 35:219, 45:615, 854
- \RefProperty 36:140, 36:294, 851
- \refstepcounter 556
- refstepcounter (socket) 35:134
- refstepcounter (tag socket) 55:206
- \refstepcounter 35:132, 38:399, 38:406,
 - 38:600, 39:233, 43:51, 44:59, 45:9, 854
- refstepcounter/target (socket) ... 35:135
- \RefUndefinedWarn ... 36:237, 36:295, 853
- regex commands:
 - \regex_match:nnTF 11:534
- \registernumber 42
- registernumber 04:390
- \relax 1210
- \relax_ 1375
- \Relbar 30:487, 30:495, 30:497, 30:503, 1403
- \relbar 30:484, 30:499, 30:501
- \relpenalty 02:194
- remove commands:
 - remove_from_callback 04:937
- \remove_from_callback 44
- \RemoveFromHook
 - 08:2818, 08:2947, 37:41, 37:42,
 - 37:43, 37:106, 37:107, 37:108, 221
- \removelastskip
 - 02:389, 02:391, 02:393, 02:395
- \Renew... 117
- \renewcommand 81
- \renewcommand 30:78, 30:80, 30:82,
 - 30:83, 30:85, 30:87, 30:89, 30:90,
 - 30:96, 30:98, 30:100, 30:101, 30:114,
 - 30:115, 30:116, 30:124, 30:125,
 - 06:163, 38:548, 38:568, 38:589, 737
- \RenewCommandCopy ... 06:565, 06:567, 100
- \RenewDocumentCommand 07:3103, 117
- \RenewDocumentEnvironment
 - 07:2901, 07:3139, 117
- \renewenvironment 82
- \renewenvironment . 06:193, 38:597, 38:609
- \RenewEnvironmentCopy 06:606, 06:608, 1428
- \RenewExpandableDocumentCommand
 - 48:522,
 - 48:523, 48:524, 48:525, 07:3222, 125
- \repeat 04:154, 04:164, 24:787,
 - 41:459, 50:1289, 50:1350, 50:1420,
 - 50:1483, 50:1534, 50:1572, 57:418,
 - 57:429, 57:439, 57:450, 57:480,
 - 57:506, 57:516, 01:65, 01:67, 02:365
- \requestedLaTeXdate 08:2967,
 - 50:1635, 50:1676, 50:1697, 50:1784
- \requestedpatchdate 50:1707, 50:1785
- \RequirePackage 1096
- \RequirePackage 04:24, 50:711, 50:718,
 - 50:761, 50:770, 50:1232, 54:2578, 1362
- \RequirePackageWithOptions 1096
- \RequirePackageWithOptions 50:753
- reserved@a commands:
 - \reserved@a: 20:252,
 - 20:324, 20:386, 20:433, 50:1262, 50:1397
- reserved@b commands:
 - \reserved@b: 50:267, 50:284
- reserved@c commands:
 - \reserved@c: 20:774
- \RestoreAtCatcode 1376
- \restorecr 18:578
- \RestoreLastSkip
 - 17:67, 17:69, 17:73, 17:75, 17:82,
 - 17:95, 17:101, 17:118, 17:122, 1442
- \ResumeTagging 1309
- \ResumeTagging 23:22, 55:3, 1309
- \ReverseBoolean 07:3316, 124
- \reversemarginpar 45:387
- \rfloor 30:615
- \rgroup 30:619
- \rhd 29:889

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\rho` 30:294
 - `\rhook` 30:490, 30:491
 - `\right` 30:636, 30:638,
30:640, 30:642, 30:647, 30:648,
30:649, 30:650, 38:168, 38:173, 38:197
 - `\Rightarrow` 30:422, 30:497, 30:509
 - `\rightarrow` 30:449, 30:451,
30:455, 30:489, 30:499, 30:507, 30:560
 - `\rightarrowfill` 30:542, 30:558
 - `\rightharpoonowdown` 30:466
 - `\rightharpoonup` 30:465, 30:477
 - `\righthyphenmin` 56:11, 02:246
 - `\rightleftharpoons` 30:475
 - `\rightline` 40:725
 - `\rightmargin` 39:9, 39:40, 39:51, 909
 - `\rightmark` 49:106, 1064
 - `\rightskip` 24:758, 37:454, 37:458,
37:464, 37:474, 37:477, 37:482,
37:531, 37:553, 39:75, 40:455,
40:476, 44:232, 44:255, 02:292, 02:403
 - `\rlap` 21:451, 21:454, 21:821,
38:521, 38:549, 40:729, 41:81, 1389
 - `\rm` 1365
 - `\rmdefault` 29:7, 29:301, 29:477,
29:486, 29:518, 29:529, 29:561,
29:779, 30:62, 30:131, 33:7, 33:577, 554
 - `rmfamily (hook)` 244, 244, 244
 - `\rmfamily` .. 29:5, 29:6, 29:484, 29:527,
29:528, 29:559, 29:560, 32:15, 244
 - `rmfamily` 29:499
 - `\rmmath` 1358
 - `\rmoustache` 30:576
 - `\rmsubstdefault`
.... 30:18, 30:30, 33:28, 33:39, 33:49
 - `\Roman` 556
 - `\Roman` 22:305, 1189
 - `\roman` 556
 - `\roman` 22:304
 - `\romannumeral`
... 22:310, 22:311, 37:318, 37:350,
37:367, 39:43, 39:265, 39:276, 875
 - `\root` 38:66, 38:414
 - `\rootbox` 38:66
 - `\rowcolor` 972
 - `\rq` 02:314
 - `\rule` 926
 - `\rule` 40:553, 40:570,
40:587, 40:593, 45:539, 45:557, 45:575
- S**
- `\S` 21:331, 1382
 - `\safesubencodingfoundfalse` 33:1219
 - `\safesubencodingfoundtrue` 33:1209
 - `\samepage` 463
 - `\samepage` 18:11, 18:29, 1359
 - `\SaveAtCatcode` 1376
 - `\savebox` 925
 - `\savebox` 40:160
 - `\savecatcodetable` . 04:117, 04:168, 04:170
 - `\SaveLastSkip` . 17:68, 17:118, 17:122, 1442
 - `\savepos` 854
 - `\sb` 38:212
 - `\sbox` 925
 - `\sbox` 19:4, 21:521, 21:537, 39:236,
40:166, 40:173, 40:177, 40:182,
40:187, 45:413, 45:472, 02:397, 1391
 - scan commands:
 - `\scan_new:N` 11:101, 11:102, 08:38
 - `\scan_stop:` 08:2578,
08:2579, 09:486, 09:487, 09:545,
09:546, 09:581, 09:582, 16:96,
48:106, 48:108, 48:118, 48:120,
48:122, 48:123, 48:280, 48:281,
48:282, 51:88, 51:98, 53:25, 53:27,
53:28, 53:50, 53:52, 53:53, 53:105,
07:1614, 07:1615, 07:1640, 07:1646,
07:1655, 07:1662, 07:2101, 07:2102,
08:683, 08:690, 08:701, 08:710,
08:745, 08:756, 08:792, 08:804,
08:814, 08:844, 08:876, 08:1218, 1152
 - scan internal commands:
 - `\s__file_stop` 52:103,
52:105, 52:108, 52:110, 52:112, 52:125
 - `\s__hook_mark` 08:2295,
08:2309, 08:2358, 08:2366, 08:2469,
08:2471, 08:2532, 08:2534, 08:2538,
08:2539, 08:2546, 08:2547, 08:2560,
08:2561, 09:23, 09:25, 09:40, 09:42,
09:51, 09:53, 09:129, 09:134, 09:328,
09:339, 09:491, 09:494, 09:506,
09:510, 09:515, 09:550, 09:570,
09:586, 09:598, 08:38, 08:39,
08:40, 08:342, 08:345, 08:348,
08:352, 08:355, 08:356, 08:734,
08:785, 08:787, 08:795, 08:797,
08:800, 08:802, 08:904, 08:929,
08:930, 08:934, 08:1239, 08:1256,
08:1260, 08:1302, 08:1303, 08:1358, 247
 - `\s__keys_stop` ... 51:47, 51:206, 51:207
 - `\s__template_mark` 11:101, 11:1203, 383
 - `\s__template_stop`
..... 11:102, 11:377, 11:388,
11:399, 11:420, 11:434, 11:442,
11:594, 11:597, 11:620, 11:625,
11:1053, 11:1056, 11:1203, 11:1207, 383
 - `\scdefault` 25:3214, 29:28, 30:106
 - `\scriptfont` 26:339
 - `\scriptscriptfont` 26:340

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \scriptscriptstyle 38:65, 38:68
- \scriptspace 45:435, 45:494, 02:274
- \scriptstyle 30:348, 38:64
- \scshape 21:320, 25:3212,
25:3213, 29:26, 29:27, 32:23, 1399
- \searrow 30:417
- \sec 38:20
- sec/begin (tag socket) 55:220
- sec/end (tag socket) 55:220
- sec/title/begin (tag socket) 55:222
- sec/title/end (tag socket) 55:222
- sec/title/hang (tag socket) 55:224
- sec/title/init (tag socket) 55:225
- sec/title/number (tag socket) 55:228
- sec/title/runin/number (tag socket) 55:225
- sec/title/split (tag socket) 55:225
- \secdef 44:142
- \secondoftwo 500
- \section 328
- \sectionmark 44:143, 1064
- selectfont (hook) 245
- \selectfont 19:7,
21:322, 21:349, 21:380, 21:470,
21:833, 21:895, 21:1249, 21:1283,
21:1601, 24:354, 24:364, 24:374,
25:2916, 25:2921, 25:2926, 25:3204,
25:3208, 25:3211, 25:3214, 25:3217,
25:3220, 25:3223, 25:3226, 26:115,
26:116, 26:160, 26:163, 26:165, 29:7,
29:10, 29:13, 29:16, 29:19, 29:22,
29:25, 29:28, 29:31, 29:343, 29:366,
29:404, 29:424, 29:439, 29:450,
29:461, 29:464, 29:488, 29:493,
29:498, 29:531, 29:536, 29:541,
29:555, 29:558, 29:561, 29:564,
29:567, 29:647, 29:692, 29:752,
29:776, 29:809, 29:824, 29:839,
33:36, 33:58, 33:68, 33:600, 33:637,
33:877, 33:894, 45:403, 45:455, 656
- selectfont 26:151
- seq commands:
 - \seq_clear:N 08:1579,
08:1661, 11:246, 11:305, 11:767
 - \seq_clear_new:N 08:1587, 08:1669
 - \seq_const_from_clist:Nn 11:56, 11:73
 - \seq_count:N 55:352
 - \seq_gclear_new:N 11:198
 - \seq_gpop:NN 08:2609
 - \seq_gpop:NNTF .. 52:72, 08:451, 08:459
 - \seq_gpop_right:NN 08:424
 - \seq_gpush:Nn
..... 08:2597, 08:2603, 52:64, 08:444
 - \seq_gput_right:Nn
..... 48:23, 08:102, 08:131, 08:413, 08:417
 - \seq_gset_eq:NN 11:199
 - \seq_if_empty:NNTF 08:412, 08:469
 - \seq_if_exist:NNTF 11:241, 52:60
 - \seq_if_in:NnTF 11:136,
11:149, 11:470, 11:505, 48:9, 48:248
 - \seq_map_break: 11:340, 11:431
 - \seq_map_function:NN
..... 11:321, 11:437, 11:1018
 - \seq_map_inline:Nn
..... 08:1603, 08:1621, 08:1686,
08:1709, 08:2990, 48:139, 48:172,
48:214, 48:225, 48:236, 48:390,
48:467, 07:2484, 08:1448, 08:1475
 - \seq_mapthread_function:NNN ... 202
 - \seq_new:N 08:1570,
08:2587, 10:36, 11:87, 11:89, 48:6,
52:61, 55:355, 07:2468, 08:28, 08:33
 - \seq_put_right:Nn
..... 08:1585, 08:1667, 08:1770, 08:1777,
10:91, 11:354, 11:515, 11:886, 11:894
 - \seq_remove_all:Nn ... 11:854, 11:868
 - \seq_set_eq:NN 11:243
 - \seq_set_split:Nnn ... 55:351, 07:2482
 - \seq_use:Nnnn . 08:1830, 08:1835, 10:65
- seq internal commands:
 - \g__mark_classes_seq
..... 48:6, 48:9, 48:23,
48:139, 48:172, 48:214, 48:225,
48:236, 48:248, 48:390, 48:467, 1074
 - \series 1358
 - \seriesdefault
.. 21:1602, 28:410, 28:460, 29:313,
29:315, 29:771, 29:804, 29:820,
29:836, 29:853, 29:930, 30:131, 745
 - \setattribute 41
 - \setattribute 04:82, 04:238
 - \setbox 918
 - \setbox... 1397
 - \setbox0 1394
 - \setcounter 556
 - \setcounter 20:464, 22:2,
22:59, 22:80, 22:99, 28:144, 39:256,
54:2991, 54:2994, 54:2997, 54:3001, 707
 - \SetCurrentKnownTemplateKeys 11:1624, 373
 - \SetDefaultHookLabel 08:2820, 225
 - \SetKeys 20:770, 51:285, 1151
 - \SetKnownTemplateKeys 11:1620, 373
 - \setlength 569
 - \setlength 18:84, 18:266, 18:281,
18:540, 23:4, 38:617, 38:622, 38:623,
38:624, 40:47, 40:62, 40:252, 40:271,
40:372, 40:375, 40:397, 40:400,
40:423, 40:426, 40:502, 40:609,
40:610, 40:611, 40:688, 40:689,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 40:696, 40:697, 40:698, 41:200,
41:461, 54:3007, 54:3008, 54:3009,
54:3012, 54:3013, 54:3017, 54:3018,
54:3019, 54:3023, 54:3024, 54:3025, 468
- \SetMathAlphabet 24:12,
27:140, 27:141, 28:767, 30:166, 30:167
- \setminus 30:404
- \SetProperty 36:38, 36:291, 853
- \setrangecatcode
..... 04:96, 04:104, 04:113, 04:114
- \SetSymbolFont
..... 28:602, 30:156, 30:157, 30:158, 1362
- \SetTemplateKeys 11:1620, 373
- \settodepth 569
- \settodepth 23:17
- \settoheight 569
- \settoheight 23:17
- \settowidth 569
- \settowidth 23:17
- \sf 1365
- \sfcode 18:485, 20:45,
20:115, 20:173, 57:250, 57:594,
02:306, 02:307, 02:308, 02:309, 02:426
- \sfdefault 29:10, 29:305, 29:478, 29:491,
29:519, 29:534, 29:564, 30:62, 740
- sffamily (hook) 244
- \sffamily .. 29:8, 29:9, 29:489, 29:532,
29:533, 29:562, 29:563, 32:16, 244
- sffamily 29:499
- \sfsubstdefault 30:19, 30:31, 33:30, 33:51
- \shape 1360
- \shapedefault 21:1602, 25:3204,
28:411, 28:461, 29:772, 29:805,
29:821, 29:837, 29:854, 30:131, 756
- \sharp 30:341
- shipout (hook) 1186, 1186, 1187, 1188, 1188
- \shipout 53:65, 53:113, 53:484,
53:487, 54:882, 54:957, 54:1015, 1196
- shipout 53:214
- shipout commands:
- \l_shipout_box
53:84, 53:96, 53:99, 53:111, 53:122,
53:187, 53:210, 53:240, 53:245,
53:268, 53:269, 53:276, 53:287,
53:290, 53:291, 53:295, 53:302,
53:312, 53:320, 53:327, 53:337,
53:343, 53:344, 53:347, 53:354,
53:356, 53:357, 53:363, 53:365, 1202
- \l_shipout_box_dp_dim
..... 53:255, 53:258,
53:260, 53:291, 53:344, 53:586, 1185
- \l_shipout_box_ht_dim
..... 53:254, 53:258, 53:260,
53:290, 53:310, 53:343, 53:585, 1185
- \l_shipout_box_ht_plus_dp_dim ...
..... 53:257, 53:260,
53:276, 53:327, 53:338, 53:340, 1185
- \l_shipout_box_wd_dim ... 53:256,
53:260, 53:302, 53:354, 53:587, 1185
- \shipout_debug_off:
..... 53:68, 53:74, 53:468, 1189
- \shipout_debug_on:
..... 53:68, 53:69, 53:467, 1189
- \shipout_discard:
..... 53:390, 53:390, 53:464, 1188
- \g_shipout_readonly_int
..... 36:256, 53:163, 53:165,
53:172, 53:177, 53:393, 53:402,
53:411, 53:415, 53:419, 53:423, 1189
- \g_shipout_totalpage_int 1189
- \g_shipout_totalpages_int
..... 53:148, 53:395, 1196
- shipout internal commands:
- __shipout_add_background_box:n .
53:241, 53:267, 53:267, 53:385, 53:473
- __shipout_add_background_-
picture:n
..... 53:130, 53:384, 53:384, 53:477
- __shipout_add_firstpage_-
material:Nn
53:234, 53:250, 53:250, 53:466, 53:471
- __shipout_add_firstpage_-
specials: 53:171,
53:227, 53:239, 53:239, 53:242, 1200
- __shipout_add_foreground_box:n .
53:178, 53:318, 53:318, 53:388, 53:475
- __shipout_add_foreground_-
picture:n
..... 53:125, 53:387, 53:387, 53:479
- __shipout_debug:n .. 53:68, 53:68,
53:81, 53:164, 53:176, 53:209, 1193
- \g__shipout_debug_bool
..... 53:67, 53:71, 53:76, 53:82
- __shipout_debug_gset:
..... 53:68, 53:72, 53:77, 53:79
- \g__shipout_discard_bool
53:142, 53:149, 53:151, 53:264, 53:391
- __shipout_drop_firstpage_-
specials: 53:188,
53:228, 53:239, 53:244, 53:246, 1200
- __shipout_excuse_extra_page: ...
..... 53:437, 53:445, 53:445
- __shipout_execute:
..... 53:107, 53:107, 53:113, 1195
- __shipout_execute_cont:
..... 53:118, 53:120, 53:120
- __shipout_execute_main_cont:Nnnn
53:121, 53:138, 53:138, 53:208, 1195

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltlyphen.dtx, 57=ltfinal.dtx

- \ShowTemplateCode [11:1600](#), [379](#)
- \ShowTemplateDefaults [11:1600](#), [379](#)
- \ShowTemplateInterface [11:1600](#), [379](#)
- \ShowTemplateVariables [11:1600](#), [379](#)
- \showtokens [06:752](#), [108](#)
- \Sigma [30:314](#)
- \sigma [30:295](#)
- \sim [30:456](#), [30:468](#)
- \simeq [30:457](#)
- \sin [38:9](#)
- \sinh [38:11](#)
- \size [1358](#)
- \skew [30:555](#)
- \skip [04:33](#), [40:527](#), [45:391](#), [02:28](#), [02:49](#),
[54:321](#), [54:617](#), [54:744](#), [02:132](#), [1234](#)
- skip commands:
 - \skip_eval:n [05:179](#)
 - \skip_new:N [11:96](#)
 - \skip_set:Nn [16:25](#)
 - \skip_zero:N [16:58](#), [53:283](#),
[53:284](#), [53:285](#), [53:334](#), [53:335](#), [53:336](#)
 - \l_tmpa_skip [373](#)
 - \c_zero_skip [16:26](#)
- \skipdef [04:228](#), [02:45](#), [02:49](#)
- \skipeval [77](#)
- \skipeval [05:172](#), [05:189](#), [77](#)
- \skipzero [04:228](#)
- \slash [06:982](#), [06:1003](#), [02:382](#)
- \sldefault [25:3211](#), [29:25](#), [30:106](#)
- \SLiTeX [1380](#)
- \sloppy [40:460](#), [40:479](#), [49:130](#), [49:135](#)
- sloppypar (env.) [49:135](#)
- \sloppypar [49:135](#)
- \slshape [21:461](#), [21:824](#), [25:3209](#),
[25:3210](#), [29:23](#), [29:24](#), [32:22](#), [33:591](#)
- \small [222](#)
- \smallbreak [06:983](#), [06:1004](#), [02:390](#)
- \smallint [30:368](#)
- \smallskip [18:469](#), [02:391](#)
- \smallskipamount .. [18:469](#), [18:472](#), [02:390](#)
- \smash [17:122](#),
[17:147](#), [17:170](#), [30:484](#), [30:558](#),
[30:559](#), [30:562](#), [30:563](#), [38:126](#), [892](#)
- \smile [30:461](#)
- socket commands:
 - \socket_assign_plug:nn
..... [10:42](#), [10:44](#), [10:122](#),
[10:122](#), [10:196](#), [55:13](#), [55:16](#), [359](#)
 - \socket_debug_off:
..... [10:7](#), [10:14](#), [10:199](#), [361](#)
 - \socket_debug_on: [10:7](#), [10:9](#), [10:198](#), [361](#)
 - \socket_get_plug:nN [10:144](#), [10:144](#), [361](#)
 - \socket_if_exist:n [10:53](#)
 - \socket_if_exist:nTF [10:28](#),
[10:53](#), [10:60](#), [10:79](#), [10:99](#), [10:123](#),
[10:146](#), [10:200](#), [10:201](#), [10:202](#), [360](#)
 - \socket_if_exist_p:n [10:53](#)
 - \socket_if_plug_assigned:nn .. [10:138](#)
 - \socket_if_plug_assigned:nnTF ...
.. [10:138](#), [10:206](#), [10:207](#), [10:208](#), [360](#)
 - \socket_if_plug_assigned_p:nn [10:138](#)
 - \socket_if_plug_exist:nn [10:117](#)
 - \socket_if_plug_exist:nnTF
..... [10:81](#), [10:101](#), [10:117](#),
[10:125](#), [10:203](#), [10:204](#), [10:205](#), [360](#)
 - \socket_if_plug_exist_p:nn .. [10:117](#)
 - \socket_log:n
..... [10:58](#), [10:58](#), [10:77](#), [10:194](#), [360](#)
 - \socket_new:nn
..... [10:27](#), [10:27](#), [10:192](#), [55:8](#), [359](#)
 - \socket_new_plug:nnn [10:38](#),
[10:41](#), [10:78](#), [10:78](#), [10:195](#), [55:14](#), [359](#)
 - \socket_set_plug:nnn
..... [10:78](#), [10:98](#), [10:185](#), [359](#)
 - \socket_show:n [10:58](#), [10:77](#), [10:193](#), [360](#)
 - \socket_use:n [10:154](#), [10:160](#), [360](#)
 - \socket_use:nn ... [10:154](#), [10:161](#), [360](#)
 - \socket_use:nnn .. [10:154](#), [10:162](#), [360](#)
 - \socket_use:nnnn .. [10:154](#), [10:163](#), [360](#)
 - \socket_use:nw
..... [10:154](#), [10:154](#), [10:160](#),
[10:161](#), [10:162](#), [10:163](#), [10:197](#), [360](#)
 - \socket_use_expandable:n
..... [10:164](#), [10:167](#), [1434](#)
 - \socket_use_expandable:nw
..... [10:164](#), [10:164](#), [10:167](#), [360](#)
- socket internal commands:
 - _socket_debug:n [10:7](#), [10:7](#), [10:21](#), [362](#)
 - \g_socket_debug_bool
..... [10:6](#), [10:11](#), [10:16](#), [10:22](#), [10:24](#)
 - _socket_debug_gset:
..... [10:7](#), [10:12](#), [10:17](#), [10:19](#)
 - _socket_debug_term:n
..... [10:7](#), [10:8](#), [10:23](#), [10:45](#),
[10:92](#), [10:108](#), [10:127](#), [10:155](#), [362](#)
- Sockets:
 - build/column/baselineattach
..... [54:723](#),
[1233](#), [1234](#), [1246](#), [1247](#), [1250](#), [1250](#)
 - build/column/footins [1234](#)
 - build/column/footnotes .. [54:722](#), [1234](#)
 - build/column/outputbox
..... [54:663](#), [1233](#), [1234](#), [1246](#)
 - build/page/footer [1234](#)
 - build/page/header [1234](#)
 - foo [356](#), [356](#)
 - refstepcounter [35:134](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- refstepcounter/target 35:135
- \sourceLaTeXdate 08:2966, 03:164, 50:82, 50:120
- \sp 38:212
- \space 02:322, 1376
- space:floats:footnotes (plug) 54:688, 1233
- space:footnotes:floats (plug) 54:664, 1233
- \spacefactor 17:121, 17:123,
17:146, 17:148, 17:169, 17:171,
18:130, 18:139, 18:162, 18:180,
18:194, 18:206, 18:220, 18:234,
18:485, 18:526, 18:531, 21:88, 21:91,
45:593, 45:595, 02:380, 02:381, 1401
- \spacefactor_in_math_mode_gh/643 . 1424
- \spaceskip 33:6, 33:576, 02:296
- \spadesuit 30:345
- \span 41:460
- \special 17:63, 17:144, 17:167, 1197
- \SplitArgument 07:3316, 123
- \splitfirstmark 54:2915
- \SplitList 07:3316, 123
- \splitmaxdepth
45:531, 45:550, 45:568, 54:2909, 02:269
- \splittopskip 45:530, 45:549, 45:567, 02:294
- \sqcap 30:387
- \sqcup 30:388
- \sqrt 38:413, 1389
- \sqrtsign 30:540, 38:71, 38:413, 1389
- \sqsubset 29:885
- \sqsubseteq 30:410
- \sqsupset 29:886
- \sqsupseteq 30:411
- \SS .. 21:324, 21:557, 21:1185, 57:720, 1394
- \ss 21:274,
21:427, 21:580, 21:808, 21:1160, 57:720
- \sscddefault 25:2924, 25:2940, 25:3226
- \ssshape 25:2924,
25:2939, 25:3224, 25:3225, 32:31, 664
- \stackrel 38:411
- \stamp 1368
- \star 30:408
- \stepcounter 556
- \stepcounter 22:34, 22:44, 24:698,
28:49, 35:143, 35:156, 35:166,
35:177, 35:186, 35:193, 38:425,
38:454, 38:516, 38:610, 45:514,
45:582, 54:943, 54:1001, 54:1060, 1400
- \stockheight 54:71
- \stockwidth 54:71
- \stop 33:1600, 37:242
- \storedpar 04:156, 04:161
- str commands:
- \c_backslash_str
. 09:163, 09:432, 07:1639, 07:1646,
07:1662, 07:1666, 07:1717, 07:2841
- \c_hash_str 09:515, 09:519,
09:543, 07:1424, 07:1556, 08:568, 08:574
- \c_right_brace_str 08:2088
- \str_case:nn
..... 09:629, 11:666, 57:611, 07:1159
- \str_case:nnTF
..... 09:56, 36:29, 51:30, 07:3088
- \str_case_e:nnTF 07:2732
- \str_count:n 08:1885, 09:177,
09:252, 07:2824, 08:37, 08:37, 08:1271
- \str_gset:Nn 08:2829
- \str_head:n 07:2677
- \str_if_empty:NTF 57:637
- \str_if_eq:nn 287
- \str_if_eq:NNTF 52:555
- \str_if_eq:nnTF . 08:1823, 08:1941,
08:2041, 08:2148, 08:2360, 08:2564,
08:2618, 08:2688, 09:231, 09:303,
09:402, 09:554, 09:561, 09:590,
09:593, 10:140, 11:162, 11:175,
11:334, 11:355, 11:720, 11:776,
11:1220, 05:151, 52:131, 52:549,
52:559, 07:104, 07:106, 07:428,
07:1238, 07:1243, 07:1738, 07:1792,
07:1795, 07:1798, 07:1801, 07:1860,
07:2279, 07:2290, 07:2292, 07:2320,
07:2338, 07:2340, 07:2357, 07:2384,
07:2410, 07:2412, 07:2433, 07:2435,
07:2456, 07:2573, 07:2585, 07:2613,
07:2789, 08:340, 08:438, 08:484,
08:541, 08:543, 08:630, 08:632,
08:762, 08:820, 08:853, 08:906,
08:988, 08:995, 08:1023, 08:1030, 193
- \str_if_eq_p:nn
. 08:1614, 08:1700, 11:568, 11:569,
11:1036, 11:1037, 11:1038, 07:489,
07:490, 07:491, 07:492, 07:768,
07:964, 07:969, 07:2621, 07:2643, 08:910
- \str_if_exist:NTF 10:54
- \str_if_in:nnTF 11:532
- \str_lowercase:n 57:649
- \str_map_function:NN 05:158
- \str_new:N 10:35, 11:85, 07:22
- \str_replace_all:Nnn 09:542
- \str_set:Nn 10:130, 11:815,
52:399, 52:400, 07:232, 07:246,
07:247, 07:291, 57:627, 57:635, 57:639
- \str_tail:n
..... 07:2279, 07:2674, 07:2700, 190

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\str_use:N` 10:67,
 10:70, 10:128, 10:157, 10:158, 10:165
 str internal commands:
 `\__str_if_eq:nn` 08:23, 246
 `\strcmp` 50:381, 50:397
 `\stretch` 18:554
 `\string` 1393
 `\stripmeaning` 1365
 `\strongem` ... 29:684, 29:685, 29:705, 32:43
 `\strongemforce` 756
 `\strongemforce` 29:681, 29:702, 756
 `\strongemph` 756
 `\strongemph` 29:705, 32:43, 758
 `\strongemreset` 756
 `\strongemreset` 29:681, 29:706, 57:786, 756
 `struct:mc` (tag socket) 55:62
 `struct/begin` (tag socket) 55:72
 `struct/end` (tag socket) 55:72
 `\strut` 38:191, 38:192,
 41:29, 06:984, 06:1005, 02:398, 245
 `\strutbox` 26:190, 40:553,
 40:570, 40:587, 41:238, 41:239,
 41:261, 41:262, 45:531, 45:539,
 45:550, 45:557, 45:568, 45:575, 02:398
 `\subencodingresult` .. 33:1200, 33:1205,
 33:1374, 33:1378, 33:1380, 33:1381
 `\subparagraphmark` 44:143
 `\subsectionmark` 44:143, 1064
 `\subset` 30:437
 `\subseteq` 30:439
 `\substdefault` 1414
 `\subsubsectionmark` 44:143
 `\succ` 30:431
 `\succeq` 30:434
 `\sum` 30:361
 `\sup` 38:24
 `\supereject` 1393
 `\suppressfloats` 54:2583
 `\supset` 30:436
 `\supseteq` 30:438
 `\surd` 30:347
 `\SuspendTagging` 1309
 `\SuspendTagging` 23:22, 55:3, 1310
 `\swarrow` 30:419
 `\swdefault` 25:2919, 25:2938, 25:3223
 `\swshape` 25:2919,
 25:2937, 25:3221, 25:3222, 32:30
 `\symbol` 21:180, 29:730
 `\symletters` ... 33:8, 33:14, 33:578, 33:1100
 `\symoperators` 30:654
 sys commands:
 `\c_sys_day_int` 11:1578
 `\sys_if_engine luatex:TF`
 09:381, 53:13, 53:87
 `\c_sys_month_int` 11:1577
 `\c_sys_year_int` 11:1576

 T
 `\T` 14:23, 21:355,
 21:357, 21:359, 21:361, 21:363,
 21:365, 21:367, 21:369, 21:371,
 21:394, 50:1615, 50:1619, 50:1620
 `\t` 21:302, 21:765, 21:873, 33:121,
 33:122, 33:149, 33:153, 33:155,
 33:167, 33:170, 33:172, 33:652,
 33:820, 33:1087, 33:1089, 33:1282, 1382
 `tabbing` (env.) 41:71
 `\tabbing` 41:71
 `\tabbingsep` 41:130, 41:132, 41:166
 `\tabcolsep` 41:336, 41:415
 `\tableofcontents` 870
 `\tabskip` 38:208, 38:209, 38:431,
 38:434, 38:437, 38:439, 38:460,
 38:463, 38:466, 38:468, 38:615,
 38:628, 38:631, 38:633, 41:167,
 41:244, 41:267, 02:295, 02:409, 1362
 `tabular` (env.) 41:198
 `\tabular` 41:198
 `\tabular*` 41:199
 `\tabularnewline` 41:246, 41:269, 41:284, 1386
 tag commands:
 `\tag_if_active:` 55:26
 `\tag_if_active:TF` 55:357, 1434
 `\tag_mc_begin:n`
 55:57, 55:66, 55:89, 55:157
 `\tag_mc_begin_pop:n` 55:97
 `\tag_mc_end:` 55:59, 55:68, 55:95, 55:164
 `\tag_mc_end_push:` 55:87
 `\c_tag_para_semantic_tl` 55:110
 `\c_tag_para_textblock_tl` 55:111
 `\tag_resume:n` 55:3, 55:4, 55:6
 `\tag_socket_use:n` .. 55:26, 55:29, 1310
 `\tag_socket_use:nn` . 55:26, 55:30, 1310
 `\tag_socket_use:nnn` 55:26, 55:31, 1310
 `\tag_socket_use_expandable:n` ...
 55:26, 55:28, 1310
 `\tag_struct_begin:n`
 55:65, 55:76, 55:88, 55:130, 55:151
 `\tag_struct_end:`
 55:69, 55:80, 55:96, 55:141, 55:166
 `\tag_struct_recordtarget:` ... 55:210
 `\tag_suspend:n` 55:3, 55:3, 55:5
 tag internal commands:
 `\l__tag_block_flattened_level_` -
 `int` 55:100, 55:113
 `\__tag_check_para_begin_show:nn` .
 55:156
 `\__tag_check_para_end_show:nn` 55:165

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

<code>__tag_check_typeout_v:n</code>	55:150 , 55:163 , 55:176	<code>math/luamml/array/initcol</code> ...	55:286
<code>__tag_gincr_para_begin_int:</code>	55:149	<code>math/luamml/array/save</code>	55:284
<code>__tag_gincr_para_end_int:</code> ..	55:162	<code>math/luamml/artifact</code>	55:299
<code>__tag_gincr_para_main_begin-</code>		<code>math/luamml/finph@nt</code> .	55:297 , 55:298
<code>int:</code>	55:129	<code>math/luamml/hbox</code>	55:296
<code>__tag_gincr_para_main_end_int:</code> .		<code>math/luamml/mtable/aligncol</code> .	55:290
.....	55:140	<code>math/luamml/mtable/finalize</code> .	55:289
<code>__tag_lastpage_label:</code>	53:407	<code>math/luamml/mtable/finalizecol</code>	55:288
<code>\l__tag_para_attr_class_tl</code>	55:126 , 55:154	<code>math/luamml/mtable/innertable/finalize</code>	55:293
<code>\l__tag_para_bool</code>			55:293
55:102 , 55:104 , 55:114 , 55:172 , 55:192		<code>math/luamml/mtable/innertable/save</code>	55:291
<code>\l__tag_para_flattened_bool</code>	55:112 , 55:180 , 55:195		55:292
<code>\l__tag_para_main_attr_class_tl</code> .		<code>math/luamml/mtable/smallmatrix/save</code>	55:292
.....	55:133		55:294
<code>\tag_resume:n</code>	1309	<code>math/luamml/mtable/tag/save</code> .	55:294
<code>\tag_socket_use:n</code>	1309	<code>math/luamml/mtable/tag/set</code> ..	55:295
<code>\tag_socket_use:nn</code>	1309	<code>math/luamml/save/nn</code>	55:281
<code>\tag_socket_use:nnn</code>	1309	<code>math/luamml/save/nNn</code>	55:282
<code>\tag_socket_use_expandable:n</code>	1310	<code>mc</code>	55:54
<code>\tag_suspend:n</code>	1309	<code>page@sofar</code>	55:270
Tagging Plugs:		<code>para/begin</code>	55:117
<code>default</code>	55:108	<code>para/end</code>	55:117
<code>kernel (para/begin)</code>	55:170	<code>para/off</code>	55:101
<code>kernel (para/end)</code>	55:190	<code>para/on</code>	55:101
<code>kernel (para/semantic/begin)</code>	55:126	<code>para/restore</code>	55:107
<code>kernel (para/semantic/end)</code> ..	55:137	<code>para/semantic/begin</code>	55:124
<code>kernel (para/textblock/begin)</code>	55:147	<code>para/semantic/end</code>	55:124
<code>kernel (para/textblock/end)</code> .	55:159	<code>para/textblock/begin</code>	55:145
<code>kernel (recordtarget)</code>	55:207	<code>para/textblock/end</code>	55:145
Tagging Sockets:		<code>refstepcounter</code>	55:206
<code>block/list/label</code>	55:218	<code>sec/begin</code>	55:220
<code>block/recipe</code>	55:219	<code>sec/end</code>	55:220
<code>build/column/footins</code>	55:269	<code>sec/title/begin</code>	55:222
<code>build/column/outputbox</code>	55:268	<code>sec/title/end</code>	55:222
<code>build/page/footer</code>	55:266	<code>sec/title/hang</code>	55:224
<code>build/page/header</code>	55:266	<code>sec/title/init</code>	55:225
<code>caption/begin</code>	55:262	<code>sec/title/number</code>	55:228
<code>caption/end</code>	55:262	<code>sec/title/runin/number</code>	55:225
<code>caption/label/begin</code>	55:264	<code>sec/title/split</code>	55:225
<code>caption/label/end</code>	55:264	<code>struct:mc</code>	55:62
<code>float/begin</code>	55:260	<code>struct/begin</code>	55:72
<code>float/end</code>	55:260	<code>struct/end</code>	55:72
<code>float/hmode/begin</code>	55:258	<code>tbl/cell/begin</code>	55:237
<code>float/hmode/end</code>	55:258	<code>tbl/cell/end</code>	55:237
<code>inline/begin</code>	55:84	<code>tbl/colspan</code>	55:247
<code>inline/end</code>	55:84	<code>tbl/finalize</code>	55:245
<code>marginpar/begin</code>	55:235	<code>tbl/hmode/begin</code>	55:248
<code>marginpar/end</code>	55:235	<code>tbl/hmode/end</code>	55:248
<code>math/luamml/annotate/false</code> ..	55:283	<code>tbl/init</code>	55:243
<code>math/luamml/array/finalize</code> ..	55:285	<code>tbl/init/celldata</code>	55:244
<code>math/luamml/array/finalizecol</code>	55:287	<code>tbl/leaders/begin</code>	55:256
		<code>tbl/leaders/end</code>	55:256
		<code>tbl/longtable/finalize</code>	55:252
		<code>tbl/longtable/foot</code>	55:254

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

tbl/longtable/head	55:254	\g__tbl_row_int	55:322,
tbl/longtable/init	55:252	55:379, 55:394, 55:399, 55:413,	
tbl/pcell/end	55:237, 55:237	55:416, 55:419, 55:426, 55:431, 1324	
tbl/restore/celldata	55:246	\l__tbl_saved_col_tl	55:329, 55:378, 55:385, 55:425
tbl/row/begin	55:237	\l__tbl_saved_row_tl	55:329, 55:379, 55:384, 55:426
tbl/row/end	55:237	\l__tbl_saved_span_tl	55:329, 55:381, 55:386, 55:427, 55:433
tbl/vmode/begin	55:248	\l__tbl_saved_table_cols_tl	55:329, 55:380, 55:388, 55:390, 55:428
tbl/vmode/end	55:248	\g__tbl_span_tl	55:322, 55:347,
toc/contentsline/after	55:229	55:348, 55:364, 55:381, 55:396,	
toc/contentsline/before	55:229	55:404, 55:427, 55:448, 55:450, 1326	
toc/leaders/after	55:233	\g__tbl_table_cols_tl	55:322, 55:352, 55:353, 55:362,
toc/leaders/before	55:233	55:380, 55:428, 55:435, 55:437, 1325	
toc/starttoc/after	55:231	\l__tbl_tmpa_seq	55:351, 55:352, 55:355
toc/starttoc/before	55:231	__tbl_trace:n	55:340, 55:343,
tagpdf (plug)	359	55:345, 55:353, 55:368, 55:383, 55:430	
\tan	38:15	tbl/cell/begin (hook)	41:469
\tanh	38:17	tbl/cell/begin (tag socket)	55:237
target	36:262, 854	tbl/cell/end (hook)	41:469
\tau	30:296	tbl/cell/end (tag socket)	55:237
tbl commands:		tbl/cell/init (hook)	41:468
\tbl_count_missing_cells:n	55:356, 55:356, 55:455, 1328	tbl/colspan (tag socket)	55:247
\tbl_count_table_cols:	55:350, 55:350, 1325	tbl/finalize (tag socket)	55:245
\tbl_crcr:n	55:452, 55:452	tbl/hmode/begin (tag socket)	55:248
\tbl_gdecr_row_count:	55:412, 55:418	tbl/hmode/end (tag socket)	55:248
\tbl_gincr_row_count:	55:412, 55:415	tbl/init (hook)	41:465
\tbl_gzero_row_count:	55:412, 55:412	tbl/init (tag socket)	55:243
\tbl_if_row_was_started:TF	55:406, 55:406, 55:409	tbl/init/celldata (tag socket)	55:244
\tbl_inbetween_rows:	55:421, 55:421	tbl/leaders/begin (tag socket)	55:256
\tbl_init_cell_data_for_row:	55:402, 55:402	tbl/leaders/end (tag socket)	55:256
\tbl_init_cell_data_for_table:	55:377, 55:377	tbl/longtable/finalize (tag socket)	55:252
\tbl_restore_outer_cell_data:	55:424, 55:424, 1319	tbl/longtable/foot (tag socket)	55:254
\tbl_update_cell_data:	55:346, 55:346	tbl/longtable/head (tag socket)	55:254
\tbl_update_cell_data_for_next_row:	55:398, 55:398	tbl/longtable/init (tag socket)	55:252
\tbl_update_multicolumn_cell_data:n	55:441, 55:441	tbl/pcell/end (tag socket)	55:237, 55:237
tbl internal commands:		tbl/restore/celldata (tag socket)	55:246
\g__tbl_col_int	55:322, 55:347, 55:358, 55:363, 55:378, 55:395, 55:400, 55:403, 55:407, 55:410, 55:422, 55:425, 55:432, 55:442, 55:445, 55:448, 55:453, 1328	tbl/row/begin (hook)	41:467
\g__tbl_missing_cells_int	55:338, 55:360, 55:367, 55:371	tbl/row/begin (tag socket)	55:237
\g__tbl_missingcells_int	55:338	tbl/row/end (hook)	41:467
		tbl/row/end (tag socket)	55:237
		tbl/row/init (hook)	41:466
		tbl/vmode/begin (tag socket)	55:248
		tbl/vmode/end (tag socket)	55:248
		template internal commands:	
		__template_assign_boolean:	11:1081, 11:1081
		__template_assign_boolean_aux:n	11:1081, 11:1084, 11:1085, 11:1087
		__template_assign_choice:	11:1107, 11:1107

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

```

__template_assign_choice_-
  aux:nTF ..... 11:1107,
    11:1109, 11:1112, 11:1124, 11:1129
__template_assign_function: ...
  ..... 11:1130, 11:1130
__template_assign_function_-
  aux:N .....
  .. 11:1130, 11:1133, 11:1134, 11:1136
__template_assign_instance: ...
  ..... 11:1147, 11:1147
__template_assign_instance_-
  aux:N .....
  .. 11:1147, 11:1150, 11:1151, 11:1153
__template_assign_variable: ...
  ..... 11:1042, 11:1165, 11:1165
__template_assign_variable:n ...
  ..... 11:1165, 11:1167, 11:1175
__template_assignments_pop: ...
  ..... 11:1236, 11:1236, 11:1619
__template_assignments_push:n ..
  ..... 11:978, 11:1237, 11:1237
\l__template_assignments_tl 11:76,
  11:979, 11:1017, 11:1092, 11:1100,
  11:1127, 11:1138, 11:1155, 11:1180,
  11:1187, 11:1236, 11:1238, 11:1261, 382
\c__template_code_root_tl .....
  ..... 11:48, 11:130,
    11:551, 11:554, 11:949, 11:951,
    11:952, 11:980, 11:1078, 11:1265, 381
__template_convert_to_assignments:N
  ..... 11:971,
    11:1015, 11:1015, 11:1073, 11:1260
__template_convert_to_assignments_-
  aux:n .... 11:1015, 11:1019, 11:1021
__template_convert_to_assignments_-
  aux:nn .....
  .. 11:1015, 11:1024, 11:1026, 11:1047
\l__template_current_template_tl
  . 11:99, 11:977, 11:1077, 11:1628, 383
\l__template_current_type_tl ...
  . 11:98, 11:976, 11:1076, 11:1627, 383
__template_date_compare:nNn .. 11:9
__template_date_compare:nNnTF ..
  ..... 11:9, 11:1574
__template_date_compare_aux:nNn
  ..... 11:9, 11:10, 11:11
__template_date_compare_-
  auxi:nnnNnn .... 11:9, 11:15, 11:21
__template_date_compare_auxii:w
  ..... 11:9, 11:16, 11:18, 11:40
__template_date_compare_-
  auxiii:w ..... 11:9, 11:43, 11:46
__template_debug:n .....
  ..... 11:107, 11:113, 11:116, 11:116

__template_debug_off: .....
  ..... 11:105, 11:111, 11:1633
__template_debug_on: .....
  ..... 11:105, 11:105, 11:1632
__template_debug_typeout:n ....
  ..... 11:108,
    11:114, 11:116, 11:117, 11:1074, 11:1232
__template_declare_instance:nnnn
  ..... 11:954, 11:954, 11:1562
__template_declare_instance_-
  aux:nnnn .....
  ..... 11:954, 11:960, 11:963, 11:1012
__template_declare_template_-
  code:nnnn .....
    11:523, 11:535, 11:537, 11:542, 11:549
__template_declare_template_-
  code:nnnnn . 11:523, 11:523, 11:1554
__template_declare_template_-
  equiv:nnnn ..... 11:443,
    11:443, 11:1572, 11:1583, 11:1591
__template_declare_template_-
  keys:nnnn .. 11:296, 11:296, 11:1552
__template_declare_type:nn ....
  ..... 11:275, 11:279, 11:281
\l__template_default_tl .. 11:77, 382
\c__template_defaults_root_tl ...
  ..... 11:49,
    11:182, 11:183, 11:225, 11:228, 381
__template_define_type:nn .....
  ..... 11:275, 11:275, 11:1550
__template_edit_defaults:nnn ...
  ..... 11:751, 11:751, 11:1558
__template_edit_instance:nnn ...
  ..... 11:997, 11:997, 11:1566
__template_edit_instance_-
  aux:nnnn . 11:1004, 11:1009, 11:1014
__template_edit_instance_-
  aux:nnnnn ..... 11:997
\l__template_error_bool .....
  ..... 11:78, 11:317,
    11:323, 11:339, 11:368, 11:381,
    11:465, 11:781, 11:965, 11:967, 382
__template_execute_if_arg_-
  agree:nnTF .....
  ..... 11:118, 11:118, 11:300, 11:527
__template_execute_if_code_-
  exist:nnTF ..... 11:128,
    11:128, 11:956, 11:1068, 11:1286
__template_execute_if_keyname_-
  free:nTF .....
  ..... 11:147, 11:147, 11:153, 11:325
__template_execute_if_keys_-
  exist:nnTF ..... 11:154

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

_template_execute_if_keytype_	11:474, 11:513, 11:515, 11:516,
exist:nTF	11:519, 11:520, 11:580, 11:583,
..... 11:134, 11:134, 11:140, 11:319	11:588, 11:645, 11:657, 11:662,
_template_execute_if_type_	11:681, 11:686, 11:698, 11:702,
exist:nTF	11:705, 11:708, 11:711, 11:712,
..... 11:141, 11:141, 11:298, 11:445, 11:525	11:735, 11:738, 11:743, 11:749,
_template_find_global:	11:797, 11:814, 11:823, 11:834,
..... 11:1040, 11:1048, 11:1048	11:839, 11:841, 11:842, 11:846,
_template_find_global_aux:w ...	11:847, 11:854, 11:855, 11:857,
..... 11:1048, 11:1053, 11:1056	11:862, 11:868, 11:869, 11:878,
\l_template_global_bool	11:886, 11:887, 11:894, 11:895,
..... 11:79, 11:1050, 11:1059,	11:922, 11:928, 11:933, 11:1032,
..... 11:1083, 11:1132, 11:1149, 11:1171, 382	11:1110, 11:1113, 11:1115, 11:1119, 400
_template_if_instance_exist:nn	\c_template_key_order_root_tl ..
..... 11:167	 11:52,
_template_if_instance_exist:nnTF	 11:198, 11:199, 11:241, 11:244, 381
..... 11:167,	\l_template_key_order_seq
..... 11:986, 11:999, 11:1063, 11:1226,	 11:87, 11:149, 11:200,
..... 11:1303, 11:1611, 11:1613, 11:1615	 11:243, 11:246, 11:305, 11:354,
_template_if_key_value:n	 11:505, 11:515, 11:971, 11:1073, 382
..... 11:160, 11:166	_template_key_to_value:
_template_if_key_value:nTF ...	.. 11:1091, 11:1179, 11:1194, 11:1194
..... 11:160, 11:1089, 11:1177	_template_key_to_value_auxi:w .
_template_if_keys_exist:nnTF ..	 11:1194, 11:1195, 11:1196
..... 11:154,	_template_key_to_value_auxii:w
..... 11:447, 11:529, 11:753, 11:1268, 11:1277	 11:1194, 11:1202, 11:1207
_template_if_use_template:n 11:173	_template_keys_set:nnn
_template_if_use_template:nTF .	 11:1239, 11:1239, 11:1623
..... 11:173, 11:1214	_template_keys_set_known:nnn ..
_template_implement_choice_	 11:1239,
elt:n	 11:1241, 11:1248, 11:1621, 11:1626
_template_implement_choice_	_template_keys_set_known_
elt:nnn 11:684, 11:716, 11:716	aux:nnn .. 11:1239, 11:1249, 11:1250
_template_implement_choice_	\l_template_keytype_arg_tl
elt_aux:n	 11:82, 11:336,
..... 11:716, 11:722, 11:730, 11:733	 11:349, 11:350, 11:357, 11:414,
_template_implement_choice_	 11:428, 11:680, 11:1143, 11:1160, 382
elt_aux:nnn	\l_template_keytype_tl ... 11:81,
..... 11:716, 11:721, 11:728, 11:740	 11:319, 11:334, 11:348, 11:355,
_template_implement_choices:nn	 11:413, 11:424, 11:484, 11:487,
..... 11:651, 11:678, 11:678	 11:514, 11:584, 11:586, 11:613,
_template_implement_choices_	 11:615, 11:666, 11:821, 11:826,
default: ... 11:678, 11:688, 11:695	 11:863, 11:898, 11:911, 11:914,
_template_instance_set_eq:nnn .	 11:1036, 11:1037, 11:1038, 11:1041, 382
..... 11:984, 11:984, 11:1564	\c_template_keytypes_arg_seq ...
_template_instance_value:nnn ..	 11:73, 11:321, 11:437, 381
..... 11:1061, 11:1061, 11:1618	\l_template_keytypes_prop
\c_template_instances_root_tl ..	... 11:86, 11:197, 11:237, 11:240,
... 11:50, 11:169, 11:972, 11:974,	 11:303, 11:352, 11:467, 11:513,
..... 11:990, 11:992, 11:993, 11:1234, 381	 11:565, 11:582, 11:588, 11:705,
\l_template_key_name_tl	 11:735, 11:775, 11:797, 11:842,
..... 11:80, 11:326,	 11:862, 11:1023, 11:1115, 11:1280, 382
..... 11:352, 11:354, 11:364, 11:375,	\c_template_keytypes_root_tl ...
..... 11:390, 11:397, 11:402, 11:468,	... 11:51, 11:156, 11:190, 11:193,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

11:195, 11:196, 11:235, 11:238, 381
 \c__template_keytypes_seq
 11:56, 11:136, 11:470, 381
 __template_map_var_type:
 11:643, 11:664, 11:664, 11:918, 11:1169
 \l__template_msg_prop
 11:91, 11:216,
 11:263, 11:266, 11:519, 11:823, 382
 \c__template_msg_root_tl .. 11:53,
 11:214, 11:215, 11:261, 11:264, 381
 __template_parse_equiv_alias:nnn
 11:492
 __template_parse_equiv_alias:nnnn
 11:462
 __template_parse_equiv_auxi:nnnn
 11:462, 11:493, 11:495, 11:496
 __template_parse_equiv_auxii:n .
 11:462, 11:499, 11:500, 11:503
 __template_parse_equiv_auxiii:nnnn
 11:462, 11:501, 11:511
 __template_parse_equiv_elt:n ...
 11:452, 11:460, 11:460
 __template_parse_equiv_elt:nnnn
 11:453, 11:462, 11:462
 __template_parse_equiv_elt_
 aux:nn 11:462, 11:476, 11:478, 11:481
 __template_parse_equiv_meta:nnn
 11:494
 __template_parse_equiv_meta:nnnn
 11:462
 __template_parse_keys_elt:n ...
 .. 11:307, 11:314, 11:314, 11:363, 390
 __template_parse_keys_elt:nn ...
 11:307, 11:361, 11:361
 __template_parse_keys_elt_aux: .
 11:314, 11:327, 11:344
 __template_parse_keys_elt_aux:n
 11:314, 11:322, 11:332
 __template_parse_known_values:nnn
 11:760, 11:762, 11:1259
 __template_parse_unknown_
 error:nn ... 11:761, 11:935, 11:937
 __template_parse_unknown_
 store:nn ... 11:763, 11:938, 11:940
 __template_parse_values:nnn ...
 11:756, 11:760, 11:760, 11:966, 11:1072
 __template_parse_values:NNnnn ..
 11:760, 11:761, 11:763, 11:764
 __template_parse_values_add:N ..
 11:787, 11:901, 11:902
 __template_parse_values_add:n ..
 11:787, 11:892, 11:900, 11:901
 __template_parse_values_add_
 commalist:n 11:925
 __template_parse_values_add_
 tokenlist:n 11:930
 __template_parse_values_
 alias:nn 11:787, 11:852
 __template_parse_values_alias_
 aux:nn 11:787, 11:858, 11:860, 11:881
 __template_parse_values_elt:NNn
 11:769, 11:773, 11:773
 __template_parse_values_elt:Nnn
 11:770, 11:777, 11:787, 11:787
 __template_parse_values_elt_
 auxi:n 11:787
 __template_parse_values_elt_
 auxi:w 11:787, 11:791, 11:810
 __template_parse_values_elt_
 auxii:nn
 11:799, 11:817, 11:844, 11:864
 __template_parse_values_elt_
 auxiii:Nn .. 11:787, 11:800, 11:831
 __template_parse_values_elt_
 auxiv:w 11:787, 11:833, 11:836
 __template_parse_values_meta:nn
 11:787, 11:866
 __template_parse_values_meta_
 elt:nnnn ... 11:787, 11:873, 11:876
 __template_parse_values_set:N ..
 11:787, 11:890, 11:891
 __template_parse_values_set:n ..
 11:787, 11:884, 11:889, 11:890
 __template_parse_vars_choice:nnn
 11:650, 11:650
 __template_parse_vars_elt:n ...
 11:563, 11:576, 11:576
 __template_parse_vars_elt:nnn ..
 11:563, 11:578, 11:578
 __template_parse_vars_elt_
 auxi:nn 11:578, 11:587, 11:592
 __template_parse_vars_elt_
 auxii:nw ... 11:578, 11:594, 11:596
 __template_parse_vars_elt_
 auxiii:nn
 11:578, 11:600, 11:604, 11:611, 11:623
 __template_parse_vars_elt_
 auxiv:nnw .. 11:578, 11:619, 11:624
 __template_parse_vars_elt_
 auxv:nnn
 11:578, 11:628, 11:632, 11:638, 11:649
 __template_parse_vars_function:nnn
 11:650, 11:652
 __template_parse_vars_instance:nnn
 11:650, 11:659
 __template_quark_if_nil:N .. 11:104
 __template_quark_if_nil:NTF 11:1209
 __template_quark_if_nil:nTF 11:104

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=ltlooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx


```

\__template_quark_if_nil_p:n 11:104
\__template_recover_defaults:nn .
    . . . . . 11:222, 11:222, 11:559,
    11:755, 11:943, 11:958, 11:1070, 11:1270
\__template_recover_keytypes:nn .
    . . . . . 11:222, 11:232,
    11:449, 11:560, 11:766, 11:945, 11:1279
\__template_recover_values:nn . . .
    . . . . . 11:222,
    11:248, 11:988, 11:1001, 11:1305
\__template_recover_vars:nn . . . .
    . . . . . 11:222, 11:258,
    11:450, 11:561, 11:947, 11:959,
    11:1011, 11:1071, 11:1258, 11:1288
\c__template_restrict_root_tl . . 381
\__template_show:Nnnn . . 11:1266,
    11:1271, 11:1280, 11:1289, 11:1293
\__template_show_code:nn . . . . .
    . . . . . 11:1264, 11:1264, 11:1601
\__template_show_defaults:nn . . .
    . . . . . 11:1266, 11:1266, 11:1603
\__template_show_keytypes:nn . . .
    . . . . . 11:1266, 11:1275, 11:1605
\__template_show_values:nn . . . .
    . . . . . 11:1301, 11:1301, 11:1609
\__template_show_vars:nn . . . . .
    . . . . . 11:1266, 11:1284, 11:1607
\__template_split_keytype:n . . . .
    . . . . . 11:316, 11:366, 11:366, 11:464
\__template_split_keytype_arg:n .
    . . . . . 11:407,
    11:411, 11:411, 11:440, 11:586,
    11:707, 11:737, 11:819, 11:1033, 11:1117
\__template_split_keytype_arg_-
    aux:n . 11:411, 11:415, 11:438, 11:441
\__template_split_keytype_arg_-
    aux:w . 11:411, 11:419, 11:434, 11:442
\__template_split_keytype_aux:w .
    . . . . . 11:366, 11:376, 11:387, 11:399
\__template_store_defaults:nn . . .
    11:179, 11:179, 11:308, 11:757, 11:944
\__template_store_key_implementation:nnn
    . . . . . 11:531, 11:557, 11:557
\__template_store_keytypes:nn . . .
    11:179, 11:187, 11:309, 11:455, 11:946
\__template_store_values:nn . . . .
    . . . . . 11:179, 11:203, 11:970, 11:989
\__template_store_vars:nn 11:179,
    11:211, 11:310, 11:456, 11:564, 11:948
\__template_template_set_eq:nnn .
    . . . . . 11:941, 11:941, 11:1556
\__template_tmp:n . . . . .
    . . . . . 11:100, 11:100, 11:879, 11:880

\l__template_tmp_clist . . . . .
    . . . . . 11:92, 11:680, 11:689,
    11:691, 11:718, 11:725, 11:727, 383
\l__template_tmp_dim . . . . 11:93, 383
\l__template_tmp_int 11:94, 11:283,
    11:284, 11:287, 11:289, 11:293, 383
\l__template_tmp_muskip . . 11:95, 383
\l__template_tmp_skip . . . 11:96, 383
\l__template_tmp_tl . . . . . 11:97,
    11:120, 11:121, 11:125, 11:346,
    11:353, 11:369, 11:370, 11:371,
    11:377, 11:468, 11:471, 11:687,
    11:697, 11:698, 11:699, 11:701,
    11:702, 11:703, 11:706, 11:707,
    11:709, 11:736, 11:737, 11:742,
    11:744, 11:775, 11:776, 11:798,
    11:819, 11:824, 11:825, 11:843,
    11:856, 11:857, 11:870, 11:874,
    11:880, 11:882, 11:896, 11:897,
    11:916, 11:919, 11:922, 11:927,
    11:928, 11:932, 11:933, 11:1003,
    11:1005, 11:1023, 11:1024, 11:1116,
    11:1117, 11:1126, 11:1127, 11:1198,
    11:1199, 11:1205, 11:1257, 11:1259, 383
\g__template_type_prop . . . 11:75,
    11:120, 11:143, 11:277, 11:288, 381
\__template_use_instance:nn . . .
    . . 11:1159, 11:1212, 11:1212, 11:1568
\__template_use_instance_-
    aux:nNnnn 11:1212, 11:1215, 11:1218
\__template_use_instance_auxi:nn
    . . . . . 11:1212, 11:1216, 11:1224
\__template_use_instance_-
    auxii:nn . 11:1212, 11:1227, 11:1230
\__template_use_template:nnn . . .
    . . 11:1066, 11:1066, 11:1221, 11:1560
\l__template_value_exp_str 11:85,
    11:795, 11:805, 11:815, 11:828, 382
\l__template_value_order_seq . . .
    . . . . . 11:89, 11:767, 11:854,
    11:868, 11:886, 11:894, 11:1260, 382
\l__template_value_tl . . . . .
    . 11:83, 11:1028, 11:1089, 11:1096,
    11:1102, 11:1110, 11:1120, 11:1144,
    11:1161, 11:1177, 11:1183, 11:1190,
    11:1195, 11:1200, 11:1202, 11:1210, 382
\l__template_values_prop . . 11:88,
    11:184, 11:208, 11:227, 11:230,
    11:253, 11:256, 11:302, 11:364,
    11:686, 11:708, 11:887, 11:895,
    11:921, 11:928, 11:933, 11:969,
    11:1002, 11:1028, 11:1271, 11:1310, 382
\c__template_values_root_tl . . .
    . . . . . 11:54, 11:206,

```

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 11:207, 11:251, 11:254, 11:1064, 381
- `\l__template_var_tl`
 11:84, 11:1030, 11:1051,
 11:1053, 11:1058, 11:1095, 11:1103,
 11:1141, 11:1157, 11:1182, 11:1189, 382
- `\l__template_vars_prop`
 11:90, 11:219, 11:270,
 11:273, 11:304, 11:516, 11:644,
 11:656, 11:661, 11:681, 11:699,
 11:703, 11:744, 11:855, 11:869,
 11:1030, 11:1126, 11:1199, 11:1289, 382
- `\c__template_vars_root_tl` . 11:55,
 11:217, 11:218, 11:268, 11:271, 381
- `\tencirc` 31:10, 42:136, 42:688
- `\tencircw` 31:10, 42:139
- `\tenln` . 31:9, 42:135, 42:137, 42:687, 42:689
- `\tenlnw` 31:9, 42:138, 42:140
- `\test (hook)` 237, 237
- `\test` 1368
- `\testallgroups`
 33:1214, 33:1387, 33:1388, 33:1389,
 33:1391, 33:1392, 33:1393, 33:1394,
 33:1395, 33:1396, 33:1397, 33:1398,
 33:1399, 33:1400, 33:1401, 33:1402,
 33:1403, 33:1404, 33:1405, 33:1406,
 33:1407, 33:1408, 33:1409, 33:1413,
 33:1414, 33:1415, 33:1416, 33:1417,
 33:1418, 33:1422, 33:1423, 33:1424,
 33:1428, 33:1429, 33:1430, 33:1431,
 33:1432, 33:1433, 33:1434, 33:1435,
 33:1436, 33:1437, 33:1438, 33:1439,
 33:1440, 33:1441, 33:1442, 33:1443,
 33:1444, 33:1445, 33:1446, 33:1447,
 33:1448, 33:1450, 33:1454, 33:1455,
 33:1456, 33:1457, 33:1458, 33:1459,
 33:1460, 33:1461, 33:1462, 33:1464,
 33:1465, 33:1466, 33:1467, 33:1468,
 33:1469, 33:1470, 33:1471, 33:1472,
 33:1473, 33:1474, 33:1475, 33:1476,
 33:1477, 33:1478, 33:1479, 33:1480,
 33:1481, 33:1482, 33:1483, 33:1484,
 33:1485, 33:1486, 33:1487, 33:1488,
 33:1489, 33:1490, 33:1491, 33:1492,
 33:1493, 33:1494, 33:1495, 33:1496,
 33:1497, 33:1498, 33:1499, 33:1500,
 33:1501, 33:1502, 33:1503, 33:1504,
 33:1505, 33:1506, 33:1507, 33:1508,
 33:1509, 33:1510, 33:1511, 33:1512,
 33:1513, 33:1514, 33:1515, 33:1516,
 33:1517, 33:1518, 33:1519, 33:1520,
 33:1521, 33:1522, 33:1523, 33:1524,
 33:1525, 33:1526, 33:1527, 33:1528,
 33:1529, 33:1530, 33:1531, 33:1532,
 33:1533, 33:1534, 33:1535, 33:1536,
 33:1537, 33:1538, 33:1539, 33:1540,
 33:1541, 33:1542, 33:1543, 33:1544,
 33:1545, 33:1546, 33:1547, 33:1548,
 33:1549, 33:1550, 33:1551, 33:1552,
 33:1553, 33:1554, 33:1555, 33:1556,
 33:1557, 33:1558, 33:1559, 33:1560,
 33:1561, 33:1562, 33:1563, 33:1564,
 33:1565, 33:1566, 33:1567, 33:1568,
 33:1569, 33:1570, 33:1571, 33:1572,
 33:1573, 33:1574, 33:1575, 33:1576,
 33:1577, 33:1578, 33:1579, 33:1598
- `\testallkerneldefinedfamilies`
 33:1386, 33:1595
- `\testFont` 33:1182, 33:1183, 33:1215, 33:1216
- `\testgroup` . 33:1187, 33:1225, 33:1234,
 33:1240, 33:1246, 33:1250, 33:1254,
 33:1264, 33:1268, 33:1336, 33:1340
- `\TeX` 19:1, 19:12, 1413
- `\TeX` and `\LaTeX` 2_ε commands:
- `\...-h@@k` 1125
- `\..@without@substitution` 676
- `\@` 09:408, 04:20, 14:19,
 18:522, 19:2, 04:1041, 50:43, 50:59,
 50:75, 50:88, 50:97, 50:139, 50:150,
 06:925, 06:926, 57:561, 01:49, 1363
- `\@...hook` 255
- `\@?@?` 1363
- `\@@` . 13:15, 13:19, 13:20, 13:21, 13:22,
 13:24, 13:27, 13:28, 13:30, 13:31,
 20:772, 20:792, 26:527, 26:529,
 26:530, 41:315, 41:316, 41:317,
 41:327, 01:315, 01:316, 54:6, 54:7
- `\@@DeclareMathDelimiter` 1374
- `\@@data@glyphtounicode` 57:345, 57:351
- `\@@data@glyphtounicode@cmex`
 57:347, 57:352
- `\@@defaultsubs` 24:682
- `\@@enc@update`
 21:201, 24:322, 24:326, 1386
- `\@@end` 20:741, 20:742, 01:206,
 06:23, 37:35, 37:100, 37:176, 37:242,
 56:18, 57:749, 57:770, 01:53, 1368
- `\@@endpbox` 41:245, 41:268, 41:313, 41:463
- `\@@eqnocr` 38:443,
 38:472, 38:496, 38:506, 38:511, 38:634
- `\@@fileswith@pti@ns`
 50:640, 50:659, 50:1147
- `\@@hyph` 06:24, 06:942
- `\@@hyphenation` 21:225
- `\@@if@newlist` 54:877,
 54:941, 54:954, 54:999, 54:1012, 54:1058
- `\@@if@definable` 21:35, 06:173
- `\@@input` 20:639,
 20:652, 20:692, 06:22, 37:29,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- 37:94, 37:156, 50:1772, 52:172,
52:188, 52:196, 57:383, 01:52, 1405
 $\backslash$ @italiccorr 06:25, 32:114, 32:118, 1370
 $\backslash$ @line 40:725
 $\backslash$ @math@bgroup 32:132, 32:139
 $\backslash$ @math@egroup 32:129
 $\backslash$ @par . 15:4, 16:142, 16:179, 06:21,
37:242, 37:533, 37:538, 37:541,
37:555, 37:559, 37:562, 39:82, 39:85,
40:373, 40:398, 40:424, 40:448,
40:469, 41:247, 41:251, 41:270,
41:274, 44:67, 44:118, 54:262, 1396
 $\backslash$ @patterns 21:225
 $\backslash$ @protect 06:358, 06:364, 06:373
 $\backslash$ @sqrt 1392
 $\backslash$ @startpbox
..... 41:245, 41:268, 41:313, 41:463
 $\backslash$ @sverb 37:648, 885
 $\backslash$ @text@case@aux
..... 57:629, 57:663, 57:668, 57:673
 $\backslash$ @underline
..... 40:614, 40:622, 40:659, 40:660
 $\backslash$ @unprocessedoptions
..... 50:1059, 50:1124, 50:1222
 $\backslash$ @warning 14:215, 1366
 $\backslash$ @write 54:112, 54:116
 $\backslash$ @Alph 22:307, 22:323, 111
 $\backslash$ @DeclareEncodingSubset .. 24:191,
24:193, 24:194, 24:195, 24:196, 24:199
 $\backslash$ @DeclareMathDelimiter
..... 28:1039, 28:1058, 1374
 $\backslash$ @DeclareMathSizes
..... 24:269, 24:270, 24:272
 $\backslash$ @Esphack
. 18:213, 45:201, 45:223, 45:241, 1360
 $\backslash$ @Ialph 1376
 $\backslash$ @IncludeInRele@se 03:71
 $\backslash$ @IncludeInRelease 03:71
 $\backslash$ @LRmoderr 14:271, 18:261, 18:312, 1433
 $\backslash$ @M 18:11,
18:12, 18:13, 18:14, 18:15, 18:16,
18:17, 18:18, 18:19, 18:120, 24:752,
24:760, 26:456, 26:469, 06:37,
06:39, 38:484, 39:225, 41:56, 44:67,
44:100, 44:118, 44:130, 44:234,
44:257, 02:21, 54:155, 54:182,
54:200, 54:203, 54:263, 02:383, 02:384
 $\backslash$ @MM 45:531,
45:550, 45:568, 02:21, 54:304, 421
 $\backslash$ @Mi 12:3, 54:114
 $\backslash$ @Mii 12:3, 45:53, 45:122,
45:194, 45:216, 45:241, 45:311,
54:300, 54:1511, 54:1660, 54:1822
 $\backslash$ @Miii 12:3, 45:55, 45:124, 45:313, 54:303
 $\backslash$ @Miv 12:3,
45:195, 45:201, 45:217, 45:223, 54:277
 $\backslash$ @Roman 22:305, 22:311, 1399
 $\backslash$ @TeXversion 14:28, 01:310, 2
 $\backslash$ @abspage@last ... 53:166, 53:172,
53:398, 53:400, 53:402, 53:415,
53:419, 53:422, 53:462, 53:463, 1206
 $\backslash$ @acci ... 29:932, 40:450, 40:471, 1376
 $\backslash$ @accii .. 29:932, 40:450, 40:471, 1376
 $\backslash$ @acciii . 29:932, 40:450, 40:471, 1376
 $\backslash$ @acol 41:172, 41:179, 41:206, 41:213,
41:337, 41:338, 41:350, 41:351,
41:354, 41:371, 41:386, 41:394, 41:404
 $\backslash$ @acolampacol 41:335, 41:352,
41:354, 41:361, 41:369, 41:403, 41:406
 $\backslash$ @activechar@info 54:826, 1373
 $\backslash$ @activechar@warning 1373
 $\backslash$ @addamp 41:328, 41:337,
41:338, 41:353, 41:367, 41:404, 41:405
 $\backslash$ @addfield 41:43, 41:53,
41:86, 41:93, 41:125, 41:140, 41:142
 $\backslash$ @addmarginpar 54:336, 54:2388
 $\backslash$ @addtobot
54:1326, 54:1413, 54:1481, 54:1533,
54:1630, 54:1679, 54:1786, 54:1845
 $\backslash$ @addtocurcol 54:333, 54:1417, 54:2576
 $\backslash$ @addtodblcol 54:1205, 54:2139
 $\backslash$ @addtofilelist
... 20:64, 20:133, 20:188, 20:639,
20:761, 29:906, 29:909, 29:916,
29:919, 29:926, 29:929, 50:1771,
52:169, 52:188, 52:196, 57:281,
57:284, 57:792, 01:85, 01:87, 1370
 $\backslash$ @addtofilelist@aux 57:792
 $\backslash$ @addtonextcol 54:1204, 54:1882, 54:2577
 $\backslash$ @addtopreamble .. 41:388, 41:401,
41:407, 41:408, 41:409, 41:411, 41:423
 $\backslash$ @addtoreset 22:33,
22:62, 22:82, 22:101, 22:117, 22:148,
22:209, 22:224, 22:238, 22:241, 561
 $\backslash$ @addtotoporbot
..... 54:1363, 54:1527, 54:1675,
54:1839, 54:1932, 54:2013, 54:2101
 $\backslash$ @afterheading ... 44:92, 44:125, 1397
 $\backslash$ @afterindentfalse 44:45
 $\backslash$ @afterindenttrue
..... 44:43, 44:124, 44:233, 44:256
 $\backslash$ @alph 22:306, 22:319, 45:399, 111
 $\backslash$ @ampacol 41:335, 41:352, 41:363, 41:406
 $\backslash$ @arabic 22:70, 22:90, 22:107,
22:246, 22:293, 22:303, 22:309, 45:397
 $\backslash$ @argarraycr 41:280, 41:281
 $\backslash$ @argdef 06:98
 $\backslash$ @argrsbox 40:686

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- `\@argtabularcr` 41:287, [41:288](#)
- `\@array` 41:222, 41:227, [41:230](#)
- `\@arrayacol` 41:172, 41:179, [41:335](#)
- `\@arrayclassiv` 41:173, 41:180, [41:408](#)
- `\@arrayclassz` 41:172, 41:179, [41:352](#)
- `\@arraycr` 41:174, 41:181, [41:278](#), [41:280](#)
- `\@arrayparboxrestore`
. [40:440](#), 40:482, 41:461
- `\@arrayrule` 41:386,
41:388, 41:392, 41:394, 41:396, [41:423](#)
- `\@arstrut` 41:244, 41:267, 41:314, [41:420](#)
- `\@arstrutbox` 41:237,
41:260, 41:295, [41:420](#), 41:462, [1374](#)
- `\@author` 44:8, [44:32](#)
- `\@auxout` 20:236,
20:242, 20:286, 20:304, 20:328,
20:361, 20:390, 20:413, 20:437,
20:452, 35:101, 35:119, 35:128,
36:55, 44:189, 44:199, 47:7, 47:8,
47:39, 47:49, 47:57, 47:67, 47:84, 53:414
- `\@backslashchar` 14:234, 14:236,
30:277, [06:272](#), 06:573, 06:722,
06:734, 06:738, 06:739, 06:744,
06:788, 50:1351, 50:1484, 50:1573, [1376](#)
- `\@backup@outputbox@depth`
. 54:579, [54:589](#), 54:725, [1245](#)
- `\@badcrrr` [14:279](#), [1374](#)
- `\@badend` [14:247](#), 37:392, [1423](#)
- `\@badlinearg` [14:267](#), 42:176,
42:188, 42:199, 42:200, 42:204,
42:253, 42:258, 42:268, 42:273, 42:286
- `\@badmath` [14:251](#),
38:275, 38:277, 38:282, 38:285,
38:311, 38:325, 38:330, 38:339,
38:351, 38:356, 38:364, 38:377,
38:382, 38:557, 38:569, 38:585, 38:594
- `\@badpoptabs` [14:255](#), 41:85, 41:151
- `\@badrequireerror` 50:532, [50:1230](#)
- `\@badtab` [14:258](#), 41:22,
41:87, 41:108, 41:114, 41:121, 41:148
- `\@begin@tempboxa` 40:27,
40:46, 40:61, 40:251, 40:270, 40:373,
40:398, 40:424, 40:687, 40:695, [1365](#)
- `\@begin@documenthook`
. 20:61, 20:127, 20:130, 20:182,
20:185, 47:53, [50:1158](#), 50:1179, [252](#)
- `\@begin@dv`
54:910, 54:978, 54:1037, [54:1065](#), [1191](#)
- `\@begin@dvbox`
. 53:502, 53:503, [54:62](#), 54:1066, [1186](#)
- `\@begin@parpenalty` 18:15, 38:560,
38:572, 38:598, [39:23](#), 39:201, [909](#)
- `\@begin@theorem` 43:57, [43:62](#)
- `\@bezier` [42:694](#), 42:693
- `\@bibitem` 47:3, [47:8](#)
- `\@biblabel` 47:4, [47:97](#)
- `\@bitor` 54:11,
54:1232, 54:1252, 54:1288, 54:1311,
54:1378, 54:1463, 54:1473, 54:1612,
54:1622, 54:1765, 54:1776, 54:1919,
54:2000, 54:2086, 54:2204, 54:2329
- `\@botlist` 54:41, 54:389, 54:391,
54:436, 54:438, 54:637, 54:657,
54:810, 54:1089, 54:1098, 54:1099,
54:1340, 54:1343, 54:1378, 54:1473,
54:1622, 54:1776, 54:2532, 54:2560
- `\@botnum` 45:274, 54:87, 54:1337,
54:1338, 54:1343, 54:1347, 54:1955,
54:1960, 54:2036, 54:2041, 54:2128,
54:2135, 54:2524, 54:2552, 54:2594
- `\@botroom` 45:275, 54:88,
54:1340, 54:1343, 54:2525, 54:2553
- `\@boxfpsbit` 54:2652, [54:2654](#), 54:2659
- `\@break@loop` [1374](#)
- `\@break@tfor`
. [13:31](#), 20:612, 20:629, 32:99, [1374](#)
- `\@bsphack`
. 18:37, [18:126](#), 18:409, 18:425,
18:443, 18:459, 35:98, 35:116,
35:127, 36:82, 44:186, 44:187, 45:52,
45:121, 45:310, 46:6, 46:18, 46:23,
46:35, 47:63, 47:80, 54:2459, [849](#)
- `\@caption` 45:12, [45:14](#)
- `\@capttype` 45:5, 45:9, 45:12,
45:40, 45:88, 45:109, 45:157, 54:2616
- `\@car` 19:14,
21:107, 21:128, [06:53](#), 06:166, 06:222
- `\@carcube` [06:55](#), 06:176, 06:725
- `\@cclv` [02:16](#),
54:305, 54:309, 54:387, 54:388,
54:417, 54:434, 54:435, 54:464,
54:492, 54:739, 54:743, 57:67, [1191](#)
- `\@cclvi` 04:30, 04:58, [02:21](#), 50:1349,
50:1482, 50:1571, 02:53, 02:78, 02:103
- `\@cdr` 20:270, [06:53](#), 06:868, 06:869
- `\@centercr` 37:415, 37:453, 37:458,
37:463, 37:473, 37:477, 37:481, [1413](#)
- `\@centering` 38:418,
38:419, 38:431, 38:434, 38:437,
38:460, 38:463, 38:466, 38:627, 38:631
- `\@cflb` 54:639, 54:810, [54:1070](#)
- `\@cflt` 54:634, 54:809, [54:1070](#)
- `\@changed@cmd` 21:3, 21:81,
21:243, 24:132, 24:330, 57:525, [1387](#)
- `\@changed@x` 21:3, 21:231, 21:239, [1386](#)
- `\@changed@x@err` [1394](#)
- `\@changed@x@mouth` 21:231, 21:239, [1386](#)
- `\@charlb` 20:458, [20:466](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@charrb</code>	20:460, 20:466	50:1076, 50:1137, 50:1150, 50:1190,
<code>\@chclass</code>		50:1205, 50:1657, 51:100, 51:167, 1170
41:348, 41:349, 41:412 , 41:425, 41:430		
<code>\@check@IncludeInRelease</code>	03:71	<code>\@clubpenalty</code> ..
<code>\@check@c</code>	06:230, 06:232	20:7, 20:25, 20:95,
<code>\@check@eq</code>	06:236, 06:237, 06:241	20:152, 39:128, 39:227, 44:106, 44:135
<code>\@checkcommand</code>	1378	<code>\@colht</code> 20:22, 20:92, 20:149, 45:273,
<code>\@checkend</code>		45:275, 45:278, 45:284, 45:285,
.... 37:16, 37:84, 37:148, 37:339,		45:298, 45:299, 54:92, 54:236,
37:354, 37:371, 37:381, 37:391 , 1388		54:247, 54:256, 54:257, 54:392,
<code>\@chnum</code>	41:356,	54:404, 54:439, 54:452, 54:481,
41:375, 41:412 , 41:427, 41:428, 41:429		54:509, 54:526, 54:532, 54:536,
<code>\@circ</code>		54:545, 54:550, 54:759, 54:777,
42:632, 42:650, 42:659, 42:664, 42:667		54:781, 54:788, 54:942, 54:1000,
<code>\@circle</code>	42:615, 42:616	54:1059, 54:1128, 54:1166, 54:1210,
<code>\@circlefnt</code>	42:136,	54:1235, 54:1254, 54:1294, 54:1316,
42:139, 42:458, 42:501, 42:530,		54:2219, 54:2345, 54:2717, 57:155
42:555, 42:625, 42:641, 42:673, 42:688		<code>\@colnum</code>
<code>\@cite</code>	47:36, 47:95	.. 45:276, 54:89, 54:1346, 54:1391,
<code>\@cite@ofmt</code>	47:44, 47:96	54:1461, 54:1462, 54:1490, 54:1498,
<code>\@citea</code>	47:35, 47:37	54:1610, 54:1611, 54:1639, 54:1647,
<code>\@citeb</code>	47:38, 47:39,	54:1763, 54:1764, 54:1796, 54:1808,
47:40, 47:43, 47:44, 47:66, 47:67,		54:1917, 54:1918, 54:1955, 54:1960,
47:68, 47:69, 47:83, 47:84, 47:85, 47:86		54:1998, 54:1999, 54:2036, 54:2041,
<code>\@citex</code> 47:20, 47:21, 47:29, 47:34 , 1054		54:2084, 54:2085, 54:2127, 54:2134,
<code>\@citex@checkblank</code> 47:17, 47:18, 47:30		54:2520, 54:2548, 54:2587, 54:2827
<code>\@classi</code>	41:348, 41:384	<code>\@colroom</code>
<code>\@classii</code>	41:348, 41:398	20:23, 20:93, 20:150, 54:93, 54:257,
<code>\@classiii</code>	41:348, 41:403	54:278, 54:279, 54:290, 54:293,
<code>\@classiv</code>		54:392, 54:439, 54:1128, 54:1345,
41:173, 41:180, 41:208, 41:215, 41:349		54:1390, 54:1457, 54:1460, 54:1489,
<code>\@classoptionslist</code>		54:1606, 54:1609, 54:1638, 54:1758,
.... 50:9, 50:581, 50:596, 50:597,		54:1762, 54:1795, 54:1913, 54:1916,
50:614, 50:853, 50:854, 50:882,		54:1994, 54:1997, 54:2079, 54:2083,
50:883, 50:909, 50:910, 50:1853, 1421		54:2521, 54:2549, 54:2732, 54:2737,
<code>\@classv</code>	41:349, 41:409	54:2782, 54:2787, 54:2832, 57:154, 1395
<code>\@classz</code>	41:172,	<code>\@combinedblfloats</code>
41:179, 41:207, 41:214, 41:348, 1321		.. 54:1101, 54:2891, 54:2933, 54:2970
<code>\@cline</code>	41:444	<code>\@combinefloats</code>
<code>\@clnht</code> 42:206, 42:207, 42:215, 42:217,		 54:755, 54:808, 54:1069 , 1247
42:219, 42:229, 42:236, 42:284, 42:682		<code>\@comdblfllelt</code>
<code>\@clnwd</code>	42:208,	54:1101
42:214, 42:218, 42:220, 42:221, 42:682		<code>\@comflelt</code> ..
<code>\@cls@pkg</code>	08:2721,	54:1071, 54:1087, 54:1101
50:388, 50:389, 50:402, 50:403,		<code>\@compatibility</code>
50:998, 50:1008, 50:1056, 50:1103,		1369
50:1133, 50:1185 , 50:1215, 50:1217,		<code>\@cons</code>
50:1234, 50:1781, 50:1803, 50:1833		22:124, 22:136,
<code>\@clsextension</code>	50:31,	22:144, 06:52 , 45:193, 45:215,
50:191, 50:199, 50:257, 50:418,		45:239, 45:379, 54:242, 54:1239,
50:434, 50:446, 50:528, 50:552,		54:1258, 54:1274, 54:1298, 54:1300,
50:563, 50:581, 50:595, 50:613,		54:1320, 54:1322, 54:1493, 54:1561,
50:712, 50:727, 50:751, 50:852,		54:1642, 54:1706, 54:1801, 54:1874,
50:881, 50:908, 50:1002, 50:1049,		54:1948, 54:2029, 54:2118, 54:2221,
		54:2244, 54:2347, 54:2372, 54:2389,
		54:2390, 54:2833, 02:119, 02:137
		<code>\@contentsline@destination</code>
		 44:213, 44:215, 44:222, 1023
		<code>\@contfield</code>
		41:50 , 41:141, 41:153
		<code>\@copy@...</code>
		103

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltaloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@copy@DeclareRobustCommand</code>	01:139, 01:149, <u>01:162</u> , 01:227, 06:589, 06:659, <u>06:680</u> , 06:758, 06:761
<code>\@copy@newcommand</code>	50:1410, 50:1432, 50:1461, 50:1544, <i>9</i>
. 06:410, 06:590, 06:659,	<code>\@current@cmd</code> 21:43, 24:334, <i>1386</i>
06:701, <u>06:731</u> , 06:758, 06:764, <i>96</i>	<code>\@currentHpage</code> . . . 36:263, 36:264, <i>854</i>
<code>\@copytexsys</code> <i>1368</i>	<code>\@currentHref</code> 17:128, 17:132, 17:152, 17:156, 17:177, 35:106, 35:113, 35:121, 35:137, 36:262, <i>854</i>
<code>\@ctrerr</code>	<code>\@currentcounter</code> 22:31, <u>35:139</u> , 35:146, 35:159, 35:167, 35:176, 35:178, 35:185, 36:265, 38:427, 38:456, 38:612, 40:548, 43:67, 43:69, 45:533, <i>845</i>
<code>\@curfield</code> <u>41:16</u> , 41:41, 41:47, 41:51, 41:52, 41:54, 41:130, 41:131	<code>\@currentlabel</code> 35:105, 35:120, 35:129, 35:148, 35:161, 35:168, 35:179, 35:187, 35:194, <u>35:230</u> , 36:259, 38:426, 38:455, 38:611, 40:549, 40:566, 40:583, 45:534, 45:552, 45:570, <i>854</i>
<code>\@curline</code> <u>41:16</u> , 41:27, 41:39, 41:44, 41:53, 41:54, 41:55, 41:90, 41:91, 41:103, 41:128, 41:129	<code>\@currentlabelname</code> 35:106, 35:112, 35:121, 36:261, <i>854</i>
<code>\@curr@enc</code> 21:172, 21:174	<code>\@currentmetafamily</code> <u>29:163</u> , 29:165, 29:197, 29:204, 29:206, 29:229, 29:781, 29:784, 29:787, 29:789, <i>1439</i>
<code>\@curr@file</code>	<code>\@curr@envir</code>
. . . 20:246, 20:247, 20:256, 20:258,	 14:249, <u>37:3</u> , 37:253, 37:274, 37:292, 37:302, 37:392, 39:112, 40:193, 50:1339, 50:1351, 50:1359, 50:1363, 50:1370, 50:1472, 50:1484, 50:1492, 50:1496, 50:1502, 50:1561, 50:1573, 50:1581, 50:1585, 50:1591, 07:1632, 07:1738, 07:1753, 07:1764, <i>435</i>
20:282, 20:290, <u>20:471</u> , 20:490,	<code>\@curr@envline</code> 14:249, 37:254, 37:275, 37:293, 37:303, <u>37:393</u> , 40:194
20:659, 20:674, 50:983, 50:1296,	<code>\@curr@ext</code> <u>50:30</u> , 50:42, 50:58, 50:74, 50:87, 50:96, 50:138, 50:149, 50:414, 50:417, 50:418, 50:433, 50:434, 50:445, 50:446, 50:552, 50:563, 50:574, 50:581, 50:595, 50:613, 50:642, 50:722, 50:735, 50:737, 50:745, 50:759, 50:951, 50:952, 50:954, 50:958, 50:961, 50:963, 50:968, 50:973, 50:975, 50:980, 50:986, 50:994, 50:1000, 50:1002, 50:1006, 50:1011, 50:1017, 50:1026, 50:1029, 50:1034, 50:1037, 50:1040, 50:1042, 50:1043, 50:1045, 50:1049, 50:1058, 50:1060, 50:1061, 50:1066, 50:1069, 50:1072, 50:1076, 50:1082, 50:1095, 50:1100, 50:1101, 50:1106, 50:1112, 50:1116, 50:1118, 50:1119, 50:1121, 50:1123, 50:1125, 50:1126, 50:1129, 50:1135, 50:1137, 50:1150, 50:1163, 50:1190,
50:1301, 50:1307, 50:1313, 50:1317,	
50:1328, 50:1337, 50:1364, 50:1427,	
50:1432, 50:1438, 50:1461, 50:1470,	
50:1496, <u>52:266</u> , 52:374, 52:376, <i>1174</i>	
<code>\@curr@file@reqd</code>	
. <u>52:266</u> , 52:376, 52:380, <i>1175</i>	
<code>\@curr@box</code> 45:60, 45:91, 45:95, 45:129, 45:160, 45:164, 45:193, 45:214, 45:215, 45:239, 45:257, 45:259, 45:261, 45:319, 45:322, 45:327, 45:331, 54:218, 54:219, 54:230, 54:231, 54:233, 54:234, 54:242, 54:316, 54:317, 54:1204, 54:1205, 54:1453, 54:1456, 54:1464, 54:1487, 54:1491, 54:1493, 54:1508, 54:1549, 54:1561, 54:1603, 54:1605, 54:1613, 54:1636, 54:1640, 54:1642, 54:1657, 54:1694, 54:1706, 54:1753, 54:1756, 54:1793, 54:1798, 54:1801, 54:1818, 54:1863, 54:1874, 54:1906, 54:1923, 54:1937, 54:1948, 54:1988, 54:2004, 54:2018, 54:2029, 54:2070, 54:2107, 54:2118, 54:2158, 54:2162, 54:2173, 54:2179, 54:2181, 54:2185, 54:2190, 54:2199, 54:2208, 54:2214, 54:2221, 54:2244, 54:2279, 54:2283, 54:2295, 54:2302, 54:2304, 54:2308, 54:2314, 54:2324, 54:2339, 54:2347, 54:2372, 54:2390, 54:2399, 54:2622, 54:2623, 54:2652, 54:2682, 54:2687, 54:2738, 54:2741, 54:2753, 54:2761, 54:2788, 54:2791, 54:2803, 54:2811, 54:2828, 54:2833, 02:161, 02:162, 02:163	
<code>\@curr@dir</code>	
. . . 01:92, 01:114, 01:116, 01:122, 01:124, 01:130, 01:132, 01:137,	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

50:1193, 50:1205, 50:1223, 50:1224, 51:66, 51:83, 51:85, 51:100, 51:107, 51:109, 51:167, 51:169, 51:172, <i>1124</i>	<code>\@curtabmar</code> 41:11 , 41:25, 41:26, 41:38, 41:44, 41:89, 41:102, 41:106, 41:107
<code>\@currextension</code> <i>1362</i>	<code>\@d@r</code> 01:145, 01:146
<code>\@currlist</code> 45:193, 45:215, 45:379, 54:43, 54:316, 54:393, 54:396, 54:440, 54:443, 54:2389	<code>\@dashbox</code> 42:335, 42:336, 42:337, 42:338, 42:339, 42:342, 42:348, 42:350, 42:360, 42:362, 42:363, 42:364, 42:365, 42:369, 42:373, 42:376, 42:393, 42:395, 42:396, 42:397, 42:398, 42:401, 42:405, 42:407, 42:416, 42:418, 42:419, 42:420, 42:421, 42:424, 42:428, 42:433, 42:684
<code>\@currname</code> 03:71, 03:116, 03:124, 17:27, 20:781, 20:803, 20:809, 50:29 , 50:41, 50:57, 50:73, 50:86, 50:95, 50:137, 50:148, 50:386, 50:388, 50:400, 50:402, 50:414, 50:417, 50:433, 50:445, 50:574, 50:642, 50:735, 50:737, 50:745, 50:759, 50:949, 50:952, 50:954, 50:958, 50:961, 50:963, 50:968, 50:970, 50:973, 50:975, 50:980, 50:985, 50:994, 50:998, 50:1000, 50:1006, 50:1008, 50:1009, 50:1011, 50:1017, 50:1026, 50:1028, 50:1034, 50:1037, 50:1040, 50:1042, 50:1043, 50:1047, 50:1051, 50:1058, 50:1060, 50:1061, 50:1066, 50:1069, 50:1073, 50:1077, 50:1082, 50:1094, 50:1118, 50:1119, 50:1121, 50:1123, 50:1125, 50:1126, 50:1163, 50:1215, 50:1217, 50:1224, 50:1234, 50:1781, 50:1803, 50:1833, 51:66, 51:80, 51:83, 51:85, 51:107, 51:109, 51:169, 51:172, 51:214, 51:216, 51:227, 51:285, 08:362, 08:368, 08:423, <i>1124</i>	<code>\@dashcnt</code> 42:328, 42:330, 42:331, 42:332, 42:333, 42:334, 42:347, 42:349, 42:352, 42:354, 42:355, 42:356, 42:358, 42:359, 42:372, 42:375, 42:387, 42:388, 42:389, 42:390, 42:391, 42:392, 42:404, 42:406, 42:409, 42:410, 42:411, 42:412, 42:414, 42:415, 42:427, 42:432, 42:684
<code>\@currnamestack</code> 50:33 , 50:171, 52:546, 08:420, <i>260</i>	<code>\@dashdim</code> 42:327, 42:328, 42:329, 42:330, 42:331, 42:333, 42:336, 42:338, 42:339, 42:340, 42:347, 42:349, 42:351, 42:352, 42:353, 42:354, 42:355, 42:358, 42:362, 42:364, 42:365, 42:366, 42:374, 42:377, 42:386, 42:387, 42:388, 42:389, 42:391, 42:395, 42:397, 42:398, 42:399, 42:404, 42:406, 42:408, 42:409, 42:410, 42:411, 42:414, 42:418, 42:420, 42:421, 42:422, 42:430, 42:435, 42:684
<code>\@curroption</code> <i>1362</i>	<code>\@date</code> 44:9, 44:33
<code>\@curroptions</code> 50:574, 50:582, 50:632, 50:651, 50:1224, 50:1225	<code>\@dbflt</code> 45:32, 45:264 , <i>1382</i>
<code>\@currpath</code> 50:15 , 50:47, 50:62, 50:174, 50:386, 50:388, 50:950, 50:968, 50:975, 50:984, 50:1026, 50:1027, 50:1054, <i>1123</i>	<code>\@dblarg</code> .. 44:54, 44:142, 45:12, 06:890
<code>\@currpkg@reqd</code> ... 50:416, 50:978, 50:980, 50:989, 50:1022, 50:1036, 50:1039, 50:1054, 50:1056, <i>1418</i>	<code>\@dbldeferlist</code> 45:239, 54:46, 54:450, 54:455, 54:457, 54:1167, 54:1174, 54:1175, 54:2329, 54:2372, 54:2536, 54:2565, <i>1407</i>
<code>\@currsiz</code> 29:750, <i>1361</i>	<code>\@dblfloat</code> 45:31
<code>\@currtype</code> 54:97, 54:1229, 54:1230, 54:1231, 54:1232, 54:1249, 54:1250, 54:1251, 54:1252, 54:1378, 54:1463, 54:1473, 54:1612, 54:1622, 54:1765, 54:1776, 54:1919, 54:2000, 54:2086, 54:2204, 54:2329, 54:2622, 54:2624, 54:2625, 54:2628	<code>\@dblfloatplacement</code> 20:31, 20:101, 20:159, 45:280 , 54:406, 54:454, 54:2517, 54:2545, 54:2895, 54:2937, 54:2976
<code>\@curtab</code> 41:11 , 41:26, 41:86, 41:87, 41:88, 41:94, 41:95, 41:98, 41:102, 41:103, 41:107, 41:146, 41:147, <i>1393</i>	<code>\@dblflset</code> 45:26
	<code>\@dblfpbot</code> 45:290, 45:304, 54:3020
	<code>\@dblpsep</code> 45:289, 45:303, 54:3020
	<code>\@dblpftop</code> 45:288, 45:302, 54:3020
	<code>\@dbltoplist</code> 54:45, 54:237, 54:240, 54:242, 54:402, 54:403, 54:450, 54:451, 54:1106, 54:1110, 54:1112, 54:1113, 54:2216, 54:2221, 54:2341, 54:2347, 54:2535, 54:2563

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@dbltopnum 45:283,
45:297, 54:85, 54:105, 54:243,
54:245, 54:1117, 54:2155, 54:2156,
54:2220, 54:2223, 54:2251, 54:2256,
54:2276, 54:2277, 54:2346, 54:2350,
54:2379, 54:2384, 54:2528, 54:2556
- \@dbltoproom 45:284, 45:286, 45:298,
45:300, 54:86, 54:2158, 54:2161,
54:2162, 54:2171, 54:2172, 54:2175,
54:2178, 54:2181, 54:2185, 54:2189,
54:2193, 54:2198, 54:2218, 54:2279,
54:2282, 54:2283, 54:2292, 54:2293,
54:2294, 54:2297, 54:2301, 54:2304,
54:2308, 54:2313, 54:2317, 54:2322,
54:2323, 54:2344, 54:2529, 54:2557
- \@dec@text@cmd 21:3
- \@declarecommandcopylisthook ...
. 06:586, 06:588, 06:600, 07:1167, 163
- \@declaredoptions
..... 50:8, 50:535, 50:578,
50:616, 50:637, 50:656, 50:1156, 1362
- \@declareoption 50:533, 50:534, 50:542
- \@defaultfamilyhook 29:508, 29:808,
29:823, 29:838, 29:843, 29:858, 1415
- \@defaultsubs 24:636, 24:670,
24:682, 37:59, 37:124, 37:163, 1391
- \@defaultunits
... 24:277, 24:281, 24:282, 24:283,
24:298, 24:391, 26:180, 26:182, 42:13
- \@defaultunitsset ... 40:74, 40:85,
42:8, 42:40, 42:41, 42:43, 42:45,
42:71, 42:74, 42:95, 42:96, 42:118,
42:119, 42:175, 42:252, 42:327,
42:329, 42:343, 42:351, 42:353,
42:368, 42:490, 42:491, 42:622,
42:658, 42:700, 42:701, 42:703,
42:704, 42:707, 42:708, 42:710,
42:711, 42:722, 42:723, 42:725,
42:726, 42:728, 42:729, 42:731, 42:732
- \@defdefault@ds 50:533, 50:538, 50:543
- \@deferlist 54:44,
54:389, 54:398, 54:399, 54:402,
54:407, 54:409, 54:415, 54:436,
54:445, 54:447, 54:1129, 54:1137,
54:1138, 54:1149, 54:1154, 54:1155,
54:1463, 54:1561, 54:1612, 54:1706,
54:1765, 54:1874, 54:1919, 54:1948,
54:2000, 54:2029, 54:2086, 54:2118,
54:2204, 54:2244, 54:2534, 54:2562
- \@definecounter 22:12, 22:53,
38:393, 39:258, 39:259, 39:260,
39:261, 43:8, 43:16, 45:396, 45:398, 559
- \@depth 26:192,
06:26, 30:569, 30:570, 30:572,
30:573, 40:613, 40:714, 40:721,
41:239, 41:262, 41:296, 42:238,
42:311, 42:314, 42:335, 42:344,
42:394, 42:402, 42:737, 42:793, 54:2428
- \@dir
01:144, 01:147, 01:149, 01:151, 01:152
- \@disable@packageload@do
..... 50:973, 52:477
- \@dischyp
. 40:449, 40:470, 06:927, 06:961, 1371
- \@docclearpage ... 54:301, 54:376, 1227
- \@documentclass 1364
- \@documentclasshook
... 50:3, 50:858, 50:886, 50:913, 1369
- \@doendpe 37:340, 37:355,
37:372, 37:382, 39:123, 39:159, 1436
- \@dofilelist 20:778,
20:820, 37:44, 37:109, 37:158, 1386
- \@dofilelist@hash 20:809, 20:815
- \@dofilelist@size 20:803, 20:815
- \@donoparitem 39:175, 39:189
- \@dot 42:615, 42:653
- \@dotsep 44:240, 44:263
- \@dottedtocline 44:225, 44:251, 44:252
- \@downline 42:308, 42:312, 42:317
- \@downvector 42:279, 42:317
- \@eha 03:195, 14:219,
14:237, 14:239, 14:241, 14:250,
14:252, 14:284, 20:243, 20:287,
20:305, 21:70, 21:102, 24:29, 24:59,
24:103, 24:145, 24:250, 24:316,
24:402, 26:107, 28:26, 28:71, 28:100,
28:173, 28:204, 28:234, 28:266,
28:550, 28:571, 28:623, 28:674,
28:719, 28:724, 28:779, 28:897,
28:901, 28:905, 28:940, 28:944,
28:948, 28:1005, 28:1015, 28:1100,
28:1105, 28:1108, 28:1140, 28:1143,
28:1216, 28:1219, 28:1222, 28:1289,
28:1295, 32:147, 33:24, 33:45,
33:876, 33:885, 37:252, 37:273,
37:291, 37:301, 06:384, 06:423,
06:451, 47:71, 47:88, 54:2453, 54:2469
- \@ehb 14:219, 14:244, 14:270, 14:272,
14:274, 14:276, 54:239, 54:395, 54:442
- \@ehc 03:184, 14:219,
14:279, 14:282, 14:288, 14:290,
06:169, 06:196, 37:657, 37:674,
37:688, 37:703, 37:716, 38:515,
39:251, 06:573, 06:614, 44:31, 50:1833
- \@ehd 03:102, 03:159, 14:219,
14:246, 14:254, 14:257, 14:259,
14:265, 28:119, 05:93, 41:100,
41:109, 45:6, 50:792, 50:812, 50:1056

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@elt 20:459, 21:1604, 21:1606,
22:37, 22:52, 22:157, 22:160, 22:177,
22:180, 29:162, 29:174, 29:176,
29:177, 29:202, 29:236, 29:245,
29:247, 29:248, 29:265, 29:584,
29:586, 29:587, 29:599, 29:899,
30:17, 30:25, 06:52, 54:4, 54:7,
54:11, 54:19, 54:21, 54:22, 54:23,
54:24, 54:27, 54:28, 54:29, 54:30,
54:31, 54:32, 54:33, 54:34, 54:36,
54:39, 54:495, 54:752, 54:1071,
54:1082, 54:1087, 54:1097, 54:1109,
54:1111, 54:1139, 54:1156, 54:1176,
54:1195, 54:1208, 54:1215, 54:1266,
54:1269, 54:1278, 54:2495, 54:2507, *742*
- \@empty *13:14, 94*
- \@emptycol *54:203,*
54:250, 54:253, 54:282, 54:286, 1400
- \@enc@info
. 21:7, 21:12, 21:103, 21:211, 21:221
- \@end@check@IncludeInRelease . . .
. 03:143, 03:145
- \@end@tempboxa
40:36, 40:55, 40:64, 40:263, 40:275,
40:389, 40:416, 40:437, 40:693, 40:703
- \@enddocument@kernel@warnings . . .
. 37:45, *37:54*, 37:110, 37:119, 37:178
- \@enddocumenthook
. *37:147, 50:1158, 50:1180*
- \@endfloatbox
. *45:190, 45:211, 45:236, 45:248*
- \@endparenv 39:120, *39:123, 917*
- \@endparpenalty 18:16, 38:561,
38:573, 38:599, 39:23, 39:124, 909
- \@endpbox 41:245, 41:268,
41:313, 41:343, 41:410, 41:461, 41:464
- \@endpefalse
. 37:258, 37:278, 37:296, 37:306,
39:129, 39:133, 39:137, 39:138,
39:140, 40:138, 40:144, 40:196, 917
- \@endpetrue
. 39:124, 39:126, 39:136, *39:140, 919*
- \@endpreamblehook *489*
- \@endtheorem
. 43:13, 43:19, 43:29, 43:38, *43:62*
- \@enlargepage *54:2438, 54:2443, 54:2445*
- \@ensuredmath 38:526, *38:528*
- \@enumctr 39:265, 39:268, 39:269
- \@enumdepth
. *39:257, 39:263, 39:264, 39:265, 922*
- \@enumspacing *922*
- \@eqcnt *38:415, 38:512,*
38:517, 38:614, 38:629, 38:630, 38:632
- \@eqncr 38:432,
38:461, 38:481, 38:518, 38:519, 38:616
- \@eqnnum 38:400, 38:407,
38:410, 38:516, 38:547, 38:606, 1360
- \@eqnset *38:415, 38:628, 1362*
- \@eqnswfalse 38:480
- \@eqnswtrue
38:417, 38:428, 38:457, 38:517, 38:613
- \@eqpen
38:415, 38:484, 38:486, 38:497, 38:507
- \@er@ext *1363*
- \@err@ 14:37, 14:41,
14:44, 14:52, 14:64, 14:68, 14:71, 14:79
- \@esphack 18:39,
18:132, 18:414, 18:431, 18:448,
18:465, 35:108, 35:123, 35:130,
36:86, 44:186, 44:187, 45:385, 46:17,
46:19, 46:34, 47:74, 47:90, 54:2461, 849
- \@evenfoot
. 49:12, 49:15, 54:897, 54:968, 54:1027
- \@evenhead
. 49:12, 49:15, 54:896, 54:967, 54:1026
- \@execute@begin@hook
. 37:255, 37:259, 37:262, *873*
- \@executeoption *1363*
- \@expandtwoargs
. 03:188, *06:258, 50:259,*
50:385, 50:580, 50:616, 50:670, 50:679
- \@expast *41:316, 41:344*
- \@expl@@@filehook@clear@replacement@flag@@
. 50:484,
50:507, 52:295, 52:318, 52:339, 52:519
- \@expl@@@filehook@drop@extension@@N
. 52:292, 52:293, 52:315,
52:316, 52:336, 52:337, 52:521, 1174
- \@expl@@@filehook@file@pop@@ . . .
. 50:991, 52:157, 52:525, 52:535
- \@expl@@@filehook@file@pop@assign@@nnnn
. 52:162, 52:527, *1170*
- \@expl@@@filehook@file@push@@ . . .
. 50:974, 52:151, 52:523
- \@expl@@@filehook@if@file@replaced@@TF
. 52:289, 52:312, 52:333, 52:517, *1175*
- \@expl@@@filehook@if@no@extension@@nTF
. 52:285,
52:308, 52:329, 52:507, 52:509, 52:534
- \@expl@@@filehook@normalize@file@name@@w
. 52:291, 52:314, 52:335, 52:515
- \@expl@@@filehook@resolve@file@subst@@w
. 50:482,
50:505, 52:288, 52:311, 52:332, 52:513
- \@expl@@@filehook@set@curr@file@@nNN
. 50:481, 50:504,
50:943, 52:374, 52:380, 52:511, 1122

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\@expl@@hook@curr@name@pop@@ ...	\@expl@str@map@function@@NN and
..... 08:2928, 50:106	\@expl@char@generate@@nn ... 1418
\@expl@@hook@curr@name@push@@n 1419	\@expl@sys@load@backend@@
\@expl@@initialize@call@@	 20:17, 05:25, 05:27
..... 08:2928, 37:263, 08:1491	\@expl@text@uppercase@@n 1425
\@expl@@mark@update@dbcol@structures@@	\@extra@page@added 37:70, 37:135, 53:440
..... 48:515, 48:528, 54:473	\@failedlist 54:1193,
\@expl@@mark@update@singlecol@structures@@	54:1216, 54:1232, 54:1239, 54:1252,
..... 48:513, 48:527, 54:476	54:1258, 54:1274, 54:1288, 54:1311
\@expl@@@shipout@add@background@box@@n	\@fcolmadefalse 54:1184
..... 53:470, 53:576	\@fcolmadetrue 54:1272
\@expl@@@shipout@add@background@picture@@n	\@file-subst@<file> 1171
..... 53:470, 53:580	\@file@und . 20:510, 20:522, 20:523,
\@expl@@@shipout@add@firstpage@material@@Nn	20:526, 20:532, 20:564, 20:586,
..... 53:470, 53:573	20:609, 20:626, 20:639, 20:692,
\@expl@@@shipout@add@foreground@box@@n	52:155, 52:172, 52:188, 52:196, 1419
..... 53:470, 53:578	\@filehook@file@push 1124
\@expl@@@shipout@add@foreground@picture@@n	\@filehook@set@CurrentFile
..... 53:470, 53:582	20:334, 20:395, 50:976, 52:152, 52:363
\@expl@char@generate@@nn	\@filelist .. 20:63, 20:132, 20:187,
..... 05:159, 06:906, 1418	20:760, 20:761, 20:780, 29:906,
\@expl@cs@<thing>@spec@@N 1417	29:916, 29:926, 57:281, 57:774, 57:792
\@expl@cs@argument@spec@@N ... 1432	\@fileswfalse 20:200,
\@expl@cs@parameter@spec@@N ...	50:1271, 50:1272, 50:1406, 50:1407
... 05:154, 05:155, 05:156, 05:167,	\@fileswith@ptions 50:532,
06:749, 06:802, 06:813, 06:818, 1432	50:640, 50:659, 50:844, 50:845,
\@expl@cs@prefix@spec@@N	50:849, 50:851, 50:879, 50:880,
..... 05:152, 05:166, 06:747, 06:817	50:906, 50:907, 50:934, 50:1147, 1362
\@expl@cs@replacement@spec@@N ...	\@fileswith@ptions 50:829,
..... 05:157, 05:168, 06:707,	50:830, 50:837, 50:838, 50:842, 50:846
06:742, 06:749, 06:803, 06:814, 06:818	\@fileswith@ptions 50:818
\@expl@cs@to@str@@N	 50:712, 50:719, 50:727, 50:818
... 05:147, 05:150, 05:162, 05:164,	\@fileswtrue 20:5,
06:581, 06:673, 06:693, 06:695,	50:1254, 50:1257, 50:1314, 50:1318,
06:696, 06:709, 06:722, 06:734,	50:1389, 50:1392, 50:1447, 50:1451
06:738, 06:739, 06:744, 06:788,	\@finalstrut
06:797, 06:801, 06:809, 06:811, 1417	... 40:553, 40:570, 40:587, 40:704,
\@expl@finalise@setup@@	41:462, 45:539, 45:557, 45:575, 1374
..... 05:34, 57:268, 57:270	\@firstampfalse 41:331, 41:354, 41:371
\@expl@finalize@setup@@	\@firstamptrue 41:339
..... 05:35, 57:269, 57:271	\@firstcolfirstmark
\@expl@pop@filename@@ ... 05:30,	 54:2916, 54:2917, 54:2921
50:105, 50:109, 50:115, 50:126, 1417	\@firstcoltopmark ... 54:2914, 54:2922
\@expl@push@filename@@ ... 05:28,	\@firstcolumnfalse
50:39, 50:52, 50:55, 50:69, 50:71,	 54:2878, 54:2907, 54:2948
50:81, 50:93, 50:113, 50:119, 1100	\@firstcolumntrue
\@expl@push@filename@aux@@	 20:28, 20:98, 20:156,
..... 08:2826, 05:29, 50:49, 50:52,	54:76, 54:212, 54:2882, 54:2925, 54:2954
50:64, 50:77, 50:81, 50:99, 50:119, 1417	\@firstoffive 35:23, 35:25, 35:40, 35:57
\@expl@str@if@eq@@nnTF ... 05:151,	\@firstofone
05:165, 06:545, 06:547, 50:385, 1417	... 04:12, 04:100, 04:108, 04:166,
\@expl@str@map@function@@NN	17:7, 17:12, 20:126, 20:181, 20:478,
..... 05:158, 05:169, 06:903, 111	21:86, 21:171, 26:363, 05:39, 05:66,
	28:54, 28:82, 05:81, 28:147, 28:185,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=lt page.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

28:215, 28:246, 28:1050, 05:216,
 05:224, 06:253, 37:146, 38:524,
 41:449, 06:569, 06:574, 06:575,
 06:578, 06:579, 06:610, 06:615,
 06:616, 06:619, 06:620, 45:10, 47:38,
 47:66, 47:83, 50:322, 50:324, 50:326,
 50:328, 50:330, 50:332, 50:334,
 50:336, 50:344, 50:992, 50:1021,
 52:352, 52:369, 57:321, 57:408, *1388*
 \@firstoftwo . 03:85, 17:11, 20:525,
 20:531, 20:565, 20:610, 20:627,
 21:151, 22:358, 22:363, 24:219,
 24:239, 05:40, 05:48, 28:1054,
 29:589, 33:798, 33:848, 33:864,
 35:42, 35:78, 06:253, 06:298, 06:319,
 06:546, 06:548, 06:570, 06:611,
 06:676, 06:727, 49:16, 50:196,
 50:230, 50:242, 06:842, 50:265,
 50:283, 50:342, 06:852, 50:372,
 06:862, 06:889, 06:913, 50:1842,
 54:644, 54:653, 54:660, 01:71, *503*
 \@firsttab 41:2, 41:74,
 41:75, 41:76, 41:106, 41:118, *1374*
 \@flcheckspace 54:1340, 54:1376, 54:2723
 \@flfail 54:1216, 54:1267,
 54:1288, 54:1298, 54:1311, 54:1320
 \@float 45:26, 45:32
 \@floatboxreset 45:101, 45:170, 45:174
 \@floatpenalty
 . 45:3, 45:53, 45:55, 45:58, 45:122,
 45:124, 45:127, 45:191, 45:194,
 45:199, 45:201, 45:212, 45:216,
 45:221, 45:223, 45:237, 45:241,
 45:311, 45:313, 45:317, 45:321, 45:379
 \@floatplacement . . . 20:31, 20:101,
 20:159, 45:271, 54:128, 54:214,
 54:258, 54:484, 54:2518, 54:2546, *1364*
 \@flsetnum 54:1337, 54:1373, 54:1461,
 54:1610, 54:1763, 54:1917, 54:1998,
 54:2084, 54:2155, 54:2276, 54:2691
 \@flsettextmin
 54:1436, 54:1587, 54:1733,
 54:1902, 54:1984, 54:2066, 54:2707
 \@flstop 54:2583
 \@flsucceed 54:1209,
 54:1217, 54:1266, 54:1300, 54:1322
 \@fltovf . 14:275, 45:93, 45:162, 45:322
 \@flupdates . 54:1343, 54:1388, 54:2824
 \@flushglue . 12:17, 37:454, 37:458,
 37:464, 37:474, 37:477, 37:482,
 37:532, 37:554, 39:76, 40:456, 40:477
 \@fnsymbol 22:308, 22:327
 \@font@aliasinfo 26:556
 \@font@info . 24:134, 24:172, 24:178,
 24:201, 24:202, 24:463, 24:480,
 24:718, 25:2866, 26:31, 26:39, 26:47,
 26:75, 26:88, 26:155, 26:201, 26:215,
 26:226, 26:240, 26:256, 26:262,
 26:275, 26:282, 26:289, 26:294,
 26:304, 26:316, 26:328, 26:508,
 26:520, 26:525, 26:532, 26:562,
 26:575, 26:583, 28:280, 28:293,
 28:304, 28:309, 28:314, 28:328,
 28:345, 28:354, 28:373, 28:388,
 28:423, 28:438, 28:498, 28:554,
 28:653, 28:659, 28:703, 28:716,
 28:799, 28:888, 28:931, 28:996,
 28:1090, 28:1257, 28:1286, 33:63, 57:527
 \@font@series@contextfalse
 29:581, 29:621
 \@font@series@contexttrue
 29:602, 29:606, 29:620
 \@font@shape@subst@warning
 25:2833, 25:2836,
 25:2841, 25:2899, 25:3178, 25:3181, *668*
 \@font@warning 24:4,
 24:632, 24:637, 24:664, 24:671,
 25:2844, 26:20, 26:34, 26:42, 26:50,
 26:62, 26:78, 26:493, 26:507, 26:519,
 26:524, 26:531, 26:574, 26:582,
 27:30, 37:56, 37:121, 37:160, 57:302
 \@fontenc@load@list
 . . 21:1605, 29:899, 30:17, 30:25, *1415*
 \@fontswitch 32:127, 32:129, *1374*
 \@footnotemark 45:516,
 45:522, 45:584, 45:590, 45:591, 45:617
 \@footnotetext 40:518,
 45:516, 45:522, 45:523, 45:600, 45:606
 \@for 13:16, 20:252,
 20:324, 20:386, 20:433, 20:780,
 47:36, 47:65, 47:82, 50:267, 50:284,
 50:578, 50:597, 50:614, 50:632,
 50:637, 50:651, 50:656, 50:692,
 50:702, 50:1225, 50:1262, 50:1397, *1054*
 \@forced@seriesfalse
 25:2782, 25:2795, 26:141
 \@forced@seriestrue . 25:2786, 25:2797
 \@forloop 13:19, 13:20
 \@fornoop 13:15, 13:23, 13:29
 \@fortmp
 13:17, 13:18, 13:26, 50:690, 50:692,
 50:1261, 50:1262, 50:1396, 50:1397
 \@fpbot 45:290, 45:304, 54:1214, 54:3014
 \@fpmin 45:278,
 45:287, 45:301, 54:91, 54:1271,
 54:2526, 54:2554, 54:2841, 54:2858

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`\@fps` 45:41, 45:42, 45:44, 45:47, 45:64,
 45:110, 45:111, 45:113, 45:116,
 45:133, 54:2614, 54:2616, 54:2619
`\@fpsadddefault`
 . 45:45, 45:48, 45:114, 45:117, 54:2611
`\@fpsep` ... 45:289, 45:303, 54:1212,
 54:1221, 54:1293, 54:1315, 54:3014
`\@fpstype` 54:1334, 54:1355, 54:1356,
 54:1370, 54:1401, 54:1402, 54:1426,
 54:1428, 54:1431, 54:1433, 54:1485,
 54:1541, 54:1542, 54:1577, 54:1579,
 54:1582, 54:1584, 54:1634, 54:1686,
 54:1687, 54:1721, 54:1724, 54:1727,
 54:1730, 54:1791, 54:1853, 54:1854,
 54:1892, 54:1894, 54:1897, 54:1899,
 54:1974, 54:1976, 54:1979, 54:1981,
 54:2054, 54:2057, 54:2060, 54:2063,
 54:2152, 54:2167, 54:2169, 54:2187,
 54:2196, 54:2232, 54:2233, 54:2273,
 54:2288, 54:2290, 54:2310, 54:2320,
 54:2359, 54:2360, 54:2597, 54:2623,
 54:2625, 54:2627, 54:2630, 54:2631,
 54:2632, 54:2634, 54:2635, 54:2639,
 54:2640, 54:2642, 54:2643, 54:2677,
 54:2679, 54:2681, 54:2693, 54:2695,
 54:2709, 54:2711, 54:2746, 54:2749,
 54:2760, 54:2796, 54:2799, 54:2810
`\@fptop` 45:288, 45:302, 54:1211, 54:3014
`\@frameb@x`
 . 40:223, 40:262, 40:274, 40:278, 1380
`\@framebox` 40:230, 40:237, 40:241
`\@framepicbox` .. 40:230, 40:237, 40:299
`\@freelist` 45:60, 45:129,
 45:319, 45:320, 54:20, 54:25, 54:37,
 54:218, 54:496, 54:753, 54:1083,
 54:1098, 54:1112, 54:1217, 54:2389,
 54:2390, 02:119, 02:137, 02:161, 1241
`\@generic@error` 1379
`\@generic@message` 1379
`\@getcirc` 42:448,
 42:495, 42:524, 42:551, 42:623, 42:639
`\@getfpsbit` 54:1331,
 54:1367, 54:2149, 54:2270, 54:2650
`\@getlarrow` 42:277, 42:285, 42:287
`\@getlinechar` 42:201, 42:240
`\@getpen` ... 18:35, 18:38, 18:47, 18:118
`\@getrarrow` 42:278, 42:285, 42:294
`\@glossaryfile` ... 46:21, 46:22, 46:31
`\@gnewline` 18:96, 18:102, 18:109, 18:112
`\@gobble` 03:163, 03:187,
 04:11, 04:98, 13:6, 13:9, 14:101,
 14:127, 14:147, 14:155, 14:180,
 14:189, 14:202, 17:6, 17:13, 17:21,
 18:76, 18:581, 20:64, 20:133, 20:188,
 20:485, 20:487, 20:760, 21:47,
 24:633, 24:666, 26:362, 05:38, 27:26,
 05:70, 28:29, 28:31, 05:93, 28:502,
 28:513, 28:597, 28:664, 28:665,
 28:694, 28:700, 28:708, 28:713,
 28:731, 28:745, 28:755, 28:764,
 28:777, 28:794, 28:803, 28:877,
 28:879, 28:883, 28:891, 28:925,
 28:934, 28:986, 28:988, 28:999,
 28:1083, 28:1093, 28:1174, 28:1179,
 28:1248, 28:1279, 29:140, 29:202,
 29:265, 05:216, 05:224, 29:599,
 29:909, 29:919, 29:929, 06:129,
 33:825, 06:174, 06:249, 06:272,
 37:283, 06:340, 06:344, 06:381,
 06:387, 06:390, 06:400, 06:420,
 06:426, 06:429, 06:438, 06:448,
 06:454, 06:457, 06:466, 06:484,
 06:488, 06:490, 06:491, 06:493,
 06:501, 06:505, 06:507, 44:143,
 44:144, 44:145, 44:146, 44:147,
 44:200, 45:7, 47:11, 47:45, 47:46,
 50:322, 50:324, 50:326, 50:328,
 50:330, 50:332, 50:334, 50:336,
 50:344, 50:780, 50:971, 50:1246,
 50:1310, 50:1337, 50:1441, 50:1470,
 50:1554, 50:1559, 50:1633, 50:1787,
 50:1799, 52:261, 54:974, 54:975,
 54:976, 54:1033, 54:1034, 54:1035,
 54:1278, 54:2509, 54:2842, 54:2859,
 57:284, 57:460, 57:792, 57:796, 1375
`\@gobble@AddToHook@args`
 08:2944, 08:2945
`\@gobble@IncludeInRelease` 03:71
`\@gobble@RemoveFromHook@arg`
 08:2947, 08:2948
`\@gobble@om` 35:102, 35:104,
 44:180, 44:190, 44:192, 44:203, 90
`\@gobble@som` 35:103,
 44:180, 44:191, 44:197, 44:204, 90
`\@gobble@with@sphack@om`
 ... 44:180, 44:205, 54:904, 54:906, 90
`\@gobble@with@sphack@som`
 44:180, 44:206, 54:905, 90
`\@gobblecr` 18:579, 18:580
`\@gobblefour` 28:25, 28:499,
 28:655, 28:657, 28:661, 28:663,
 28:673, 28:677, 28:801, 28:853,
 06:249, 50:1339, 50:1472, 50:1561
`\@gobblethree` 06:249,
 06:736, 06:748, 50:163, 57:323, 1103
`\@gobbletwo` 13:12,
 20:32, 20:102, 20:160, 24:638,
 24:672, 28:133, 05:128, 05:133,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- 06:216, 06:217, 06:249, 37:24, 37:57,
37:91, 37:122, 37:153, 37:161, 49:11,
49:13, 50:1309, 50:1440, 50:1553,
52:260, 52:489, 57:308, 57:363, *1399*
\@gtempa 20:704,
20:705, 20:711, 20:712, 20:713,
20:738, 20:739, 20:741, 20:742,
20:743, 06:166, 06:168, 06:222,
06:224, 41:3, 41:5, 41:6, 41:7,
41:8, 06:667, 06:675, 50:384, 50:386,
50:399, 50:400, 50:419, 50:421,
50:435, 50:437, 50:447, 50:449, *1370*
\@halfwidth 42:2,
42:137, 42:140, 42:142, 42:238,
42:310, 42:313, 42:335, 42:344,
42:360, 42:372, 42:375, 42:394,
42:402, 42:416, 42:427, 42:432,
42:690, 42:716, 42:735, 42:736,
42:737, 42:776, 42:791, 42:792, 42:793
\@halignto 41:174,
41:181, 41:198, 41:201, 41:243, 41:266
\@hangfrom 44:66, 44:117, 44:138
\@height 18:419,
18:427, 18:453, 18:461, 21:313,
21:315, 26:191, 06:26, 30:351,
30:569, 30:570, 30:572, 30:573,
40:207, 40:212, 40:285, 40:295,
40:613, 40:714, 40:721, 41:238,
41:261, 41:296, 41:436, 41:453,
42:238, 42:311, 42:314, 42:335,
42:344, 42:362, 42:370, 42:394,
42:402, 42:418, 42:425, 42:600,
42:610, 42:736, 42:792, 54:2428, 02:377
\@highpenalty 18:119, 57:3
\@hightab 41:11,
41:21, 41:23, 41:74, 41:86, 41:95,
41:96, 41:111, 41:146, 41:147, *1393*
\@hline . 42:178, 42:190, 42:237, 42:276
\@holdpg 54:100, 54:305,
54:307, 54:308, 54:313, 54:314, 54:315
\@hspace . 18:534, 18:535, 18:551, *1361*
\@hspacer 18:534, 18:550, *1361*
\@hvector 42:255, 42:270, 42:276
\@ialph *1376*
\@icentercr 37:432, 37:433
\@iden 06:256
\@if 06:212, 06:213, 06:215
\@if.. *1376*
\@if@DeclareRobustCommand
..... 09:142, 06:589,
06:646, 06:657, 06:658, 06:662,
06:756, 06:757, 06:760, 06:783, *104*
\@if@bottomfloats@TF
..... 54:577, 54:656, 54:667,
54:676, 54:689, 54:712, 54:815, *1233*
\@if@flushbottom@TF .. 54:642, 54:813
\@if@footnotes@TF
..... 54:574, 54:649, 54:665,
54:683, 54:691, 54:708, 54:814, *1233*
\@if@newcommand .. 09:143, 09:154,
06:408, 06:590, 06:647, 06:658,
06:700, 06:712, 06:718, 06:784, *104*
\@if@newlist *1403*
\@if@pti@ns
50:259, 50:262, 50:264, 50:281, 50:282
\@if@options 50:256,
50:257, 50:258, 50:1006, 50:1101
\@if@short@command *1376*
\@ifatmargin 41:55, 41:106
\@ifbothcounters 22:185,
22:200, 22:208, 22:223, 22:238,
22:240, 22:261, 22:273, 22:286, 22:288
\@ifclasslater . 50:198, 50:206, 50:214
\@ifclassloaded 50:190, 50:303, 50:312
\@ifclasswith .. 50:256, 50:305, 50:314
\@ifdefinable
..... 21:32, 21:35, 22:11, 22:18,
23:3, 29:729, 06:102, 06:104, 06:171,
06:173, 06:295, 06:316, 06:342,
40:159, 43:7, 43:15, 43:26, 43:35, *1364*
\@iffileonpath
20:506, 20:560, 20:582, 20:599, 20:620
\@ifl@aded 50:190, 50:191,
50:192, 50:952, 50:1082, 50:1100, *1123*
\@ifl@t@r
.. 21:1566, 05:215, 05:223, 50:204,
50:207, 50:212, 50:215, 50:219,
50:223, 50:225, 50:236, 50:237, 50:801
\@ifl@ter 21:1611, 21:1612,
50:198, 50:199, 50:218, 50:994, 50:1129
\@ifl@ter@@ 21:1611, 21:1612
\@ifnch
.. 05:54, 05:55, 06:874, 06:876, 06:888
\@ifnextchar 18:94, 18:580, 20:652,
22:13, 26:428, 05:50, 37:431, 38:413,
39:174, 40:9, 40:11, 40:18, 40:20,
40:26, 40:68, 40:165, 40:166, 40:172,
40:173, 40:180, 40:184, 40:229,
40:230, 40:236, 40:237, 40:242,
40:300, 40:308, 40:316, 40:323,
40:327, 40:488, 40:492, 40:496,
40:597, 40:602, 40:673, 40:680,
40:685, 41:57, 41:222, 41:227,
41:280, 41:287, 42:34, 42:143,
42:154, 42:464, 43:3, 43:5, 43:46,
43:52, 45:27, 45:264, 45:324, 45:514,
45:581, 45:598, 47:3, 47:17, 47:29,
50:391, 50:405, 06:870, 06:875,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 06:889, 50:796, 50:828, 50:836,
50:843, 50:1255, 50:1258, 50:1390,
50:1393, 54:214, 54:2585, 01:82, [1404](#)
`\@iforloop` [13:21](#), [13:22](#)
`\@ifpackagelater`
..... [50:198](#), 50:205, 50:213, [1405](#)
`\@ifpackageloaded`
..... [50:190](#), 50:302, 50:311, 54:2569, [1097](#)
`\@ifpackagewith`
..... [50:256](#), 50:304, 50:313, [1097](#)
`\@ifframebox` [40:243](#), 40:244, [40:245](#)
`\@ifframepicbox` [40:300](#), [40:301](#)
`\@ifstar` 18:60, 18:72, 18:399, 18:534,
22:236, 22:284, 24:269, 27:121,
06:78, 06:87, 37:420, 37:427, 37:727,
37:736, 38:483, 41:56, 41:279,
41:286, 42:153, 42:615, 44:52,
44:142, 50:533, 50:575, [06:889](#), 54:2433
`\@ifundefin@d@i`
..... 06:832, 06:833, 06:850, 06:853
`\@ifundefin@d@ii` 06:832, 06:835, 06:838
`\@ifundefined` ... 22:3, 22:7, 22:15,
22:33, 22:58, 22:60, 22:115, 22:122,
22:123, 22:152, 22:153, 22:154,
22:174, 22:189, 22:191, 24:101,
24:200, 24:249, 26:441, 28:544,
05:146, 33:827, 06:168, 06:175,
06:195, 06:202, 06:224, 35:85,
06:235, 37:218, 37:235, 37:251,
37:272, 37:290, 37:300, 06:381,
06:387, 06:420, 06:426, 06:448,
06:454, 06:484, 06:501, 06:582,
06:622, 06:623, 43:25, 43:34, 47:40,
47:68, 47:85, 49:3, 49:7, [06:827](#),
50:163, 50:188, 50:486, 50:492,
50:509, 50:521, 50:599, 50:633,
50:652, 50:954, 52:484, 07:3329, [1375](#)
`\@ignore...` [1397](#)
`\@ignorefalse` 24:433, [37:4](#), 37:257,
37:277, 37:295, 37:305, 37:341,
37:358, 37:374, 37:383, 45:384, [1360](#)
`\@ignoretrue` 18:223, 18:236,
24:428, [37:4](#), 37:7, 38:392, 38:400,
38:407, 38:446, 38:475, 38:637, [1396](#)
`\@iiiminipage`
..... [40:490](#), 40:494, 40:497, 40:498, [40:499](#)
`\@iiiparbox` [40:310](#), 40:318,
40:325, 40:328, 40:329, [40:365](#), 40:535
`\@iiiminipage` [40:493](#), [40:495](#)
`\@iininput` [20:652](#), [20:653](#), [504](#)
`\@iiparbox` [40:324](#), [40:326](#)
`\@iirsbox` [40:685](#), [40:694](#)
`\@imakebox` [40:26](#), [40:41](#), 40:182
`\@imakepicbox`
..... [40:68](#), [40:69](#), 40:187, 40:302
`\@iminipage` [40:489](#), [40:491](#)
`\@in@minipage@envtrue` [40:514](#)
`\@include`
..... [20:247](#), 20:290, 20:306, [20:310](#), [1416](#)
`\@includeinreleasefalse`
..... 03:74, 03:79, 03:133, 03:141, 06:965
`\@includeinreleasetrue` 03:123
`\@index` [46:18](#), [46:19](#), 46:35
`\@indexfile` [46:4](#), 46:5, 46:14
`\@inlabeledfalse` [39:28](#), 39:104,
39:215, 54:142, 54:169, 54:195, [1368](#)
`\@inlabeledtrue` [39:28](#), 39:209
`\@inmatherr`
..... [14:285](#), 39:112, 39:173, 42:615, [1373](#)
`\@inmathwarn` [21:3](#)
`\@inpenc@test` 57:405, 57:472
`\@input` [20:34](#),
20:104, 20:162, 20:318, 20:380,
20:427, [20:691](#), 44:152, 57:805, [1160](#)
`\@input@`
..... [20:338](#), 20:365, 20:398, 20:417,
20:442, [20:693](#), 24:490, 47:51, [1376](#)
`\@input@file@exists@with@hooks` ..
..... [52:143](#), [1419](#)
`\@inputcheck` [20:501](#), 20:502,
20:509, 20:555, 20:556, 20:563,
20:577, 20:578, 20:585, 20:607,
20:608, 20:611, 20:624, 20:625,
20:628, 01:175, 01:176, 01:179,
01:187, 05:43, 05:44, 05:47, 06:38,
06:45, 50:1298, 50:1299, 50:1333,
50:1429, 50:1430, 50:1466, 50:1541,
50:1542, 50:1549, [02:177](#), 01:54, [1421](#)
`\@insertfalse`
..... 54:1424, 54:1575, 54:1719, 54:1890,
54:1972, 54:2052, 54:2147, 54:2268
`\@inserttrue`
..... 54:1350, 54:1395, 54:1513,
54:1662, 54:1825, 54:2226, 54:2353
`\@invalidchar` [14:290](#)
`\@iparbox` [40:309](#), 40:317, [40:322](#)
`\@irsbox` [40:673](#), 40:680, 40:685, [40:686](#)
`\@isavebox` [40:180](#), [40:181](#)
`\@isavepicbox` [40:185](#), [40:186](#)
`\@ishortstack` [42:144](#), [42:152](#)
`\@istackcr` [42:154](#), [42:155](#)
`\@itabcr` [41:57](#), [41:58](#)
`\@item` [39:174](#), [39:187](#)
`\@itemdepth`
..... [39:272](#), 39:274, 39:275, 39:276, [922](#)
`\@itemfudge` [41:38](#), 41:44, 41:82
`\@itemitem` [39:276](#), 39:279

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- \@itemlabel . 39:44, 39:96, 39:174, 1365
- \@itempenalty 18:17, 39:23, 39:206, 909
- \@itemspacing 922
- \@iwhiledim 13:7
- \@iwhilenum 13:3
- \@iwhilesw 13:10
- \@ixpt 24:816
- \@ixstackcr 42:153
- \@kernel@... 1100
- \@kernel@Ref 35:202, 35:207
- \@kernel@after@{hook} 238
- \@kernel@after@begindocument 09:77, 20:58, 20:76, 05:7, 51:265, 1421
- \@kernel@after@begindocument@before 20:16, 20:76, 25:3251, 489
- \@kernel@after@endddocument 05:1, 37:15, 37:83, 53:399
- \@kernel@after@endddocument@afterlastpage 05:1, 37:20, 37:87, 53:410, 08:455
- \@kernel@after@para@after 16:8, 16:108, 444
- \@kernel@after@para@end 16:8, 16:101, 443
- \@kernel@after@shipout@background 53:133, 53:221, 1195
- \@kernel@after@shipout@lastpage 53:175, 53:180, 53:221, 53:430, 53:436, 1207
- \@kernel@before@{hook} 238
- \@kernel@before@begindocument 20:56, 20:76, 05:7, 53:460, 1421
- \@kernel@before@endddocument 37:13, 37:81, 37:182, 869
- \@kernel@before@endddocument@afterlastpage 05:1, 37:18, 53:405, 1437
- \@kernel@before@insertmark 48:251, 48:278, 1078
- \@kernel@before@para@before 16:8, 16:21, 16:54, 444
- \@kernel@before@para@begin 16:8, 16:30, 16:62, 444
- \@kernel@before@shipout@background 53:129, 53:131, 53:221, 1195
- \@kernel@currrpathstack 50:46, 50:48, 50:61, 50:63, 50:108, 50:158, 1103
- \@kernel@eqno . . 38:536, 38:538, 38:544
- \@kernel@get@ctr@root 22:114, 22:129, 22:131, 22:166, 22:168
- \@kernel@leqno . 38:537, 38:539, 38:545
- \@kernel@make@file@csname 52:287, 52:290, 52:310, 52:313, 52:331, 52:334, 52:363
- \@kernel@new@label@record@testdef 36:171, 36:302, 37:26
- \@kernel@pageref 35:65, 35:69
- \@kernel@pageref@exp 35:70
- \@kernel@ref 35:64, 35:67, 35:202
- \@kernel@ref@exp 35:72
- \@kernel@refstepcounter 35:154, 35:171, 43:45, 1431
- \@kernel@rename@newcommand 06:389, 06:406, 06:443, 06:471, 06:476, 06:489, 96
- \@kernel@reserved@label@data 35:106, 35:114, 35:121
- \@kernel@sRef 35:204, 35:207
- \@kernel@spageref 35:26, 35:43, 35:60, 35:65, 35:69
- \@kernel@sref . 35:13, 35:25, 35:30, 35:42, 35:47, 35:59, 35:64, 35:67, 35:204
- \@kernel@whatsit . 17:63, 17:65, 17:118
- \@killglue 42:70, 42:84, 42:114, 42:126, 42:134
- \@kludgeins . 54:324, 54:325, 54:326, 54:328, 54:381, 54:382, 54:428, 54:429, 54:499, 54:521, 54:522, 54:523, 54:527, 54:542, 54:546, 54:556, 54:756, 54:778, 54:787, 54:794, 54:821, 54:822, 54:2429, 54:2460
- \@labels 39:27, 39:177, 39:178, 39:220, 39:237, 39:238, 908
- \@largefloatcheck 45:192, 45:213, 45:238, 45:256, 1374
- \@lastchclass 41:339, 41:349, 41:350, 41:352, 41:360, 41:385, 41:399, 41:403, 41:412, 41:425, 41:426
- \@latex@error . . . 03:100, 08:2971, 03:159, 03:184, 03:195, 14:163, 14:217, 14:233, 14:239, 14:241, 14:244, 14:246, 14:248, 14:252, 14:254, 14:256, 14:259, 14:263, 14:268, 14:272, 14:274, 14:276, 14:278, 14:279, 14:281, 14:284, 14:288, 14:290, 17:16, 20:243, 20:287, 20:305, 21:68, 21:102, 24:6, 24:29, 24:59, 24:103, 24:145, 24:250, 24:316, 24:402, 26:106, 27:100, 27:111, 28:24, 28:69, 28:98, 28:118, 05:85, 28:171, 28:202, 28:232, 28:264, 28:384, 28:400, 28:434, 28:450, 28:550, 28:571, 28:623, 28:673, 28:677, 28:719, 28:724, 28:779, 28:847, 28:853, 28:897, 28:901, 28:905, 28:940, 28:944, 28:948, 28:1005, 28:1015, 28:1100, 28:1105, 28:1108, 28:1140, 28:1143,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 28:1216, 28:1219, 28:1222, 28:1289,
28:1295, 29:50, 29:61, 29:83, 29:94,
29:711, 29:876, 32:144, 06:169,
06:196, 37:252, 37:273, 37:291,
37:301, 37:657, 37:674, 37:688,
37:703, 37:715, 06:382, 38:515,
06:421, 39:250, 06:449, 41:100,
41:109, 06:573, 06:614, 44:31, 45:6,
45:83, 47:71, 47:88, 50:723, 50:774,
50:787, 50:808, 50:824, 50:1007,
50:1055, 50:1102, 50:1214, 50:1231,
50:1239, 50:1244, 50:1266, 50:1323,
50:1401, 50:1456, 50:1802, 50:1832,
54:239, 54:395, 54:2451, 54:2468
- `\@latex@info` 14:163,
14:208, 33:40, 06:277, 06:402,
06:440, 06:468, 06:927, 52:487, 433
- `\@latex@info@no@line`
. 14:163, 14:209, 53:150, 54:827, 1378
- `\@latex@note` 14:190, 24:34
- `\@latex@note@no@line`
14:190, 50:1280, 50:1300, 50:1306, 1422
- `\@latex@warning`
... 14:163, 14:215, 20:363, 20:415,
21:73, 22:72, 22:92, 22:109, 22:128,
22:130, 22:165, 22:167, 35:18, 35:35,
35:52, 36:233, 36:244, 36:250, 40:51,
40:257, 40:385, 40:411, 40:510,
42:460, 45:260, 47:42, 47:69, 47:86,
50:1362, 50:1369, 50:1495, 50:1501,
50:1584, 50:1590, 53:235, 54:2617
- `\@latex@warning@no@line`
..... 14:163, 14:216, 20:19,
20:89, 20:146, 20:758, 05:103, 35:8,
35:88, 35:89, 06:243, 36:155, 36:156,
37:64, 37:71, 37:129, 37:136, 37:168,
44:32, 50:387, 50:401, 50:802,
50:995, 50:1130, 50:1282, 50:1431,
50:1437, 50:1460, 50:1543, 50:1550,
50:1617, 50:1703, 53:140, 53:159,
53:421, 54:248, 54:280, 54:2404, 54:2683
- `\@latexbug` 14:277, 54:338, 54:2390
- `\@latexerr` 14:215, 54:442, 1375
- `\@latexinfo` 1373
- `\@latexrelease@catcode@null`
..... 07:6, 07:3330
- `\@lbbibitem` 47:3, 47:4
- `\@ldots` 30:515, 30:517
- `\@leftcolumn` 54:99, 54:2879, 54:2885,
54:2908, 54:2928, 54:2949, 54:2958
- `\@leftmark` 49:16, 49:119
- `\@let@token` 18:479,
18:480, 18:487, 05:54, 05:56, 32:84,
32:97, 38:256, 38:258, 38:261,
06:874, 06:877, 06:880, 06:888, 1392
- `\@lign` 38:208, 38:210
- `\@linechar` .. 42:201, 42:202, 42:203,
42:207, 42:208, 42:210, 42:215,
42:217, 42:218, 42:219, 42:220,
42:222, 42:226, 42:227, 42:230,
42:231, 42:236, 42:283, 42:680, 1404
- `\@linefnt` ... 42:135, 42:138, 42:201,
42:276, 42:284, 42:315, 42:318, 42:687
- `\@linelen` ... 42:175, 42:176, 42:187,
42:188, 42:214, 42:221, 42:230,
42:232, 42:237, 42:238, 42:239,
42:252, 42:253, 42:267, 42:268,
42:311, 42:314, 42:316, 42:317, 42:681
- `\@list` 909
- `\@listctr` ... 39:233, 39:256, 47:9, 1360
- `\@listdepth` 39:23,
39:35, 39:38, 39:43, 39:99, 40:519, 908
- `\@listfiles`
. 20:62, 20:131, 20:186, 20:772, 20:791
- `\@listi` 909
- `\@listii` 909
- `\@listvi` 909
- `\@lnbk` 1398
- `\@loadwithoptions`
..... 50:729, 50:751, 50:761, 50:770
- `\@lowpenalty` 18:118, 57:3
- `\@ltab` 41:71, 41:106
- `\@ltxnomath` 1373
- `\@m` 18:325, 18:358, 18:526,
18:531, 20:45, 20:115, 20:173, 39:80,
42:224, 42:228, 47:37, 02:21, 02:304,
02:306, 02:307, 02:373, 02:374, 421
- `\@mainaux`
20:3, 20:37, 20:38, 20:107, 20:108,
20:165, 20:166, 20:236, 20:318,
20:361, 20:380, 20:413, 20:427,
20:452, 37:23, 37:90, 37:152, 1384
- `\@make@normalcolbox`
..... 54:502, 54:508, 54:797
- `\@make@specialcolbox`
..... 54:500, 54:517, 54:798
- `\@makebox` 40:11, 40:20, 40:25
- `\@makecaption` 45:24, 1320
- `\@makecol` ... 54:266, 54:418, 54:465,
54:489, 54:490, 54:735, 54:737, 1242
- `\@makecol@handlesplitfootnotes` ..
..... 54:613, 54:628
- `\@makefcolumn`
... 54:398, 54:399, 54:407, 54:409,
54:445, 54:447, 54:455, 54:457,
54:2837, 54:2839, 54:2855, 54:2856
- `\@makefnmark` 45:400, 45:594

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@makefn</code>	40:552, 40:569, 40:586, 45:538, 45:556, 45:574
<code>\@makeother</code>	01:110, 24:503, 24:504, 24:505, 24:506, 24:507, 24:508, 24:509, 24:510, 24:511, 24:512, 24:513, 37:543, 37:564, 37:709, 37:724, 37:734, 50:473, 50:474, 06:892, 06:893, 50:1344, 50:1477, 50:1566, 54:863, 54:864, 54:865, 54:866, 54:867, 54:868, 54:869, 54:870, 54:871, 54:872, 54:873, 54:874, 01:60, 01:81
<code>\@makepicbox</code>	... 40:10, 40:19, 40:67, 42:377, 42:435
<code>\@makespecialcolbox</code>	... 54:757, 54:770, 1363
<code>\@marbox</code>	... 45:320, 45:322, 45:326, 45:330, 45:331, 45:379, 54:2389, 54:2399, 54:2402, 54:2410, 54:2412, 54:2413, 54:2415, 54:2416, 54:2417, 54:2426
<code>\@marginparreset</code>	45:343, 45:361, 45:370
<code>\@markright</code>	49:52, 49:75, 49:98, 49:121
<code>\@math@level</code>	24:404, 24:422, 24:429, 24:434, 28:288, 584
<code>\@maxdepth</code>	20:60, 20:129, 20:184, 54:67, 54:504, 54:603, 54:742, 54:767, 57:152, 1391
<code>\@maxtab</code>	41:2, 41:94, 1374
<code>\@medpenalty</code>	18:119, 57:3
<code>\@meta@family@list</code>	... 29:117, 29:162, 29:175, 29:236, 29:246, 29:289, 29:585, 741
<code>\@midlist</code>	... 54:42, 54:496, 54:497, 54:753, 54:754, 54:1378, 54:1380, 54:1493, 54:1642, 54:1801, 54:2533, 54:2561
<code>\@minipage...</code>	1396
<code>\@minipagefalse</code>	... 39:212, 40:483, 40:485, 40:532, 45:187, 45:250, 45:345, 45:363
<code>\@minipagerestore</code>	40:520, 40:522
<code>\@minipagetrue</code>	40:484, 45:186
<code>\@minus</code>	06:26, 54:3007, 54:3008, 54:3009, 54:3012, 54:3013
<code>\@missing@onefilewithoptions</code>	... 50:969, 50:1025, 50:1143
<code>\@missingfile@area</code>	... 20:665, 20:706, 20:719, 50:1027
<code>\@missingfile@base</code>	... 20:665, 20:707, 20:720, 50:1028
<code>\@missingfile@ext</code>	... 20:665, 20:708, 20:721, 50:1029
<code>\@missingfileerror</code>	20:660, 20:675, 20:685, 20:694, 50:1026, 50:1123, 1383
<code>\@mkboth</code>	49:11, 49:13
<code>\@mklab</code>	39:45, 39:171
<code>\@mkpream</code>	... 41:241, 41:264, 41:311, 41:339, 1325
<code>\@mparbottom</code>	45:387, 45:388, 54:96, 54:483, 54:2400, 54:2408, 54:2409, 54:2410, 54:2411
<code>\@mpargs</code>	40:503, 40:535
<code>\@mparswitchfalse</code>	54:80
<code>\@mpfn</code>	... 40:517, 45:514, 45:519, 45:603, 45:607
<code>\@mpfootins</code>	40:509, 40:526, 40:527, 40:530, 40:536, 40:543, 40:544, 40:561, 40:562, 40:578, 40:579
<code>\@mpfootnotetext</code>	40:518, 40:538
<code>\@mplistdepth</code>	40:519, 40:536
<code>\@multicnt</code>	41:447, 41:449, 41:450, 41:451, 41:458, 41:459, 41:460, 42:114, 42:115, 42:117, 42:126, 42:127, 42:129, 42:677, 42:714, 42:716, 42:717, 42:718, 42:720, 42:721, 42:727, 42:733, 42:744, 42:748, 42:774, 42:776, 42:778, 42:780, 42:781, 42:785, 42:789, 42:800, 42:804, 1393
<code>\@multiplelabels</code>	... 20:33, 20:103, 20:161, 35:87, 35:93, 36:154, 37:62, 37:68, 37:127, 37:133, 37:166, 37:172
<code>\@multipt</code>	42:97, 42:106, 42:109
<code>\@multispan</code>	41:448, 41:452, 41:456
<code>\@mypkg@name</code>	1150
<code>\@mypkg@other@name</code>	1150
<code>\@namedef</code>	... 20:465, 24:136, 24:137, 24:161, 24:203, 25:6, 25:1436, 25:2913, 26:435, 28:473, 28:477, 28:486, 06:50, 33:57, 33:830, 35:90, 37:289, 37:349, 37:366, 37:380, 37:586, 37:595, 38:519, 38:520, 41:199, 43:12, 43:13, 43:18, 43:19, 43:28, 43:29, 43:36, 43:37, 43:38, 50:1257, 50:1392, 52:482, 57:529, 57:530, 1378
<code>\@nameuse</code>	... 20:355, 20:411, 20:450, 20:464, 06:51, 43:36, 49:5, 50:963, 52:486, 54:888, 54:962, 54:1020, 1378
<code>\@nbitem</code>	39:199, 39:252
<code>\@ne</code>	02:16, 421
<code>\@needsf@rmat</code>	50:797, 50:800, 50:805
<code>\@needsformat</code>	50:785, 50:795, 50:799
<code>\@negargfalse</code>	42:197
<code>\@negargtrue</code>	42:196
<code>\@newcommand</code>	06:97, 06:98

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@newctr</code>	22:13, 22:32, 43:8	26:397, 26:399, 26:406, 26:417,
<code>\@newenv</code>	06:191, 06:192, 06:201	26:418, 26:420, 26:451, 26:456,
<code>\@newenva</code>	06:190, 06:189	06:521, 06:527, 42:13, 50:865,
<code>\@newenvb</code>	06:192, 06:191	50:866, 50:873, 50:893, 50:894, 50:901
<code>\@newfontswitch</code>	1367	<code>\@no@font@optfalse</code>
<code>\@newl@bel</code>		27:17, 27:129
	35:84, 37:25, 37:92, 37:154, 47:10, 1390	<code>\@no@lnbk</code>
<code>\@newline</code>	18:95, 18:97	18:9, 18:10, 18:41, 1398
<code>\@newlistfalse</code>	39:29, 39:33,	<code>\@no@pgbk</code>
	39:108, 39:213, 54:878, 54:955, 54:1013	18:7, 18:8, 18:33, 1391
<code>\@newlisttrue</code>	39:29, 39:33, 39:87	<code>\@nobreake</code>
<code>\@next</code>	45:60, 45:129,	1396
	45:319, 45:320, 54:5, 54:218, 54:316,	<code>\@nobreakefalse</code>
	54:1228, 54:1248, 54:2389, 02:161	18:121,
<code>\@nextchar</code>		18:123, 39:224, 44:94, 44:129,
	41:346, 41:347, 41:407, 41:408, 41:409	44:157, 45:182, 54:144, 54:171,
<code>\@nil</code>		54:197, 54:1502, 54:1651, 54:1812, 1397
	03:13, 03:19, 03:83, 03:104, 03:105,	<code>\@nobreaketrue</code>
	03:117, 03:118, 03:165, 03:166,	18:122, 44:126, 45:181
	03:167, 13:13, 13:19, 13:27, 19:14,	<code>\@nocnterr</code>
	01:145, 01:146, 20:266, 20:267,	14:240, 1372
	20:268, 21:107, 21:128, 21:1020,	<code>\@nocounterr</code>
	21:1024, 21:1091, 21:1103, 21:1105,	
	24:455, 24:466, 24:582, 24:597,	14:240, 22:4, 22:8, 22:16, 22:33,
	24:701, 24:704, 24:705, 24:713,	22:189, 22:191, 43:25, 43:34, 1372
	25:2819, 25:2821, 25:2853, 25:2855,	<code>\@nodocument</code>
	25:3165, 25:3167, 25:3191, 25:3193,	14:245, 16:145,
	26:367, 26:368, 26:370, 26:383,	20:68, 20:137, 20:192, 20:315,
	26:389, 26:393, 26:394, 26:430,	20:377, 37:243, 45:39, 45:108,
	26:451, 26:456, 26:536, 26:550,	54:135, 54:162, 54:188, 54:217, 1398
	27:26, 27:44, 27:53, 27:57, 28:41,	<code>\@noitemargfalse</code>
	28:643, 28:651, 28:684, 28:1300,	39:32, 39:231
	28:1302, 06:53, 06:54, 06:58, 06:63,	<code>\@noitemargtrue</code>
	32:59, 32:63, 06:166, 06:176, 06:222,	39:32, 39:174
	06:543, 06:544, 41:444, 41:445,	<code>\@noitemerr</code>
	06:725, 50:107, 50:108, 50:114,	
	50:122, 50:125, 50:136, 50:147,	14:280, 18:286, 18:303, 18:373,
	50:175, 50:226, 50:227, 50:238,	18:395, 39:69, 39:81, 39:107, 1433
	50:239, 50:249, 50:250, 50:252,	<code>\@noitemerror</code>
	50:482, 50:505, 06:868, 06:869,	1384
	50:549, 50:555, 50:671, 50:691,	<code>\@noligs</code>
	50:695, 50:701, 50:705, 50:920,	37:544,
	50:929, 50:946, 50:947, 50:1675,	37:565, 37:725, 37:735, 37:746, 1396
	50:1680, 50:1683, 50:1685, 50:1686,	<code>\@nolnbnk</code>
	50:1702, 50:1722, 50:1739, 50:1794,	1398
	50:1816, 50:1840, 52:168, 52:176,	<code>\@nolnerr</code>
	52:369, 52:411, 52:413, 57:794, 57:795	14:238,
<code>\@nmbrlistfalse</code>	39:33, 39:46, 39:91, 1360	18:43, 18:114, 37:419, 37:426, 1374
<code>\@nmbrlisttrue</code>	39:256	<code>\@nomath</code>
<code>\@nnil</code>	13:13, 13:20,	24:3,
	13:21, 13:22, 13:28, 20:267, 20:268,	24:400, 29:634, 29:669, 29:675,
	24:277, 24:281, 24:282, 24:283,	29:685, 29:724, 29:726, 29:748, 1358
	24:298, 26:180, 26:182, 26:362,	<code>\@noparitemfalse</code>
	26:364, 26:376, 26:378, 26:383,	39:30, 39:176
		<code>\@noparitemtrue</code>
		39:30, 39:66
		<code>\@nopalrlistfalse</code>
		39:31, 39:70
		<code>\@nopalrlisttrue</code>
		39:31, 39:67
		<code>\@nopgbk</code>
		1391
		<code>\@normalcr</code>
		18:53, 18:93, 40:482, 1413
		<code>\@normalsize</code>
		50:4, 50:5, 1368
		<code>\@normalsize_{check}</code>
		1371
		<code>\@noskipsecfalse</code>
		20:54, 20:124,
		20:179, 44:98, 54:137, 54:164, 54:190
		<code>\@noskipsectrue</code>
		44:38, 44:95
		<code>\@notdefinable</code>
		14:232,
		06:177, 06:178, 06:182, 06:570, 06:611
		<code>\@notprerr</code>
		14:283, 20:66, 20:135, 20:190
		<code>\@nthm</code>
		43:3, 43:4
		<code>\@nxttabmar</code>
		
		41:11, 41:21, 41:23, 41:25, 41:75,
		41:111, 41:112, 41:118, 41:119, 1393
		<code>\@obsoletefile</code>
		20:757

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@oddfoot</code>	49:11, 49:14, 49:15, 54:102, 54:893, 54:965, 54:1024
<code>\@oddhead</code>	49:11, 49:14, 54:101, 54:892, 54:965, 54:1024
<code>\@onefilewithoptions</code>	50:857, 50:861, 50:867, 50:885, 50:889, 50:895, 50:912, 50:916, 50:922, 50:938, 50:940, 50:1023, 50:1090, 50:1644, 1122
<code>\@onefilewithoptions@clashchk</code>	50:955, 50:1005
<code>\@onelevel@sanitize</code>	45:42, 45:111, 06:894, 1382
<code>\@onfilewithoptions</code>	1127
<code>\@onlypreamble</code>	.. 08:2852, 20:197, 20:206, 20:262, 20:759, 20:819, 21:41, 21:42, 21:79, 21:80, 21:84, 21:143, 21:163, 21:207, 21:208, 21:224, 24:18, 24:116, 24:118, 24:124, 24:140, 24:168, 24:183, 24:266, 24:271, 24:313, 24:609, 26:436, 27:28, 27:36, 27:42, 27:79, 27:83, 27:88, 27:93, 27:98, 27:108, 27:126, 27:127, 27:128, 27:134, 27:138, 27:142, 28:18, 28:20, 28:45, 28:47, 28:108, 28:117, 28:137, 28:466, 28:467, 28:480, 28:526, 28:574, 28:586, 28:588, 28:601, 28:626, 28:683, 28:685, 28:727, 28:766, 28:782, 28:859, 28:953, 28:962, 28:1018, 28:1021, 28:1024, 28:1044, 28:1057, 28:1111, 28:1146, 28:1157, 28:1171, 28:1225, 28:1245, 28:1249, 28:1313, 06:66, 32:141, 32:142, 33:831, 06:229, 06:231, 35:92, 06:240, 06:248, 46:12, 46:29, 47:64, 47:81, 48:16, 48:400, 50:10, 50:13, 50:102, 50:129, 50:155, 50:157, 50:189, 50:392, 50:455, 50:461, 50:526, 50:529, 50:530, 50:541, 50:542, 50:543, 50:570, 50:576, 50:589, 50:626, 50:644, 50:664, 50:684, 50:708, 50:713, 50:717, 50:720, 50:728, 50:749, 50:752, 50:762, 50:781, 50:794, 50:799, 50:805, 50:841, 50:846, 50:934, 50:1023, 50:1148, 50:1157, 50:1165, 50:1166, 50:1184, 50:1212, 50:1221, 50:1228, 50:1229, 50:1237, 50:1242, 50:1247, 50:1623, 50:1624, 50:1625, 50:1626, 50:1628, 51:229, 1391
<code>\@opargbegintheorem</code>	43:59, 43:62
<code>\@opcol</code>	54:267, 54:275, 54:399, 54:418, 54:447, 54:465, 54:470, 1067
<code>\@options</code>	50:663
<code>\@othm</code>	43:3, 43:21
<code>\@outerparskip</code>	... 39:1, 39:88, 39:117, 39:183, 39:253
<code>\@outputbox</code>	48:418, 48:421, 48:425, 48:450, 48:453, 48:457, 54:98, 54:492, 54:509, 54:511, 54:512, 54:528, 54:531, 54:536, 54:537, 54:544, 54:550, 54:552, 54:564, 54:602, 54:604, 54:605, 54:739, 54:741, 54:759, 54:761, 54:762, 54:771, 54:773, 54:774, 54:779, 54:781, 54:782, 54:788, 54:790, 54:928, 54:991, 54:1050, 54:1074, 54:1080, 54:1090, 54:1091, 54:1114, 54:1121, 54:1207, 54:1210, 54:1213, 54:1219, 54:1220, 54:2879, 54:2883, 54:2889, 54:2908, 54:2911, 54:2912, 54:2926, 54:2932, 54:2949, 54:2955, 54:2964, 1234
<code>\@outputbox@append</code>	54:525, 54:567, 54:573, 54:601, 54:609, 54:615, 54:638, 54:709, 54:713, 54:802, 1244
<code>\@outputbox@appendfil</code>	54:609, 54:666, 54:668, 54:677, 54:684, 54:690, 54:692, 1233
<code>\@outputbox@appendfootnotes</code>	54:611, 54:671, 54:675, 54:686, 54:695, 54:699, 54:703, 54:718, 54:803, 1250
<code>\@outputbox@attachbottomfloats</code>	54:629, 54:806, 1233
<code>\@outputbox@attachfloats</code>	54:629, 54:672, 54:679, 54:682, 54:694, 54:698, 54:704, 54:719, 54:804, 54:1069, 1233
<code>\@outputbox@attachttopfloats</code>	54:629, 54:805, 1233
<code>\@outputbox@depth</code> .	54:507, 54:511, 54:513, 54:525, 54:590, 54:591, 54:593, 54:596, 54:598, 54:604, 1245
<code>\@outputbox@reinsertskip</code>	54:565, 54:571, 54:669, 54:678, 54:685, 54:693, 54:700, 54:705, 54:717, 54:800, 1250
<code>\@outputbox@removebskip</code>	54:493, 54:563, 54:799, 1244
<code>\@outputdblcol</code>	54:474, 54:2874, 54:2876, 54:2904, 54:2905, 54:2945, 54:2946
<code>\@outputpage</code>	54:408, 54:457, 54:477, 54:841, 54:2892, 54:2897, 54:2935, 54:2939, 54:2971, 54:2979, 1254
<code>\@oval</code>	42:464, 42:481
<code>\@ovbtrue</code>	42:486, 42:514, 42:544

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@ovdx</code>	42:442 , 42:497 , 42:499 , 42:505 , 42:507 , 42:526 , 42:528 , 42:536 , 42:538 , 42:553 , 42:561 , 42:563 , 42:599 , 42:602 , 42:609 , 42:611 , 42:703 , 42:704 , 42:705 , 42:706 , 42:722 , 42:723 , 42:724 , 42:727 , 42:743 , 42:763 , 42:764 , 42:765 , 42:766 , 42:782 , 42:783 , 42:785 , 42:799	42:746 , 42:767 , 42:768 , 42:769 , 42:773 , 42:775 , 42:788 , 42:789 , 42:802
<code>\@ovdy</code>	42:442 , 42:498 , 42:500 , 42:506 , 42:507 , 42:527 , 42:529 , 42:537 , 42:538 , 42:554 , 42:562 , 42:563 , 42:574 , 42:579 , 42:587 , 42:591 , 42:710 , 42:711 , 42:712 , 42:713 , 42:728 , 42:729 , 42:730 , 42:733 , 42:747 , 42:770 , 42:771 , 42:772 , 42:773 , 42:786 , 42:787 , 42:789 , 42:803	<code>\@p@pfilename</code> 50:107 , 50:114 , 50:122 , 50:125 , 50:135 , 50:136 , 50:145 , 50:147 , 50:155
<code>\@ovhlinefalse</code>	 42:487 , 42:515	<code>\@p@pfilepath</code> .. 50:108 , 50:173 , 50:182
<code>\@ovhlinetrue</code>	 42:467 , 42:471 , 42:475 , 42:493 , 42:499 , 42:521 , 42:528	<code>\@p@pfilepath@aux</code> 50:174 , 50:175 , 50:183
<code>\@ovhorz</code>	 42:504 , 42:505 , 42:535 , 42:536 , 42:560 , 42:561 , 42:594	<code>\@pagedp</code> ... 54:95 , 54:313 , 54:318 , 54:1442 , 54:1740 , 54:2418 , 54:2428
<code>\@ovltrue</code>	 42:486 , 42:514 , 42:544	<code>\@pageht</code> 54:94 , 54:314 , 54:318 , 54:320 , 54:321 , 54:322 , 54:326 , 54:1441 , 54:1592 , 54:1739 , 54:2401 , 54:2408
<code>\@ovri</code>	40:33 , 42:442 , 42:496 , 42:525 , 42:552 , 42:574 , 42:587 , 42:603 , 42:612	<code>\@par</code> 15:3 , 15:5
<code>\@ovro</code>	 42:442 , 42:496 , 42:505 , 42:506 , 42:525 , 42:536 , 42:537 , 42:552 , 42:561 , 42:562 , 42:573 , 42:579 , 42:586 , 42:591 , 42:598 , 42:608 , 42:624 , 42:631 , 42:640 , 42:649	<code>\@parboxrestore</code> 40:373 , 40:398 , 40:424 , 40:482 , 40:516 , 40:547 , 40:565 , 40:582 , 45:19 , 45:100 , 45:169 , 45:342 , 45:360 , 45:532 , 45:551 , 45:569 , 54:224 , 54:879 , 54:956 , 54:1014 , 1397
<code>\@ovrtrue</code>	 42:486 , 42:514 , 42:544	<code>\@parboxto</code> 40:365 , 1405
<code>\@ovttrue</code>	 42:486 , 42:514 , 42:544	<code>\@parmoderr</code> 14:273 , 45:58 , 45:127 , 45:316
<code>\@ovvert</code>	 42:502 , 42:503 , 42:531 , 42:533 , 42:556 , 42:558 , 42:567	<code>\@parse@version</code> 03:83 , 03:104 , 03:105 , 03:117 , 03:118 , 03:165 , 03:166 , 03:167 , 50:232 , 50:238 , 50:239 , 50:249 , 50:1686 , 50:1702 , 50:1739 , 50:1816 , 50:1840
<code>\@ovvlinefalse</code>	 42:487 , 42:515	<code>\@parse@version@</code> 50:226 , 50:227 , 50:232 , 50:244
<code>\@ovvlinetrue</code>	 42:470 , 42:492 , 42:500 , 42:520 , 42:529	<code>\@parse@version@dash</code> 50:250 , 50:252 , 1410
<code>\@ovxx</code>	 42:442 , 42:490 , 42:492 , 42:493 , 42:497 , 42:503 , 42:504 , 42:518 , 42:520 , 42:521 , 42:526 , 42:533 , 42:535 , 42:547 , 42:549 , 42:553 , 42:558 , 42:560 , 42:598 , 42:608 , 42:700 , 42:701 , 42:702 , 42:706 , 42:715 , 42:716 , 42:725 , 42:726 , 42:727 , 42:742 , 42:760 , 42:761 , 42:762 , 42:766 , 42:775 , 42:776 , 42:784 , 42:785 , 42:798	<code>\@partaux</code> 20:3 , 20:242 , 20:286 , 20:304 , 20:328 , 20:330 , 20:331 , 20:351 , 20:390 , 20:392 , 20:393 , 20:407 , 20:437 , 20:439 , 20:440 , 20:446 , 20:458 , 20:460 , 20:463
<code>\@ovyy</code>	 42:442 , 42:491 , 42:492 , 42:493 , 42:498 , 42:505 , 42:519 , 42:520 , 42:521 , 42:527 , 42:536 , 42:548 , 42:549 , 42:554 , 42:561 , 42:571 , 42:584 , 42:707 , 42:708 , 42:709 , 42:713 , 42:715 , 42:731 , 42:732 , 42:733 ,	<code>\@partlist</code> 20:251 , 20:255 , 20:256 , 20:258 , 20:282 , 20:301 , 20:324 , 20:386 , 20:433
		<code>\@partswfalse</code> 20:6
		<code>\@partswtrue</code> ... 20:250 , 20:280 , 20:300
		<code>\@pass@options</code> 50:478 , 50:480 , 50:500 , 50:501 , 50:503 , 50:516 , 50:517 , 50:519 , 50:526 , 50:527 , 50:528 , 50:1034 , 50:1116
		<code>\@pboxswfalse</code> .. 40:396 , 40:422 , 40:501
		<code>\@pboxswtrue</code> 40:406 , 40:432
		<code>\@pdef</code> 1367
		<code>\@penup</code> 38:199 , 38:200
		<code>\@percentchar</code> ... 50:1336 , 50:1338 , 50:1340 , 50:1342 , 50:1469 , 50:1471 , 50:1473 , 50:1475 , 50:1558 , 50:1560 , 50:1562 , 50:1564 , 50:1612 , 01:90
		<code>\@pgbk</code> 1391

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@picbox
 . [42:6](#), [42:42](#), [42:54](#), [42:62](#), [42:63](#), [1359](#)
- \@picht [42:6](#), [42:40](#), [42:53](#), [42:62](#)
- \@picture [42:34](#), [42:35](#)
- \@picture@warn
 [42:234](#), [42:452](#), [42:456](#), [42:460](#)
- \@pkggetextension [1367](#)
- \@pkgextension [50:31](#), [50:190](#), [50:198](#),
 [50:256](#), [50:527](#), [50:719](#), [50:722](#),
 [50:761](#), [50:770](#), [50:868](#), [50:896](#),
 [50:923](#), [50:1045](#), [50:1072](#), [50:1193](#),
 [50:1223](#), [50:1658](#), [52:482](#), [52:492](#), [1126](#)
- \@plus [18:554](#),
 [06:26](#), [44:16](#), [44:231](#), [44:254](#),
 [49:127](#), [54:3007](#), [54:3008](#), [54:3009](#),
 [54:3012](#), [54:3013](#), [54:3017](#), [54:3018](#),
 [54:3019](#), [54:3023](#), [54:3024](#), [54:3025](#)
- \@pnumwidth [44:243](#), [44:266](#)
- \@popfilename
 [50:33](#), [50:1003](#), [50:1138](#), [1101](#)
- \@pr@videopackage
 [50:391](#), [50:405](#), [50:408](#),
 [50:429](#), [50:431](#), [50:442](#), [50:444](#), [50:455](#)
- \@preamble .. [41:242](#), [41:244](#), [41:252](#),
 [41:265](#), [41:267](#), [41:275](#), [41:314](#),
 [41:333](#), [41:335](#), [41:336](#), [41:340](#),
 [41:355](#), [41:373](#), [41:374](#), [41:411](#), [55:351](#)
- \@preamblecmds [20:67](#), [20:136](#),
 [20:191](#), [06:66](#), [50:1854](#), [50:1855](#), [1056](#)
- \@preamerr
 [14:260](#), [41:251](#), [41:274](#), [41:351](#), [41:432](#)
- \@process@pti@ns
 [50:588](#), [50:606](#), [50:623](#),
 [50:630](#), [50:631](#), [50:644](#), [50:648](#), [50:650](#)
- \@process@ptions [50:575](#), [50:577](#), [50:589](#)
- \@protect@... [1378](#)
- \@protected@ttestopt
 [06:107](#), [06:119](#), [06:720](#), [06:732](#)
- \@protecteddef [1373](#)
- \@providesfile
 [50:465](#), [57:788](#), [01:82](#), [01:83](#)
- \@optionlist
 .. [50:187](#), [50:259](#), [50:574](#), [50:1011](#),
 [50:1017](#), [50:1106](#), [50:1112](#), [50:1224](#)
- \@pushfilename
 [50:33](#), [50:948](#), [50:1093](#), [1101](#)
- \@put [42:463](#),
 [42:507](#), [42:538](#), [42:563](#), [42:631](#), [42:649](#)
- \@qend [14:236](#), [06:177](#), [06:868](#)
- \@qrelax [06:178](#), [06:868](#)
- \@raw@classoptionslist
 [50:11](#), [50:855](#), [51:98](#), [51:140](#)
- \@rc@ifdefinable [21:32](#),
 [06:171](#), [06:173](#), [06:295](#), [06:316](#), [06:342](#)
- \@reargdef [06:161](#), [1367](#)
- \@refundefined [20:55](#), [20:125](#), [20:180](#),
 [35:3](#), [37:60](#), [37:125](#), [37:164](#), [1393](#)
- \@reinserts [54:332](#), [54:335](#), [54:819](#)
- \@remove@eq@value [50:544](#), [50:611](#), [50:671](#)
- \@remove@tlig [21:1020](#), [21:1028](#)
- \@remove@tlig@ [21:1020](#), [21:1021](#)
- \@remove@tlig@@ [21:1021](#), [21:1024](#)
- \@remove@tlig@@@ [21:1034](#), [21:1053](#)
- \@removeelement . [13:32](#), [50:670](#), [50:679](#)
- \@removefromreset
 [22:148](#), [22:198](#), [22:199](#),
 [22:262](#), [22:274](#), [22:286](#), [22:289](#), [561](#)
- \@renewfontswitch [1367](#)
- \@reqcolroom
 [54:1441](#), [54:1442](#), [54:1445](#),
 [54:1447](#), [54:1448](#), [54:1453](#), [54:1454](#),
 [54:1458](#), [54:1460](#), [54:1488](#), [54:1489](#),
 [54:1592](#), [54:1595](#), [54:1597](#), [54:1598](#),
 [54:1603](#), [54:1607](#), [54:1609](#), [54:1637](#),
 [54:1638](#), [54:1739](#), [54:1740](#), [54:1744](#),
 [54:1747](#), [54:1748](#), [54:1753](#), [54:1760](#),
 [54:1762](#), [54:1794](#), [54:1795](#), [54:1906](#),
 [54:1908](#), [54:1910](#), [54:1911](#), [54:1914](#),
 [54:1916](#), [54:1988](#), [54:1990](#), [54:1992](#),
 [54:1995](#), [54:1997](#), [54:2070](#), [54:2073](#),
 [54:2076](#), [54:2081](#), [54:2083](#), [54:2597](#),
 [54:2728](#), [54:2729](#), [54:2734](#), [54:2737](#),
 [54:2779](#), [54:2784](#), [54:2787](#), [1436](#)
- \@reserveda [658](#)
- \@reset@ptions .. [50:967](#), [50:1004](#),
 [50:1032](#), [50:1097](#), [50:1139](#), [50:1149](#)
- \@resetactivechars
 [54:826](#), [54:851](#), [54:953](#), [54:1011](#)
- \@resetfps [54:1557](#),
 [54:1702](#), [54:1870](#), [54:2674](#), [1374](#)
- \@resetprotect [1388](#)
- \@restoremetafamily [29:228](#), [742](#)
- \@restorepar . [15:5](#), [18:410](#), [18:426](#),
 [18:444](#), [18:460](#), [39:127](#), [39:137](#), [1390](#)
- \@reversemarginfalse .. [45:388](#), [54:79](#)
- \@reversemargintrue [45:387](#)
- \@rightmark [49:16](#), [49:120](#)
- \@rightskip [37:458](#),
 [37:477](#), [37:495](#), [39:75](#), [40:455](#), [40:476](#)
- \@rjfieldfalse [41:34](#), [41:77](#)
- \@rjfieldtrue [41:125](#)
- \@rmfamilyhook
 [29:508](#), [29:530](#), [29:542](#), [29:569](#)
- \@roman [22:304](#), [22:310](#)
- \@rsbox [40:673](#), [40:680](#), [40:684](#)
- \@rtab [41:71](#), [41:86](#)
- \@rule [40:597](#), [40:602](#), [40:606](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@sanitize 46:7, 46:18, 46:24, 46:35, [06:892](#)
- \@savebox 40:166, 40:173, [40:179](#)
- \@savemarbox 45:326, 45:327, 45:330, [45:333](#)
- \@savepicbox ... 40:166, 40:173, [40:183](#)
- \@savsf 17:121, 17:123, 17:146, 17:148, 17:169, 17:171, [18:124](#), 18:130, 18:139, 18:162, 18:180, 18:194, 18:206, 18:220, 18:234
- \@savsk [18:124](#), 18:129, 18:140, 18:163, 18:181, 18:195, 18:207, 18:221, 18:235
- \@scolelt 54:1139, [54:1204](#)
- \@sdblcplelt 54:1156, 54:1176, [54:1205](#)
- \@secntformat ... 44:60, [44:111](#), [1361](#)
- \@secondoffive 35:24, 35:27, 35:41, 35:58
- \@secondoftwo 03:87, 03:185, 17:5, 20:528, 20:558, 20:604, 20:621, 21:149, 22:357, 22:362, 24:221, 24:241, 05:41, 05:45, 29:591, 33:800, 33:850, 33:866, 35:44, 35:80, [06:253](#), 06:300, 06:321, 06:549, 06:678, 06:729, 49:17, 50:194, 06:837, 50:228, 50:240, 06:844, 50:273, 50:291, 50:340, 50:370, 06:864, 06:911, 50:1845, 54:646, 54:651, 54:658, 57:802, 01:72
- \@secpenalty 18:18, [44:36](#), 44:50
- \@sect 44:54, [44:55](#)
- \@seqnrcr 38:518
- \@set@curr@file@aux [52:363](#), [1428](#)
- \@setckpt 20:458, [20:465](#), 37:24, 37:91, 37:153, [1391](#)
- \@setfloattypecounts ... 54:1425, 54:1576, 54:1720, 54:1891, 54:1973, 54:2053, 54:2148, 54:2269, [54:2621](#)
- \@setfontsize 29:748
- \@setfps 45:34
- \@setfpsbit ... 45:73, 45:75, 45:77, 45:85, 45:143, 45:146, 45:149, [54:2665](#)
- \@setmarks .. 54:2918, 54:2920, 54:2934
- \@setminipage 40:521, 45:21, 45:177, [45:185](#), 45:376, [1385](#)
- \@setminpage 1398
- \@setnobreak [45:179](#), [45:375](#), [1385](#)
- \@setpar 15:3, 39:78
- \@setprotect 1388
- \@setref 35:10
- \@setsize 29:748
- \@settab 41:71, [41:93](#)
- \@settodim 23:17
- \@settopoint 23:35
- \@setupverbvisiblespace .. 37:587, 37:599, [37:603](#), 37:664, 37:678, 37:692
- \@setupverbvisibletab 37:615, [37:636](#)
- \@sffamilyhook 29:508, 29:535, 29:543, 29:570
- \@sharp 41:248, 41:271, 41:312, 41:342, 41:357, 41:358, 41:378, 41:380, 41:382, 41:410
- \@shipoutsetup 54:841
- \@shortstack 42:143, 42:144
- \@show@... 103
- \@show@DeclareRobustCommand 06:646, 06:660, [06:705](#), 06:759, 06:762
- \@show@DeclareRobustCommand@env . 06:783, [06:785](#)
- \@show@environment ... 06:777, 06:778
- \@show@environment@begin 06:786, 06:792, 06:800, 06:805
- \@show@environment@end 06:790, 06:799, [06:804](#), 108
- \@show@environment@end@aux 06:808, 06:810
- \@show@macro 06:705, 06:795
- \@show@newcommand . 06:647, 06:660, 06:713, [06:740](#), 06:759, 06:765, 105
- \@show@newcommand@aux 06:740, 06:769, 06:787, 106
- \@show@newcommand@env [06:785](#), 06:784
- \@show@nonstop ... 06:795, [06:816](#), 108
- \@show@normalenv 06:781, [06:804](#)
- \@show@tokens 06:745, [06:750](#), 06:794, 06:811
- \@show@typeout 06:789, 06:794, [06:816](#), 108
- \@showcommandlisthook 06:643, [06:645](#), 06:653, 07:1484, 171
- \@showenvironmentlisthook 06:780, [06:782](#), 07:1486, [1428](#)
- \@skipping@modulefalse 03:154, 03:173, 03:200
- \@skipping@moduletrue 03:153, 03:177
- \@sline . 42:178, 42:190, [42:195](#), 42:280
- \@slowromancap 22:311, [22:312](#)
- \@spaces 14:218
- \@specialoutput 54:261
- \@specialpagefalse 54:75, 54:887, 54:962, 54:1020
- \@specialpagetrue 49:9
- \@specialstyle 49:9, 54:888, 54:962, 54:1020
- \@sptoken 06:877, [06:887](#)
- \@sqrt 38:413
- \@ssect 44:53, [44:112](#)
- \@stackcr 42:150, [42:153](#)
- \@star@or@long [06:72](#), 06:95, 06:163, 06:187, 06:193, 06:219, 06:228, 06:274

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=ltlooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@startcolumn</code> .	54:268, 54:275, 54:1126	40:281, 40:291, 40:504, 40:535,
<code>\@startdblcolumn</code>		40:690, 40:691, 40:692, 40:699,
.	54:1126, 54:2896, 54:2899,	40:700, 40:701, 40:702, 42:315,
	54:2938, 54:2940, 54:2977, 54:2983	42:316, 42:458, 42:459, 42:496,
<code>\@startfield</code>	41:28,	42:501, 42:506, 42:507, 42:525,
	41:46, 41:92, 41:104, 41:125, 41:133	42:530, 42:537, 42:538, 42:552,
<code>\@startline</code>		42:555, 42:562, 42:563, 42:624,
	41:20, 41:57, 41:62, 41:68, 41:83, 41:154	42:625, 42:630, 42:631, 42:640,
<code>\@startpagehook</code>	1364	42:641, 42:648, 42:649, 42:734,
<code>\@startpbox</code>	41:245, 41:268,	42:752, 42:790, 42:808, 44:138,
	41:313, 41:343, 41:409, 41:461, 41:463	44:139, 45:322, 45:380, 54:310,
<code>\@startsection</code>	44:39	54:382, 54:387, 54:388, 54:429,
<code>\@starttoc</code>	44:149, 1318	54:434, 54:435, 54:556, 54:794,
<code>\@stopfield</code>	41:32, 41:48, 41:86, 41:93,	54:913, 54:925, 54:926, 54:981,
	41:125, 41:127, 41:140, 41:142, 41:154	54:988, 54:989, 54:1040, 54:1047,
<code>\@stopline</code>	41:30, 41:56, 41:85	54:1048, 54:1072, 54:1076, 54:1088,
<code>\@stpelt</code>	22:37, 22:40	54:1094, 54:1101, 54:1102, 54:1103,
<code>\@string@makeletter</code>	06:898, 111	54:1104, 54:1108, 54:1116, 1360
<code>\@strip@args</code>	21:94	
<code>\@strip@tex@ext</code>	20:246,	<code>\@tempcnta</code>
	20:256, 20:258, 20:263, 20:293, 493	28:1026, 28:1027, 28:1028, 28:1029,
<code>\@strip@tex@ext@aux</code>	20:263, 20:294	28:1033, 41:319, 41:320, 41:321,
<code>\@svector</code>	42:255, 42:270, 42:280	41:322, 42:198, 42:199, 42:225,
<code>\@sverb</code>		42:226, 42:227, 42:240, 42:241,
	37:648, 37:727, 37:736, 37:739, 1361	42:242, 42:243, 42:250, 42:251,
<code>\@svsec</code>	44:57, 44:60, 44:66, 44:78, 1361	42:265, 42:266, 42:281, 42:282,
<code>\@svsechd</code>	44:76, 44:101, 44:121	42:287, 42:289, 42:290, 42:291,
<code>\@swaptwoargs</code>	20:638,	42:292, 42:293, 42:296, 42:298,
	20:640, 50:977, 52:153, 52:167, 52:187	42:299, 42:300, 42:301, 42:302,
<code>\@sxverbatim</code>	37:520, 37:588, 37:595	42:303, 42:304, 42:305, 42:306,
<code>\@tabacckludge</code>		42:307, 42:346, 42:347, 42:348,
	21:243, 21:245, 21:475,	42:349, 42:350, 42:371, 42:372,
	21:476, 33:157, 33:164, 33:166, 1415	42:373, 42:374, 42:375, 42:376,
<code>\@tabacol</code>	41:206, 41:213, 41:335	42:403, 42:404, 42:405, 42:406,
<code>\@tabarray</code>	41:174,	42:407, 42:425, 42:427, 42:429,
	41:181, 41:208, 41:215, 41:218, 1441	42:430, 42:432, 42:434, 42:449,
<code>\@tabclassiv</code>	41:208, 41:215, 41:407	42:450, 42:451, 42:453, 42:455,
<code>\@tabclassz</code>	41:207, 41:214, 41:359	42:457, 42:459, 42:572, 42:577,
<code>\@tabcr</code>	41:56, 41:73	42:585, 42:589, 42:626, 42:627,
<code>\@tabfbox</code>	41:16, 41:80, 41:82	42:628, 42:629, 42:642, 42:643,
<code>\@tablab</code>	41:72, 41:126	42:645, 42:646, 42:668, 42:669,
<code>\@tabminus</code>	41:72, 41:117	42:670, 42:671, 42:672, 42:673,
<code>\@tabplus</code>	41:72, 41:110	42:721, 42:741, 42:781, 42:797,
<code>\@tabpush</code>	41:11,	45:62, 45:68, 45:70, 45:79, 45:80,
	41:77, 41:85, 41:140, 41:143, 41:145	45:90, 45:91, 45:131, 45:137, 45:139,
<code>\@tabrj</code>	41:72, 41:124	45:152, 45:153, 45:159, 45:160,
<code>\@tabular</code>	41:198, 41:201, 41:202	50:1284, 50:1286, 50:1287, 50:1288,
<code>\@tabularcr</code>	41:208, 41:215, 41:285	50:1415, 50:1417, 50:1418, 50:1419,
<code>\@tempa</code>	1385	50:1529, 50:1531, 50:1532, 50:1533,
<code>\@tempboxa</code>	12:13, 21:87, 23:21,	54:12, 54:14, 54:16, 54:1285,
	23:22, 23:23, 23:28, 23:29, 39:236,	54:1286, 54:1287, 54:1288, 54:1308,
	39:242, 39:243, 39:245, 40:29, 40:30,	54:1309, 54:1310, 54:1311, 54:1333,
	40:31, 40:32, 40:37, 40:38, 40:39,	54:1336, 54:1369, 54:1372, 54:1484,
	40:40, 40:219, 40:253, 40:272,	54:1633, 54:1790, 54:2151, 54:2154,
		54:2272, 54:2275, 54:2390, 54:2392,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

54:2395, 54:2397, 54:2399, 54:2421,
 54:2655, 54:2656, 54:2660, 54:2666,
 54:2670, 57:227, 57:232, 57:233,
 57:234, 57:413, 57:415, 57:416,
 57:417, 57:424, 57:426, 57:427,
 57:428, 57:434, 57:436, 57:437,
 57:438, 57:445, 57:447, 57:448,
 57:449, 57:475, 57:477, 57:478,
 57:479, 57:501, 57:503, 57:504,
 57:505, 57:511, 57:513, 57:514,
 57:515, 57:539, 57:544, 57:545, 57:546
 \@tempcntb . . 12:7, 28:1027, 28:1031,
 28:1033, 42:290, 42:291, 42:292,
 42:294, 42:295, 42:296, 42:572,
 42:573, 42:577, 42:578, 42:585,
 42:586, 42:589, 42:590, 45:88, 45:89,
 45:90, 45:157, 45:158, 45:159, 54:13,
 54:16, 54:17, 54:2666, 54:2667,
 54:2668, 57:228, 57:232, 57:540, 57:544
 \@tempdima . . 12:10, 24:282, 24:287,
 38:186, 38:189, 38:195, 40:47,
 40:48, 40:62, 40:63, 40:252, 40:253,
 40:271, 40:272, 40:279, 40:280,
 40:281, 40:283, 40:372, 40:373,
 40:397, 40:398, 40:423, 40:424,
 40:502, 40:506, 40:609, 40:612,
 40:613, 40:688, 40:690, 40:696,
 40:699, 41:35, 41:36, 41:37, 41:88,
 41:89, 41:90, 41:91, 41:295, 41:296,
 42:221, 42:222, 42:224, 42:225,
 42:226, 42:227, 42:228, 42:229,
 42:448, 42:449, 42:450, 42:459,
 42:497, 42:498, 42:502, 42:503,
 42:526, 42:527, 42:531, 42:533,
 42:553, 42:554, 42:556, 42:558,
 42:627, 42:629, 42:644, 42:646,
 42:647, 42:667, 42:668, 42:669,
 44:236, 44:237, 44:259, 44:260,
 44:273, 45:196, 45:198, 45:218,
 45:220, 45:258, 45:259, 45:260,
 54:234, 54:235, 54:236, 54:526,
 54:528, 54:529, 54:534, 54:538,
 54:542, 54:547, 54:551, 54:777,
 54:779, 54:780, 54:783, 54:787,
 54:789, 54:1268, 54:1271, 54:1291,
 54:1301, 54:1313, 54:1323, 54:2214,
 54:2215, 54:2218, 54:2219, 54:2339,
 54:2340, 54:2344, 54:2345, 54:2400,
 54:2401, 54:2402, 54:2403, 54:2406,
 54:2409, 54:2412, 54:2414, 54:2828,
 54:2829, 54:2831, 54:2832, 1361
 \@tempdimb 12:10, 24:283, 24:288, 24:721,
 24:725, 26:180, 26:181, 26:454,
 26:477, 26:478, 26:487, 26:488,
 26:492, 26:510, 26:513, 26:516,
 26:518, 40:375, 40:376, 40:400,
 40:401, 40:426, 40:427, 40:610,
 40:613, 40:689, 40:691, 40:697,
 40:700, 42:222, 42:223, 42:368,
 42:370, 42:373, 42:376, 42:492,
 42:494, 42:495, 42:520, 42:523,
 42:524, 42:549, 42:550, 42:551,
 42:622, 42:623, 42:632, 42:638,
 42:639, 42:650, 42:658, 42:659,
 42:664, 54:1291, 54:1292, 54:1293,
 54:1294, 54:1301, 54:1313, 54:1314,
 54:1315, 54:1316, 54:1323, 1405
 \@tempdimc . . 12:10, 26:471, 26:472,
 26:474, 26:475, 26:477, 26:478,
 40:74, 40:75, 40:85, 40:86, 40:611,
 40:612, 40:613, 42:41, 42:42, 42:43,
 42:44, 42:45, 42:46, 42:71, 42:72,
 42:74, 42:75, 42:343, 42:344, 42:345
 \@tempdimx 1361
 \@tempskipa 12:14, 18:45, 18:48, 18:49, 18:322,
 18:329, 18:331, 18:334, 18:355,
 18:362, 18:364, 18:367, 26:182,
 26:183, 39:116, 39:117, 39:118,
 39:181, 39:183, 39:184, 39:185,
 39:253, 39:254, 39:255, 44:42, 44:44,
 44:45, 44:50, 44:62, 44:63, 44:88,
 44:89, 44:91, 44:103, 44:104, 44:113,
 44:114, 54:568, 54:569, 54:581,
 54:2449, 54:2450, 54:2452, 54:2460
 \@tempskipb 12:14,
 18:243, 18:245, 18:247, 18:250,
 18:252, 18:266, 18:281, 18:298,
 18:320, 18:322, 18:323, 18:327,
 18:329, 18:331, 18:332, 18:353,
 18:355, 18:356, 18:360, 18:362,
 18:364, 18:365, 18:387, 18:390, 476
 \@tempswa 1357
 \@tempswafalse . . . 20:322, 20:384,
 20:431, 21:1565, 24:95, 24:772,
 28:538, 28:613, 28:687, 28:768,
 28:1288, 28:1294, 37:27, 37:93,
 37:155, 37:535, 37:556, 47:17, 47:29,
 50:1257, 50:1273, 50:1392, 50:1408,
 52:286, 52:309, 52:330, 54:1339,
 54:1375, 54:2157, 54:2278, 02:148, 01:62
 \@tempswatruw . . . 20:320, 20:325,
 20:382, 20:387, 20:429, 20:434,
 21:1568, 21:1569, 24:98, 24:773,
 24:774, 24:777, 24:780, 28:541,
 28:616, 28:690, 28:771, 28:1251,
 36:174, 37:198, 37:540, 37:561,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- 47:17, 47:29, 50:1254, 50:1389,
52:286, 52:309, 52:330, 54:2159,
54:2182, 54:2280, 54:2305, 54:2739,
54:2756, 54:2789, 54:2806, 02:154, 01:63
\@temptokena 12:16, 37:210,
37:214, 37:223, 37:236, 37:237,
49:42, 49:46, 49:53, 49:56, 49:68,
49:69, 49:76, 49:77, 49:91, 49:92,
49:99, 49:100, 49:122, 49:123, 871
\@test... 1396
\@test@opt 1392
\@testdef .. 37:25, 37:92, 37:154, 37:196
\@testfalse 54:8, 54:10, 54:11
\@testfp
54:1233, 54:1253, 54:1289, 54:1312,
54:2658, 54:2842, 54:2859, 1384
\@testopt
18:7, 18:8, 18:9, 18:10, 06:33, 06:97,
06:117, 06:121, 06:189, 38:489, 1392
\@testpach 41:347, 41:425
\@testtrue
.... 54:9, 54:17, 54:361, 54:1236,
54:1255, 54:1295, 54:1317, 54:2662
\@testwrongwidth
54:350, 54:1234, 54:1290, 54:1464,
54:1613, 54:1923, 54:2004, 54:2208
\@text@composite
..... 21:94, 21:1098, 21:1103, 1386
\@text@composite@x 21:94
\@textbottom 49:127,
49:129, 54:514, 54:539, 54:553,
54:643, 54:764, 54:784, 54:791, 54:824
\@textfloatsheight 54:483,
54:1438, 54:1440, 54:1491, 54:1492,
54:1497, 54:1589, 54:1591, 54:1640,
54:1641, 54:1646, 54:1736, 54:1738,
54:1798, 54:1800, 54:1806, 54:2597
\@textmin 45:285,
45:286, 45:299, 45:300, 54:90,
54:1440, 54:1444, 54:1447, 54:1448,
54:1591, 54:1594, 54:1597, 54:1598,
54:1738, 54:1743, 54:1747, 54:1748,
54:1910, 54:1992, 54:2076, 54:2175,
54:2177, 54:2193, 54:2297, 54:2299,
54:2317, 54:2715, 54:2717, 54:2719
\@textsubscript 45:455, 45:463
\@textsuperscript
..... 45:401, 45:403, 45:404, 1389
\@texttop 49:127,
49:129, 54:510, 54:760, 54:772, 54:824
\@tf@r 13:25, 13:26, 1370
\@tfor 13:25, 20:605, 20:622,
20:774, 32:89, 40:78, 40:97, 41:345,
42:488, 42:516, 42:545, 45:63, 45:132
\@tforloop 13:27, 13:28, 13:30
\@thanks 44:11, 44:34
\@thefnmark 40:550, 40:567,
40:584, 45:400, 45:401, 45:515,
45:520, 45:535, 45:553, 45:571,
45:583, 45:588, 45:599, 45:604, 45:615
\@thefoot 54:102, 54:893,
54:897, 54:935, 54:965, 54:968,
54:995, 54:1024, 54:1027, 54:1054
\@thehead 54:101, 54:892,
54:896, 54:920, 54:965, 54:967,
54:985, 54:1024, 54:1026, 54:1044
\@themargin 54:50, 54:894,
54:898, 54:912, 54:966, 54:968,
54:980, 54:1025, 54:1027, 54:1039
\@themark 49:41, 49:42, 49:52, 49:53,
49:67, 49:68, 49:75, 49:76, 49:90,
49:91, 49:98, 49:99, 49:121, 49:123
\@thirdofthree 21:215, 06:257
\@thm .. 43:12, 43:18, 43:28, 43:37, 43:41
\@thmcounter 43:11, 43:17, 43:60
\@thmcountersep 43:10, 43:60
\@title 44:7, 44:31
\@tocrmarg 44:232, 44:255
\@toodeep . 14:253, 39:36, 39:263, 39:274
\@toplist 54:40, 54:389,
54:390, 54:436, 54:437, 54:634,
54:809, 54:1073, 54:1083, 54:1084,
54:1376, 54:1388, 54:2531, 54:2559
\@topnewpage 54:204, 1373
\@topnum ... 45:271, 54:83, 54:1373,
54:1374, 54:1388, 54:1392, 54:1955,
54:1960, 54:2036, 54:2041, 54:2128,
54:2135, 54:2522, 54:2550, 54:2591
\@toproom 45:273, 54:84,
54:1376, 54:1388, 54:2523, 54:2551
\@topsep ... 39:1, 39:71, 39:73, 39:202
\@topsepadd
.... 39:1, 39:59, 39:61, 39:71, 39:124
\@totalleftmargin
37:531, 37:553, 39:9, 39:53, 39:54,
40:454, 40:475, 41:35, 41:76, 41:81, 908
\@trivlist 39:48, 39:57, 39:92
\@tryfcolumn 54:1129, 54:1149,
54:1167, 54:1183, 54:2843, 54:2860
\@trylist 54:1192,
54:1195, 54:1228, 54:1248, 54:1270
\@ttfamilyhook
..... 29:508, 29:540, 29:544, 29:571
\@twoclasseserror 50:710, 50:1243
\@twocolumnfalse 54:77, 54:126
\@twocolumntrue 54:211
\@twoloadclasserror
..... 50:1002, 50:1137, 50:1238

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\@twosidefalse</code>	54:78	28:142, 01:226, 28:581, 28:633,
<code>\@typein</code> 06:32, 06:33, 06:40, 06:48, <i>1392</i>		28:884, 05:164, 05:165, 05:166,
<code>\@typeset@protect</code>		05:167, 05:168, 05:169, 28:991,
. . . . 21:44, 21:50, 21:230, 21:238,		05:186, 05:187, 05:188, 05:189,
29:749, 06:120, 37:282, 37:319,		29:36, 29:59, 29:72, 29:92, 29:105,
06:347, <i>06:354</i> , 06:356, 57:407, <i>1384</i>		29:106, 29:107, 29:108, 29:109,
<code>\@uclclist</code>		29:110, 29:111, 29:112, 29:113,
21:1534, 21:1535, 21:1592, <i>57:620</i> , <i>1402</i>		29:114, 29:115, 29:117, 29:118,
<code>\@undefind</code>		29:119, 29:147, 29:148, 29:149,
42:68, 42:82, 42:93, 42:102, 42:173,		29:150, 29:151, 29:152, 29:153,
42:185, 42:248, 42:263, 42:324, 42:382		29:154, 29:155, 05:209, 05:210,
<code>\@undefined</code>		29:289, 29:290, 29:291, 29:325,
. . . . 02:563, 02:578, 01:92, 01:93,		29:383, 29:384, 29:551, 05:238,
01:94, 03:53, 03:62, 10:213, 10:214,		29:569, 29:570, 29:571, 29:615,
10:215, 10:217, 10:218, 10:219,		29:616, 29:617, 29:618, 29:619,
10:221, 10:222, 10:224, 10:225,		29:620, 29:621, 29:661, 29:662,
10:226, 10:227, 10:228, 10:229,		29:663, 29:664, 29:665, 29:666,
10:230, 10:231, 10:232, 04:2, 04:15,		29:677, 29:858, 30:15, 30:55, 30:62,
04:16, 04:17, 04:29, 04:30, 01:115,		30:77, 06:34, 32:37, 32:38, 32:39,
04:74, 04:84, 01:123, 14:28, 04:173,		32:123, 33:123, 33:331, 33:581,
01:131, 04:189, 04:197, 04:205,		33:584, 33:585, 33:615, 33:616,
16:172, 16:173, 16:174, 16:175,		33:617, 33:620, 33:621, 33:622,
16:176, 04:236, 04:237, 04:238,		33:623, 33:624, 33:625, 33:630,
04:239, 04:240, 17:62, 04:241,		33:631, 33:632, 33:633, 33:638,
04:242, 04:243, 04:244, 04:245,		33:639, 33:640, 33:641, 33:644,
04:246, 04:247, 04:248, 04:249,		33:645, 33:646, 33:648, 33:649,
04:250, 04:251, 01:138, 04:252,		33:654, 33:655, 33:656, 33:657,
17:181, 04:253, 04:254, 04:260,		33:658, 33:659, 33:660, 33:661,
18:90, 18:519, 18:568, 18:574,		33:662, 33:663, 33:664, 33:665,
20:61, 20:62, 20:130, 20:131, 20:185,		33:666, 33:667, 33:668, 33:669,
20:186, 20:293, 20:294, 20:496,		33:671, 33:672, 33:674, 33:675,
20:497, 20:498, 20:503, 20:557,		33:676, 33:677, 33:678, 33:679,
20:570, 20:579, 20:594, 20:649,		33:680, 33:681, 33:682, 33:683,
21:213, 21:215, 21:353, 21:355,		33:685, 33:686, 33:687, 33:688,
21:357, 21:359, 21:361, 21:363,		33:689, 33:690, 33:691, 33:692,
21:365, 21:367, 21:369, 21:371,		33:693, 33:695, 33:696, 33:697,
21:390, 21:392, 21:394, 21:480,		33:698, 33:699, 33:700, 33:701,
21:723, 21:726, 21:1007, 21:1016,		33:702, 33:703, 33:704, 33:705,
21:1056, 21:1060, 22:188, 22:371,		33:706, 33:707, 33:708, 33:709,
24:531, 24:571, 24:633, 24:666,		33:710, 33:711, 33:712, 33:713,
24:730, 24:737, 01:189, 01:193,		33:714, 33:715, 33:716, 33:717,
25:2776, 25:2799, 25:2808, 25:2897,		33:718, 33:719, 33:720, 33:721,
25:2898, 25:2899, 25:2900, 25:2901,		33:726, 33:727, 33:729, 33:730,
25:2902, 25:2903, 25:2904, 25:2905,		33:731, 33:732, 33:733, 33:734,
25:2906, 25:2917, 25:2922, 25:2927,		33:735, 33:736, 33:738, 33:739,
25:2934, 25:2935, 25:2936, 25:2937,		33:741, 33:742, 33:744, 33:745,
25:2938, 25:2939, 25:2940, 25:3154,		33:746, 33:747, 33:749, 33:750,
25:3234, 25:3236, 25:3237, 25:3239,		33:751, 33:753, 33:755, 33:756,
25:3240, 25:3241, 25:3243, 05:4,		33:757, 33:758, 33:759, 33:760,
05:10, 05:16, 05:17, 05:18, 05:19,		33:761, 33:763, 33:764, 33:765,
05:20, 01:219, 26:569, 26:570, 27:4,		33:766, 33:767, 33:768, 33:769,
27:5, 27:6, 27:7, 27:8, 27:9, 27:10,		33:770, 33:771, 35:172, 35:185,
27:11, 27:12, 27:13, 27:14, 27:15,		35:225, 35:226, 06:276, 36:290,
27:16, 27:17, 27:18, 27:19, 27:20,		36:291, 36:293, 36:294, 36:295,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

36:297,	36:298,	36:299,	36:301,	57:493,	57:494,	57:495,	57:496,
36:302,	37:178,	37:230,	37:231,	57:497,	57:498,	57:499,	57:500,
37:232,	37:233,	37:308,	37:387,	57:517,	57:533,	57:534,	57:535,
37:407,	37:408,	37:409,	37:410,	57:578,	57:579,	57:740,	57:775,
37:517,	37:579,	37:597,	37:598,	57:776,	57:777,	57:778,	57:779,
37:599,	37:600,	37:633,	37:645,	02:112,	02:142,	02:228,	02:238,
37:694,	38:236,	38:246,	38:247,	02:240,	01:52,	01:53,	02:350,
38:283,	38:286,	38:373,	38:386,	02:351,	02:361,	02:362,	02:456,
38:529,	06:411,	06:412,	06:443,	02:523,	1181		
40:21,	06:471,	40:174,	06:475,	\@undefinedfonterror		1374	
06:476,	40:238,	40:319,	40:359,	\@unexpandable@protect			
40:360,	40:361,	40:362,	06:493,		20:215,	20:228,	06:273,
06:507,	40:603,	40:661,	40:662,	06:359,	06:365,	06:370,	41:341,
40:663,	40:664,	40:681,	01:310,	1383			
06:554,	06:555,	06:556,	01:311,	\@unknownoptionerror			
06:557,	06:558,	06:559,	06:560,		50:1153,	50:1213,	50:1226
42:18,	06:596,	06:597,	06:598,	\@unknownversion		1363	
06:599,	06:600,	06:601,	42:477,	\@unprocessedoptions		50:660,	
42:478,	06:632,	06:633,	06:634,	50:769,	50:1024,	50:1063,	50:1064,
06:652,	06:653,	44:203,	44:204,	50:1124,	50:1128,	50:1228,	51:88,
44:205,	44:206,	44:222,	44:270,	1397			
45:5,	45:460,	45:510,	45:624,	\@unused		14:15,	14:32,
47:30,	47:53,	06:760,	06:761,	06:10,	06:17,	50:1621,	02:177,
06:762,	06:763,	06:764,	06:765,	1421			
06:766,	06:767,	06:768,	06:769,	\@unusedoptionlist			
50:4,	50:25,	06:824,	50:181,	20:18,	20:20,	20:88,	20:90,
50:182,	50:183,	06:839,	50:244,	20:145,	20:147,	50:12,	50:553,
06:846,	50:451,	50:561,	50:611,	50:554,	50:564,	50:565,	50:672,
50:1024,	50:1062,	50:1064,	50:1070,	50:680,	51:128,	51:129,	51:135,
50:1127,	06:920,	50:1142,	06:921,	51:162,	1362		
50:1143,	06:922,	50:1158,	06:931,	\@upline		42:308,	42:309,
50:1297,	50:1375,	50:1378,	50:1428,	42:315			
50:1507,	50:1510,	50:1520,	50:1521,	\@upordown			
50:1522,	50:1523,	50:1524,	50:1525,	42:206,	42:207,	42:215,	42:236,
50:1526,	50:1597,	50:1600,	50:1616,	42:284			
50:1636,	50:1644,	52:18,	52:19,	\@upvector		42:279,	42:315
52:20,	52:21,	52:147,	52:189,	\@use@option	50:584,	50:600,	50:618,
52:197,	52:198,	52:199,	52:320,	50:634,	50:636,	50:653,	50:655,
52:341,	52:355,	52:359,	52:360,	50:665,	50:665		
52:492,	52:498,	52:499,	52:500,	\@use@text@encoding			
53:491,	53:492,	53:493,	53:494,		21:168,	33:16,	33:1102
53:495,	53:497,	53:498,	53:499,	\@vbsphack		18:242	
53:506,	53:507,	53:509,	53:511,	\@verb		37:727,	37:736,
53:512,	53:513,	53:516,	53:560,	\@verbatim	37:525,	37:571,	37:586,
02:61,	54:373,	54:374,	54:797,	\@verbatim	37:595		
54:798,	54:799,	54:800,	54:802,	\@verbvisiblespacebox	... 37:585,		
54:803,	54:804,	54:805,	54:806,	37:600,	37:612,	37:613,	37:626,
54:813,	54:814,	54:815,	02:77,	37:627			
02:82,	54:2501,	54:2502,	54:2503,	\@verreq		30:468,	30:469
54:2504,	54:2609,	02:89,	57:10,	\@viipt		24:815	
57:18,	57:39,	57:54,	57:73,	\@viipt		24:814	
57:82,	57:89,	57:98,	57:108,	\@vipt		24:813	
57:160,	57:161,	57:272,	57:285,	\@vline		42:177,	42:189,
57:298,	57:362,	57:374,	57:400,	\@vobeyspaces	 37:501,	37:571,	
57:401,	57:402,	57:431,	57:433,	37:588,	37:664,	37:678,	37:692,
57:472,	57:473,	57:492,		37:739			
				\@vobeytabs		37:501	
				\@vpt		24:812	
				\@vspace		18:399	
				\@vspace@calcify			
				... 18:80,	18:102,	18:264,	18:279,
				18:406,	18:411,	18:421,	18:429,
				37:437,	38:497,	41:62,	41:301,
				42:159			
				\@vspacer		18:399	
				\@vtryfc		54:1198,	54:1206

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\@vvector` [42:254](#), [42:269](#), [42:279](#)
- `\@warning` [14:215](#)
- `\@wckptelt` [20:459](#), [20:462](#)
- `\@whiledim` [13:7](#), [42:134](#), [42:214](#)
- `\@whilenoop` [13:3](#), [1391](#)
- `\@whilenum` [13:3](#),
[41:321](#), [42:115](#), [42:127](#), [42:347](#),
[42:349](#), [42:372](#), [42:375](#), [42:404](#),
[42:406](#), [42:427](#), [42:432](#), [42:741](#), [42:797](#)
- `\@whilesw` [13:10](#),
[54:269](#), [54:399](#), [54:408](#), [54:446](#),
[54:456](#), [54:2897](#), [54:2939](#), [54:2978](#)
- `\@whileswnoop` [13:10](#), [1391](#)
- `\@wholewidth`
... [40:204](#), [40:206](#), [40:207](#), [40:209](#),
[40:211](#), [40:212](#), [40:213](#), [40:214](#), [42:2](#),
[42:137](#), [42:140](#), [42:142](#), [42:310](#),
[42:313](#), [42:361](#), [42:370](#), [42:417](#),
[42:424](#), [42:575](#), [42:588](#), [42:600](#),
[42:610](#), [42:689](#), [42:690](#), [42:738](#), [42:794](#)
- `\@width` [18:550](#), [21:309](#), [21:314](#),
[26:193](#), [06:26](#), [30:629](#), [40:209](#),
[40:211](#), [40:287](#), [40:294](#), [40:613](#),
[40:714](#), [40:721](#), [41:240](#), [41:263](#),
[41:296](#), [41:424](#), [41:443](#), [42:238](#),
[42:310](#), [42:313](#), [42:336](#), [42:344](#),
[42:361](#), [42:370](#), [42:395](#), [42:403](#),
[42:417](#), [42:424](#), [42:575](#), [42:588](#),
[42:738](#), [42:794](#), [45:395](#), [54:2428](#),
[54:2887](#), [54:2930](#), [54:2961](#), [02:380](#), [1387](#)
- `\@wrglossary` [46:25](#), [46:30](#), [1383](#)
- `\@wrindex` [46:8](#), [46:13](#), [1383](#)
- `\@writeckpt`
..... [20:349](#), [20:405](#), [20:444](#), [20:456](#)
- `\@writefile` [20:32](#), [20:102](#),
[20:160](#), [37:217](#), [44:193](#), [44:201](#), [1022](#)
- `\@writesetup` [54:841](#)
- `\@wrong@font@char`
. [21:179](#), [24:634](#), [24:668](#), [24:681](#), [1394](#)
- `\@wtryfc` [54:1208](#), [54:1218](#)
- `\@x@protect` [06:123](#), [06:346](#)
- `\@x@sf` [45:593](#), [45:595](#)
- `\@xDeclareMathDelimiter`
..... [28:1056](#), [28:1112](#)
- `\@xaddvskip`
..... [18:242](#), [18:267](#), [18:282](#), [18:299](#)
- `\@xarg` [42:174](#),
[42:177](#), [42:186](#), [42:189](#), [42:196](#),
[42:200](#), [42:201](#), [42:237](#), [42:239](#),
[42:249](#), [42:250](#), [42:254](#), [42:264](#),
[42:265](#), [42:269](#), [42:277](#), [42:285](#), [42:674](#)
- `\@xargarraycr` .. [41:282](#), [41:291](#), [41:295](#)
- `\@xargdef` [06:98](#)
- `\@xarraycr` [41:279](#), [41:280](#)
- `\@xbitor` [54:11](#), [54:13](#)
- `\@xcentercr` [37:420](#), [37:427](#), [37:431](#)
- `\@xdblarg` [06:890](#)
- `\@xdblfloat` [45:264](#), [1382](#)
- `\@xdim` [42:95](#), [42:104](#), [42:116](#), [42:118](#),
[42:128](#), [42:130](#), [42:678](#), [42:742](#),
[42:743](#), [42:744](#), [42:745](#), [42:751](#),
[42:798](#), [42:799](#), [42:800](#), [42:801](#), [42:807](#)
- `\@xeqncr` [38:481](#)
- `\@xexnoop` [41:315](#), [41:325](#)
- `\@xexpast` [41:316](#), [41:317](#)
- `\@xfloat` [45:28](#), [45:29](#), [45:34](#), [45:266](#), [1383](#)
- `\@xfootnote` [45:514](#), [45:517](#), [1360](#)
- `\@xfootnotemark` . [45:581](#), [45:585](#), [1360](#)
- `\@xfootnotenext` [45:598](#), [45:601](#)
- `\@xfootnotetext` [1360](#)
- `\@xhline` [41:437](#), [41:438](#)
- `\@xifnch` [06:878](#), [06:888](#)
- `\@xiipt` . [24:819](#), [30:175](#), [30:177](#), [30:178](#)
- `\@xipt` [24:818](#), [30:174](#)
- `\@xivpt` [24:820](#), [30:176](#), [30:178](#)
- `\@xmpar` [45:324](#), [45:325](#)
- `\@xnewline`
..... [18:61](#), [18:62](#), [18:73](#), [18:74](#), [18:94](#)
- `\@xnnext` [54:6](#), [54:7](#)
- `\@xnthm` [43:5](#), [43:6](#)
- `\@xobeysp` [18:489](#), [18:514](#),
[37:506](#), [37:516](#), [37:520](#), [37:609](#),
[37:613](#), [37:623](#), [37:627](#), [37:640](#), [480](#)
- `\@xobeytab` [18:510](#), [37:508](#), [37:640](#)
- `\@xprocessOptions` [50:575](#),
[50:592](#), [50:594](#), [50:610](#), [50:612](#), [50:626](#)
- `\@xppt` ... [24:817](#), [30:173](#), [30:176](#), [30:177](#)
- `\@xssect` [44:86](#), [44:87](#), [44:123](#)
- `\@xtabcr` [41:56](#), [41:57](#)
- `\@xtabularcr` [41:286](#), [41:287](#)
- `\@xthm` [43:46](#), [43:52](#), [43:55](#)
- `\@xtryfc` [54:1195](#), [54:1223](#)
- `\@xtypein` [06:33](#), [06:35](#), [06:42](#)
- `\@xverbatim` [37:520](#), [37:571](#)
- `\@xvipt` [24:821](#), [30:177](#), [30:179](#)
- `\@xxxDeclareMathDelimiter`
..... [28:1041](#), [28:1045](#)
- `\@xxpt` [24:822](#), [30:178](#), [30:179](#)
- `\@xxvpt` [24:823](#), [30:179](#)
- `\@xxxii` [12:2](#), [21:446](#), [21:448](#),
[45:89](#), [45:158](#), [54:1230](#), [54:1231](#),
[54:1250](#), [54:1251](#), [54:1286](#), [54:1287](#),
[54:1309](#), [54:1310](#), [54:2624](#), [1394](#)
- `\@xympar` [45:328](#), [45:332](#), [45:378](#)
- `\@yarg` [42:174](#), [42:178](#),
[42:186](#), [42:190](#), [42:196](#), [42:197](#),
[42:206](#), [42:249](#), [42:255](#), [42:264](#),
[42:270](#), [42:279](#), [42:281](#), [42:308](#), [42:674](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@yargarraycr</code> ..	41:283 , 41:293 , 41:297
<code>\@yargd@f</code>	06:125
<code>\@yargdef</code>	.. 06:102 , 06:112 , 06:125 , 06:162 , 1370
<code>\@ydim</code>	42:96 , 42:105 , 42:116 , 42:119 , 42:128 , 42:130 , 42:678 , 42:746 , 42:747 , 42:748 , 42:749 , 42:750 , 42:802 , 42:803 , 42:804 , 42:805 , 42:806
<code>\@yeqncr</code>	38:481
<code>\@ympar</code>	45:324 , 45:329
<code>\@ynthm</code>	43:5 , 43:14
<code>\@ythm</code>	43:46 , 43:52 , 43:55
<code>\@ytryfc</code> ...	54:1241 , 54:1260 , 54:1264
<code>\@yyarg</code>	42:196 , 42:197 , 42:198 , 42:201 , 42:285 , 42:674
<code>\@ztryfc</code>	54:1269 , 54:1280
<code>\@@end</code>	243
<code>\accent@spacefactor</code>	.. 21:88 , 21:91 , 21:92 , 1402
<code>\active@math@prime</code>	.. 38:253 , 38:254 , 54:839 , 1402
<code>\add@accent</code>	21:83 , 21:85 , 1395
<code>\add@percent@to@temptokena</code>	.. 37:204 , 37:220 , 37:222 , 37:231 , 871
<code>\add@unicode@accent</code> ..	21:1067 , 21:1081
<code>\addto@hook</code>	.. 24:153 , 24:155 , 24:811 , 28:510 , 28:646 , 28:650 , 28:667 , 28:791 , 28:797 , 28:805 , 28:821 , 28:824 , 28:827 , 28:1260 , 28:1267 , 28:1270 , 1369
<code>\AddToHook</code>	327
<code>\AddToHookNext</code>	327
<code>\AddToHookNextWithArguments</code> ...	327
<code>\AddToHookWithArguments</code>	327
<code>\alias@ctr@F00</code>	558
<code>\alloc@</code>	04:22 , 04:26 , 04:38 , 24:15 , 02:141 , 1417
<code>\alpha@elt</code>	28:46 , 28:514 , 28:741 , 28:843 , 28:1259 , 28:1260
<code>\alpha@list</code> ..	28:42 , 28:44 , 28:523 , 28:729 , 28:741 , 28:786 , 28:841 , 28:842 , 28:1255 , 28:1261 , 28:1262
<code>\apptocmd</code>	328
<code>\atveryend@DEPRECATED</code> ..	52:580 , 52:582
<code>\best@size</code> ..	26:455 , 26:479 , 26:485 , 26:491
<code>\bf@def@ult</code>	29:476
<code>\bfdef@ult</code>	29:222 , 29:280 , 29:350 , 29:351 , 29:352 , 29:393 , 29:394 , 29:395 , 29:480 , 29:521 , 755
<code>\bfdefault@previous</code> ..	29:346 , 29:349 , 29:389 , 29:392 , 30:117 , 30:127 , 746
<code>\bfseries@..</code>	739
<code>\bfseries@previous</code>	1418
<code>\bfseries@rm</code>	29:106 , 29:129 , 29:147 , 29:300 , 29:303 , 29:337 , 29:350 , 29:393 , 29:398 , 29:434 , 754
<code>\bfseries@rm@kernel</code>	.. 29:109 , 29:132 , 29:150 , 29:300 , 744
<code>\bfseries@sf</code>	29:107 , 29:129 , 29:148 , 29:304 , 29:307 , 29:338 , 29:351 , 29:394 , 29:399 , 29:435 , 744
<code>\bfseries@sf@kernel</code>	.. 29:110 , 29:133 , 29:151 , 29:304
<code>\bfseries@tt</code>	29:108 , 29:129 , 29:149 , 29:308 , 29:311 , 29:339 , 29:352 , 29:395 , 29:400 , 29:436 , 744
<code>\bfseries@tt@kernel</code>	.. 29:111 , 29:134 , 29:152 , 29:308
<code>\bm@b</code>	40:37
<code>\bm@c</code> ..	40:37 , 40:50 , 40:256 , 40:384 , 40:410
<code>\bm@l</code>	40:37
<code>\bm@r</code>	40:37
<code>\bm@s</code>	40:37
<code>\bm@t</code>	40:37
<code>\botmark</code>	1065
<code>\bx@</code>	1290
<code>\bx@A</code>	54:21
<code>\bx@AA</code>	54:29
<code>\bx@B</code>	54:21
<code>\bx@BB</code>	54:29
<code>\bx@C</code>	54:21
<code>\bx@CC</code>	54:29
<code>\bx@D</code>	54:21
<code>\bx@DD</code>	54:29
<code>\bx@E</code>	54:21
<code>\bx@EE</code>	54:29
<code>\bx@F</code>	54:22
<code>\bx@FF</code>	54:30
<code>\bx@G</code>	54:22
<code>\bx@GG</code>	54:30
<code>\bx@H</code>	54:22
<code>\bx@HH</code>	54:30
<code>\bx@I</code>	54:22
<code>\bx@II</code>	54:30
<code>\bx@J</code>	54:22
<code>\bx@JJ</code>	54:30
<code>\bx@K</code>	54:23
<code>\bx@KK</code>	54:31
<code>\bx@L</code>	54:23
<code>\bx@LL</code>	54:31
<code>\bx@M</code>	54:23
<code>\bx@MM</code>	54:31
<code>\bx@N</code>	54:23
<code>\bx@NN</code>	54:31
<code>\bx@O</code>	54:24
<code>\bx@OO</code>	54:32
<code>\bx@P</code>	54:24

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\bx@PP</code>	54:32	<code>\c@list</code>	24:99, 24:115, 24:129, 24:157, 24:158, 28:390, 28:440, 28:542, 28:617, 28:691, 28:772, 57:522
<code>\bx@Q</code>	54:24	<code>\cf@encoding</code>	21:52, 21:59, 21:62, 21:69, 21:172, 24:319, 24:329, 24:339, 24:389
<code>\bx@QQ</code>	54:32	<code>\ch@ck</code>	50:1302, 50:1321, 50:1433, 50:1454, 50:1546, 02:129, 02:130, 02:131, 02:132, 02:133, 02:165, 02:167, 02:172
<code>\bx@R</code>	54:24	<code>\char@if@alph</code>	06:898
<code>\bx@RR</code>	54:32	<code>\chardef@text@cmd</code>	21:3
<code>\bx@S</code>	54:27	<code>\check@command</code>	06:228, 06:230
<code>\bx@SS</code>	54:33	<code>\check@icl</code>	32:9, 32:45, 32:50, 32:56, 32:64, 32:71, 32:73
<code>\bx@T</code>	54:27	<code>\check@icr</code>	32:9, 32:45, 32:51, 32:57, 32:65, 32:74, 32:79, 1399
<code>\bx@TT</code>	54:33	<code>\check@mathfonts</code>	19:5, 21:322, 21:348, 21:380, 21:1282, 24:430, 24:435, 24:442, 24:444, 26:251, 28:336, 28:475, 33:636, 40:335, 40:630, 45:412, 45:471, 1443
<code>\bx@U</code>	54:27	<code>\check@nocorr</code>	1380
<code>\bx@UU</code>	54:33	<code>\check@nocorr@</code>	32:47, 1399
<code>\bx@V</code>	54:27	<code>\check@range</code>	26:396, 26:397
<code>\bx@VV</code>	54:33	<code>\check@single</code>	26:395, 26:417
<code>\bx@W</code>	54:28	<code>\cl@ckpt</code>	20:459, 22:52
<code>\bx@WW</code>	54:34	<code>\cl@counter</code>	561
<code>\bx@X</code>	54:28	<code>\cl@page</code>	34:4
<code>\bx@XX</code>	54:34	<code>\col@number</code>	54:73, 54:127, 54:213, 54:225, 1373
<code>\bx@Y</code>	54:28	<code>\color@begingroup</code>	24:740, 24:802, 38:104, 38:134, 40:29, 40:110, 40:220, 40:505, 40:551, 40:568, 40:585, 41:47, 41:51, 45:537, 45:555, 45:573, 54:619, 54:745, 1370
<code>\bx@YY</code>	54:34	<code>\color@endbox</code>	40:110, 45:253, 45:348, 45:366, 54:229, 54:921, 54:936, 54:986, 54:996, 54:1045, 54:1055
<code>\bx@Z</code>	54:28	<code>\color@endgroup</code>	24:746, 24:808, 38:104, 38:134, 40:29, 40:119, 40:134, 40:178, 40:199, 40:222, 40:533, 40:555, 40:572, 40:588, 41:49, 45:541, 45:559, 45:576, 54:624, 54:749, 1370
<code>\bx@ZZ</code>	54:34	<code>\color@hbox</code>	40:110, 54:918, 54:933, 54:983, 54:993, 54:1042, 54:1052, 1384
<code>\c@</code>	22:188	<code>\color@setgroup</code>	40:110, 40:178, 40:197, 1375
<code>\c@*</code>	22:30, 1435	<code>\color@vbox</code>	40:110, 45:96, 45:165, 45:339, 45:357, 45:381, 54:220, 1385
<code>\c@bottomnumber</code>	45:269, 45:274, 54:2993	<code>\conditionally@traceoff</code>	14:291, 07:289, 07:301, 437
<code>\c@dbltopnumber</code>	45:268, 45:283, 45:297, 54:3000	<code>\conditionally@traceon</code>	14:291, 07:325
<code>\c@enumi</code>	39:258		
<code>\c@enumii</code>	39:258		
<code>\c@enumiv</code>	39:258, 1360		
<code>\c@equation</code>	38:393, 38:445, 38:474, 38:636		
<code>\c@errorcontextlines</code>	14:212, 1364		
<code>\c@footnote</code>	44:12, 45:397, 45:587		
<code>\c@localmathalphabets</code>	28:138, 28:151, 28:283, 28:331, 707		
<code>\c@mpfootnote</code>	40:517, 45:399		
<code>\c@ncel</code>	30:472, 30:473		
<code>\c@page</code>	34:3, 34:6, 34:7, 37:28, 48:439, 48:499, 53:432, 53:530, 54:117, 54:2395		
<code>\c@secnumdepth</code>	44:56, 44:71, 44:81, 44:140		
<code>\c@tocdepth</code>	44:140, 44:230, 44:253		
<code>\c@topnumber</code>	45:267, 45:271, 54:2989		
<code>\c@totalnumber</code>	45:270, 45:276, 54:2996		
<code>\c@totalpages</code>	53:395, 53:494		
<code>\calculate@math@sizes</code>	24:717, 26:220		
<code>\catcodetable@atletter</code>	04:93, 04:245		
<code>\catcodetable@initex</code>	04:93, 04:242		
<code>\catcodetable@latex</code>	04:93, 04:244		
<code>\catcodetable@string</code>	04:93, 04:243		
<code>\c@p@elt</code>	24:97, 24:117, 24:128, 24:129, 24:150, 24:153, 24:155, 28:372, 28:422, 28:540, 28:615, 28:689, 28:770, 57:521, 57:522		

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\contionally@tracoon</code>	437	<code>\declare@environmentcopy</code>	
<code>\copy@kernel@robust@command</code>		 06:609, 06:613, 06:618, 06:621	
.....	06:680, 06:767	<code>\declare@file@substitution</code>	
<code>\count@</code>	03:14,	 52:247, 52:567, 53:591, 1164	
03:15, 03:16, 03:17, 03:18, 03:20,		<code>\declare@robustcommand</code>	06:274
01:163, 01:164, 01:165, 01:170,		<code>\declare@robustcommand@auxi</code> .	06:274
24:776, 24:782, 24:784, 26:23,		<code>\declare@robustcommand@auxii</code>	06:274
26:303, 26:305, 26:327, 26:328,		<code>\declare@robustcommand@auxiii</code>	06:274
28:507, 28:509, 28:513, 28:872,		<code>\DeclareEncodingSubset@aux</code>	
28:873, 28:874, 28:920, 28:921,		 24:187, 24:189	
28:922, 28:981, 28:982, 28:983,		<code>\DeclareFontEncoding@</code>	
28:1029, 28:1030, 28:1031, 28:1069,		 24:123, 24:125,	
28:1070, 28:1071, 28:1077, 28:1078,		24:140, 57:432, 57:452, 57:518, 1411	
28:1079, 28:1123, 28:1124, 28:1125,		<code>\DeclareFontEncoding@saved</code>	
28:1131, 28:1132, 28:1133, 28:1192,		 57:432, 57:452, 57:534	
28:1193, 28:1194, 28:1200, 28:1201,		<code>\DeclareFontShape@</code>	24:22, 24:23
28:1202, 32:116, 32:119, 06:210,		<code>\DeclareRobustCommand</code>	329
06:214, 06:379, 06:404, 42:740,		<code>\DeclareSymbolFont@m@dropped</code> .	
42:741, 42:742, 42:745, 42:746,		 28:532, 28:537, 28:580, 28:581	
42:749, 42:753, 42:796, 42:797,		<code>\DeclareSymbolFontAlphabet@</code>	
42:798, 42:801, 42:802, 42:805,		 28:1247, 28:1250	
42:809, 50:1345, 50:1347, 50:1348,		<code>\DeclareUnicodeAccent@</code>	
50:1349, 50:1478, 50:1480, 50:1481,		 21:1074, 21:1076, 21:1080	
50:1482, 50:1567, 50:1569, 50:1570,		<code>\def</code>	329
50:1571, 02:41, 57:239, 57:240,		<code>\default@ds</code>	50:540, 50:571,
57:247, 57:249, 57:583, 57:584,		50:635, 50:654, 50:1151, 50:1153, 1362	
57:591, 57:593, 02:114, 02:115,		<code>\default@family</code>	
02:120, 02:122, 02:128, 02:129,		... 24:130, 24:162, 24:584, 24:598,	
02:130, 02:131, 02:132, 02:133,		24:601, 24:626, 24:661, 57:523, 1357	
02:134, 01:50, 02:380, 02:381, 1325		<code>\default@M</code>	24:137,
<code>\counterwithin@s</code> 22:236, 22:237, 22:252		24:177, 24:180, 24:184, 57:530, 1362	
<code>\counterwithin@x</code> 22:236, 22:239, 22:253		<code>\default@mextra</code>	27:10, 27:89
<code>\counterwithout@s</code> 22:284, 22:285, 22:299		<code>\default@series</code>	
<code>\counterwithout@x</code> 22:284, 22:287, 22:300		 24:130, 24:163, 24:585,	
<code>\curr@fontshape</code>		24:599, 24:602, 24:623, 24:658, 57:523	
... 21:198, 24:89, 24:460, 24:468,		<code>\default@shape</code>	
24:472, 24:474, 24:616, 24:622,		 24:131, 24:164, 24:586,	
24:625, 24:634, 24:641, 24:643,		24:600, 24:603, 24:621, 24:656, 57:524	
24:651, 24:657, 24:660, 24:668,		<code>\default@T</code>	
24:675, 24:677, 25:2842, 26:93,		. 24:171, 24:174, 24:184, 24:335, 1362	
26:101, 26:143, 26:170, 26:494,		<code>\define@mathalphabet</code>	
26:514, 26:546, 26:562, 26:577,		 27:18, 27:131, 1357	
28:394, 28:399, 28:444, 28:449,		<code>\define@mathgroup</code> .	27:19, 27:135, 1357
29:639, 29:648, 29:686, 29:693, 1358		<code>\define@newfont</code>	24:452, 24:461
<code>\curr@math@size</code>		<code>\delayed@f@adjustment</code> ...	24:353,
24:449, 26:257, 26:263, 26:268, 26:285		25:2783, 25:2784, 25:2785, 25:2787,	
<code>\declare@command@copy</code>	1417	25:2788, 25:2799, 25:3136, 25:3137,	
<code>\declare@commandcopy</code>	06:568,	25:3139, 25:3140, 26:123, 26:127,	
06:572, 06:577, 06:580, 06:599, 101		26:135, 26:139, 29:774, 29:806, 761	
<code>\declare@commandcopy@do</code>		<code>\delayed@merge@font@series</code>	
..... 06:580, 06:626, 06:627		 25:2784, 25:2848,	
<code>\declare@commandcopy@let</code> .	06:587,	25:2863, 25:2902, 26:134, 26:137	
06:591, 06:601, 06:702, 06:703, 101		<code>\delayed@merge@font@shape</code>	25:3137,
		25:3186, 25:3241, 26:133, 26:136	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`\development@branch@name`
 03:11, 03:39,
 03:53, 03:54, 03:55, 03:62, 03:63, 03:64
`\dimen@` 14:28, 14:29,
 18:418, 18:423, 18:452, 18:457,
 21:450, 21:451, 21:453, 21:454,
 21:820, 21:821, 24:277, 24:279,
 24:285, 24:298, 24:301, 24:305,
 24:720, 24:721, 24:722, 24:726,
 26:468, 26:469, 26:470, 26:471,
 26:475, 33:66, 33:68, 33:892, 33:894,
 38:72, 38:73, 38:199, 38:200, 38:201,
 38:202, 40:698, 40:701, 41:200,
 41:201, 02:41, 54:761, 54:763,
 54:773, 54:775, 02:377, 02:378,
 02:414, 02:415, 02:417, 02:419, 1396
`\dimen@i` 02:41
`\dimen@ii` 24:281, 24:286, 02:41
`\disable@package@load`
 52:477, 53:549, 1164
`\display` 38:204, 38:208, 38:209
`\do@add@percent@to@temptokena` ...
 37:208, 37:214, 37:232
`\do@emfont@update` 29:640, 29:644, 29:664
`\do@noligs` 37:741, 37:746
`\do@strongfont@update`
 29:687, 29:690, 29:696
`\do@subst@correction`
 24:85, 26:499, 26:554
`\document@default@language`
 20:51, 20:52,
 20:121, 20:122, 54:850, 54:952, 57:288
`\document@select@group`
 28:140, 28:146, 28:407, 28:457, 707
`\dollar@begin` 38:289,
 38:318, 38:398, 38:433, 38:625, 1437
`\dollar@end` 38:289,
 38:327, 38:400, 38:446, 38:636, 1437
`\dont@add@percent@to@temptokena` .
 37:207, 37:209, 37:233
`\do@store@version` ... 28:115, 28:120
`\ds@` 50:573, 50:1155, 1362
`\dt@pfalse` 38:205
`\dt@ptrue` 38:204
`\e@alloc` 04:15, 04:49,
 04:79, 04:89, 04:178, 04:182, 04:186,
 04:194, 04:202, 04:210, 02:47, 02:48,
 02:49, 02:51, 02:52, 02:59, 02:60,
 02:62, 02:64, 02:75, 02:78, 02:80,
 02:94, 57:12, 57:47, 02:141, 1417
`\e@alloc@attribute@count`
 04:66, 04:74, 04:75, 04:76, 04:80, 04:236
`\e@alloc@bytecode@count` ... 04:70,
 04:197, 04:198, 04:199, 04:203, 04:252
`\e@alloc@ccodetable@count`
 04:67, 04:84, 04:85, 04:86, 04:90, 04:240
`\e@alloc@chardef` 04:48,
 04:178, 04:194, 04:202, 04:210,
 02:56, 02:82, 57:12, 02:134, 02:135
`\e@alloc@intercharclass@top` .. 57:35
`\e@alloc@luachunk@count` ... 04:71,
 04:205, 04:206, 04:207, 04:211, 04:254
`\e@alloc@luafunction@count`
 ... 04:68, 04:173, 04:174, 04:175,
 04:179, 04:183, 04:187, 04:246, 04:248
`\e@alloc@top` 04:47,
 04:80, 04:179, 04:183, 04:187,
 04:195, 04:203, 04:211, 02:51,
 02:59, 02:82, 57:12, 02:111, 02:145
`\e@alloc@whatsit@count` ... 04:69,
 04:189, 04:190, 04:191, 04:195, 04:250
`\e@ch@ck` ... 04:51, 04:55, 02:96, 02:100
`\e@insert@top` .. 02:143, 02:145, 02:162
`\e@mathgroup@top`
 28:57, 28:150, 28:151,
 28:188, 28:218, 02:75, 02:89, 1407
`\em@currfont` 29:639, 29:650, 757
`\em@force`
 29:648, 29:653, 29:656, 29:666, 757
`\emfontdeclare@clist`
 29:629, 29:631, 29:635,
 29:640, 29:645, 29:651, 29:662, 756
`\empty@sfcnt` 26:507,
 26:508, 26:509, 26:523, 26:528, 26:580
`\ENC@cmd` 1377
`\enc@update` 24:320, 24:322,
 24:338, 24:341, 26:148, 26:174, 1377
`\end@dblfloat` 45:205
`\end@float`
 45:189, 45:227, 45:243, 45:383, 1040
`\endlinechar` 348
`\enlargethispage` 1071
`\err@rel@i` . 27:12, 27:99, 27:132, 27:136
`\error@fontshape`
 24:579, 24:594, 24:619,
 24:654, 26:108, 26:544, 28:393, 28:443
`\escapechar` 340
`\et@xmaxfam` . 04:22, 04:26, 04:30, 04:38
`\et@xmaxregs` 04:29, 04:31,
 04:32, 04:33, 04:34, 04:35, 04:36, 04:37
`\every@math@size`
 24:79, 26:236, 26:248, 1384
`\every@size` 1387
`\execute@size@function`
 26:379, 26:407, 26:421, 26:438
`\expand@font@defaults`
 29:38, 29:171,
 29:241, 29:335, 29:358, 29:388,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 29:409, 29:433, 29:444, 29:475,
 29:476, 29:515, 29:517, 29:549,
 29:551, 29:580, 33:26, 33:47, 244
 expand@font@defaults 29:499
 \expandafter 1179
 \external@font 26:85, 26:88, 26:99,
 26:103, 26:105, 26:408, 26:422,
 26:484, 26:518, 26:586, 26:588, 26:590
 \extra@def 27:9, 27:84, 1357
 \extract@alph@from@version
 24:694, 24:700,
 28:163, 28:194, 28:224, 28:256, 1360
 \extract@default@composite
 21:1090, 21:1097
 \extract@default@composite@a ...
 21:1099, 21:1103
 \extract@default@composite@b ...
 21:1101, 21:1105
 \extract@font 24:475, 26:82
 \extract@fontinfo 26:375, 26:382
 \extract@rangefontinfo
 26:392, 26:399, 26:418, 26:451
 \extract@sizefn 26:367, 26:389
 \f@... 1377
 \f@baselineskip
 .. 21:894, 21:1248, 24:380, 24:387,
 24:605, 26:122, 26:168, 26:183,
 26:187, 26:202, 26:216, 26:227, 26:241
 \f@depth 45:291, 54:350
 \f@encoding . 04:259, 04:274, 04:282,
 21:196, 24:314, 24:333, 24:336,
 24:337, 24:339, 24:389, 24:455,
 24:460, 24:479, 24:481, 24:483,
 24:488, 24:490, 24:521, 24:583,
 24:615, 24:650, 25:2827, 25:2831,
 25:3172, 25:3176, 26:92, 26:129,
 26:308, 26:534, 28:378, 28:428, 1377
 \f@family
 ... 24:209, 24:210, 24:213, 24:216,
 24:233, 24:236, 24:342, 24:350,
 24:362, 24:372, 24:383, 24:456,
 24:460, 24:479, 24:481, 24:483,
 24:488, 24:490, 24:522, 24:601,
 24:626, 24:661, 25:2827, 25:2831,
 25:3172, 25:3176, 26:92, 26:129,
 28:378, 28:409, 28:428, 28:459,
 29:173, 29:200, 29:201, 29:244,
 29:264, 29:337, 29:338, 29:339,
 29:360, 29:361, 29:362, 29:398,
 29:399, 29:400, 29:418, 29:419,
 29:420, 29:434, 29:435, 29:436,
 29:445, 29:446, 29:447, 29:583,
 29:598, 29:770, 29:780, 29:783,
 29:786, 29:803, 29:819, 29:835,
 29:867, 29:873, 29:894, 33:23, 33:27,
 33:29, 33:31, 33:44, 33:48, 33:50,
 33:52, 33:57, 33:64, 33:858, 33:861,
 33:875, 33:884, 33:890, 33:1105, 1357
 \f@linespread 24:383, 26:121,
 26:167, 26:184, 26:185, 26:188,
 26:196, 26:199, 26:210, 26:213, 1377
 \f@series 19:14,
 24:342, 24:373, 24:384, 24:457,
 24:460, 24:602, 24:623, 24:658,
 25:2788, 25:2798, 25:2807, 25:2817,
 25:2851, 25:2870, 25:2871, 25:3172,
 25:3176, 26:126, 26:129, 26:132,
 28:410, 28:460, 29:167, 29:182,
 29:184, 29:188, 29:189, 29:190,
 29:211, 29:217, 29:220, 29:222,
 29:252, 29:255, 29:272, 29:277,
 29:279, 29:280, 29:315, 29:601,
 29:605, 29:771, 29:804, 29:820,
 29:836, 29:896, 33:7, 33:577, 663
 \f@series@saved 26:126, 26:132
 \f@shape 24:342, 24:352,
 24:364, 24:374, 24:385, 24:458,
 24:460, 24:603, 24:621, 24:656,
 25:2827, 25:2831, 25:3140, 25:3147,
 25:3153, 25:3163, 25:3170, 25:3174,
 25:3177, 25:3180, 25:3189, 25:3196,
 25:3198, 25:3233, 26:125, 26:129,
 26:131, 28:411, 28:461, 29:772,
 29:805, 29:821, 29:837, 29:897, 764
 \f@shape@saved 26:125, 26:131
 \f@size 21:198, 21:893,
 21:1247, 24:89, 24:380, 24:386,
 24:459, 24:604, 24:643, 24:677,
 24:719, 24:720, 24:723, 24:724,
 26:122, 26:143, 26:168, 26:170,
 26:181, 26:201, 26:216, 26:219,
 26:222, 26:227, 26:234, 26:241,
 26:253, 26:256, 26:262, 26:268,
 26:285, 26:286, 26:289, 26:294,
 26:376, 26:383, 26:402, 26:404,
 26:419, 26:470, 26:472, 26:474,
 26:490, 26:491, 26:496, 26:510,
 26:522, 26:527, 26:539, 26:547,
 26:552, 26:578, 26:592, 29:639,
 29:648, 29:686, 29:693, 33:66, 33:892
 \f@user@size
 26:490, 26:495, 26:539, 26:552
 \f@warn@break 1379
 \famdef@ult 29:523
 \filec@ntents
 50:1252, 50:1255, 50:1258,
 50:1269, 50:1294, 50:1387, 50:1390,
 50:1393, 50:1404, 50:1425, 50:1518,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

50:1540, 50:1628, 50:1853, 1389	\fl@ShowFloat 54:2481, 54:2487, 54:2502
\filec@ntents@checkdir	\fl@trace 54:245, 54:272,
50:1275, 50:1277, 50:1295,	54:328, 54:356, 54:363, 54:384,
50:1410, 50:1412, 50:1426, 50:1525	54:431, 54:479, 54:521, 54:531,
\filec@ntents@force	54:532, 54:533, 54:534, 54:544,
50:1271, 50:1406, 50:1521	54:545, 54:546, 54:547, 54:548,
\filec@ntents@noheader	54:558, 54:1132, 54:1151, 54:1170,
50:1273, 50:1408, 50:1523	54:1188, 54:1190, 54:1329, 54:1333,
\filec@ntents@nosearch	54:1345, 54:1346, 54:1347, 54:1348,
50:1274, 50:1409, 50:1524	54:1354, 54:1357, 54:1365, 54:1369,
\filec@ntents@nowarn 50:1279	54:1380, 54:1385, 54:1390, 54:1391,
\filec@ntents@opt	54:1392, 54:1393, 54:1400, 54:1403,
50:1255, 50:1258, 50:1260,	54:1411, 54:1422, 54:1428, 54:1433,
50:1390, 50:1393, 50:1395, 50:1520	54:1438, 54:1444, 54:1445, 54:1450,
\filec@ntents@OPTION 1131	54:1456, 54:1457, 54:1458, 54:1466,
\filec@ntents@overwrite	54:1470, 54:1475, 54:1479, 54:1484,
50:1272, 50:1407, 50:1522	54:1495, 54:1496, 54:1498, 54:1516,
\filec@ntents@warning	54:1525, 54:1531, 54:1540, 54:1543,
50:1280, 50:1282, 50:1327	54:1549, 54:1559, 54:1563, 54:1573,
\filec@ntents@where	54:1579, 54:1584, 54:1589, 54:1594,
50:1276, 50:1278, 50:1307,	54:1595, 54:1600, 54:1605, 54:1606,
50:1411, 50:1413, 50:1438, 50:1526	54:1607, 54:1615, 54:1619, 54:1624,
\filename@area . . . 20:661, 20:676,	54:1628, 54:1633, 54:1644, 54:1645,
20:686, 20:718, 20:719, 20:723,	54:1647, 54:1665, 54:1673, 54:1677,
20:748, 20:751, 20:776, 20:792,	54:1685, 54:1688, 54:1694, 54:1704,
20:794, 01:230, 01:236, 01:243,	54:1708, 54:1717, 54:1723, 54:1729,
01:249, 01:256, 01:262, 01:269, 50:984	54:1735, 54:1742, 54:1744, 54:1750,
\filename@base 20:661,	54:1755, 54:1757, 54:1759, 54:1767,
20:676, 20:686, 20:718, 20:720,	54:1772, 54:1778, 54:1783, 54:1789,
20:723, 20:748, 20:751, 20:783,	54:1803, 54:1804, 54:1807, 54:1828,
20:792, 01:279, 01:286, 01:299, 50:985	54:1837, 54:1843, 54:1852, 54:1855,
\filename@dot 01:297, 01:303	54:1862, 54:1872, 54:1876, 54:1888,
\filename@dots . 01:281, 01:283, 01:288	54:1894, 54:1899, 54:1904, 54:1908,
\filename@ext 20:662, 20:677,	54:1913, 54:1914, 54:1921, 54:1926,
20:687, 20:714, 20:715, 20:718,	54:1930, 54:1937, 54:1946, 54:1950,
20:721, 20:723, 20:744, 20:745,	54:1954, 54:1955, 54:1959, 54:1960,
20:748, 20:751, 20:784, 01:278,	54:1970, 54:1976, 54:1981, 54:1986,
01:285, 01:295, 01:297, 50:986, 1170	54:1990, 54:1994, 54:1995, 54:2002,
\filename@parse	54:2007, 54:2011, 54:2018, 54:2027,
01:94, 20:659, 20:674, 20:684,	54:2031, 54:2035, 54:2036, 54:2040,
20:713, 20:743, 20:781, 01:226, 50:983	54:2041, 54:2050, 54:2056, 54:2062,
\filename@path	54:2068, 54:2072, 54:2078, 54:2080,
01:231, 01:232, 01:237, 01:244,	54:2088, 54:2093, 54:2098, 54:2106,
01:245, 01:250, 01:257, 01:258, 01:263	54:2115, 54:2120, 54:2125, 54:2127,
\filename@simple	54:2132, 54:2134, 54:2145, 54:2151,
01:234, 01:247, 01:260,	54:2161, 54:2167, 54:2171, 54:2172,
01:270, 01:274, 01:276, 01:291, 01:293	54:2177, 54:2178, 54:2184, 54:2187,
\finish@module@release . . 03:89, 03:93	54:2188, 54:2189, 54:2196, 54:2197,
\finph@nt 38:104, 38:106,	54:2198, 54:2206, 54:2211, 54:2223,
38:110, 38:111, 38:119, 38:120, 1322	54:2224, 54:2231, 54:2234, 54:2242,
\finsm@sh . . . 38:134, 38:141, 38:147,	54:2246, 54:2250, 54:2251, 54:2255,
38:153, 38:154, 38:158, 38:159, 1322	54:2256, 54:2266, 54:2272, 54:2282,
\fix@penalty 32:102	54:2288, 54:2292, 54:2293, 54:2299,
\fixed@sfcnt . . . 26:582, 26:583, 26:584	54:2300, 54:2307, 54:2310, 54:2311,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 54:2312, 54:2320, 54:2321, 54:2322,
54:2331, 54:2336, 54:2349, 54:2351,
54:2358, 54:2361, 54:2370, 54:2374,
54:2378, 54:2379, 54:2383, 54:2384,
54:2436, 54:2441, 54:2447, 54:2457,
54:2464, 54:2478, 54:2479, 54:2483,
54:2494, 54:2506, 54:2614, 54:2627,
54:2628, 54:2632, 54:2635, 54:2637,
54:2640, 54:2643, 54:2645, 54:2686,
54:2693, 54:2698, 54:2704, 54:2709,
54:2713, 54:2719, 54:2732, 54:2734,
54:2741, 54:2746, 54:2751, 54:2753,
54:2759, 54:2761, 54:2768, 54:2782,
54:2784, 54:2791, 54:2796, 54:2801,
54:2803, 54:2809, 54:2811, 54:2818,
54:2847, 54:2849, 54:2864, 54:2866,
54:2880, 54:2890, 54:2893, 54:2898,
54:2951, 54:2968, 54:2973, 54:2981
- `\fl@tracemessage`
.. 54:2478, 54:2495, 54:2504, 54:2506
- `\fl@traceval`
..... 54:2488, 54:2489, 54:2490,
54:2491, 54:2494, 54:2503, 54:2506
- `\float@count`
02:47, 02:48, 02:49, 02:58, 02:111,
02:128, 02:134, 02:136, 02:137, 02:141
- `\fmtversion@topatch` 57:737,
57:739, 57:751, 57:752, 57:764, 57:772
- `\font@info` 26:100, 26:382, 26:451, 26:456
- `\font@name` .. 21:197, 21:200, 24:87,
24:257, 24:259, 24:451, 24:466,
24:642, 24:676, 26:85, 26:89, 26:91,
26:106, 26:142, 26:145, 26:155,
26:169, 26:172, 26:331, 26:332,
26:333, 26:334, 26:335, 26:340, 1357
- `\font@submax` 26:458, 26:487, 26:488,
37:55, 37:57, 37:120, 37:122, 37:159,
37:161, 57:301, 57:303, 57:312, 1372
- `\fps@dbl` 45:34
- `\freeze@math@version` . 28:155, 28:274
- `\frozen@everydisplay`
.. 24:418, 24:423, 24:440, 24:442, 711
- `\frozen@everymath`
..... 24:418, 24:432, 24:444, 711
- `\g@addto@macro` 09:77,
25:3251, 28:475, 50:168, 50:495,
50:1163, 50:1179, 50:1180, 06:1010,
53:399, 53:405, 53:410, 53:461, 252
- `\G@refundefined` 1395
- `\G@refundefinedfalse` 35:5, 841
- `\G@refundefinedtrue` 35:3,
35:16, 35:33, 35:50, 36:225, 36:232,
36:243, 36:249, 47:41, 47:68, 47:85, 861
- `\gen@sfcnt` 26:519, 26:520, 26:521
- `\genb@sfcnt` 26:524, 26:525, 26:526, 1393
- `\genb@x` 26:527, 26:529
- `\genb@y` 26:529
- `\get@cdp` 28:643, 28:651, 28:684
- `\get@external@font` 26:84, 26:97, 26:553
- `\getanddefine@fonts` 24:689, 24:707,
26:321, 28:60, 28:88, 28:133, 28:160,
28:191, 28:221, 28:252, 28:299,
28:349, 28:510, 28:594, 28:648,
28:650, 28:667, 28:790, 28:791,
28:823, 28:824, 28:1266, 28:1267, 711
- `\glb@currsize`
... 20:41, 20:111, 20:169, 24:409,
26:218, 26:253, 26:257, 26:263, 26:286
- `\glb@settings`
..... 24:410, 26:218, 26:265, 26:296
- `\gobble@finish@module@release` ...
..... 03:109, 03:111, 03:170
- `\gobble@font@spec` 1391
- `\group@elt` 28:36, 28:508,
28:555, 28:556, 28:587, 28:591, 28:1298
- `\group@list` 28:512, 28:562,
28:585, 28:590, 28:591, 28:640,
28:866, 28:914, 28:975, 28:1060,
28:1063, 28:1114, 28:1117, 28:1183,
28:1186, 28:1253, 28:1304, 1400
- `\h@false` 38:81
- `\h@true` 38:82, 38:83
- `\hb@xt@` 21:446, 06:29, 38:210, 38:438,
38:467, 38:548, 38:563, 38:575,
38:602, 38:633, 40:48, 40:63, 40:86,
40:104, 40:253, 40:272, 40:725,
40:729, 40:730, 40:731, 41:37, 42:42,
42:54, 42:73, 42:85, 42:116, 42:128,
42:276, 42:310, 42:313, 42:316,
42:318, 42:325, 42:384, 42:463,
42:598, 42:608, 42:751, 42:807,
44:243, 44:266, 44:273, 54:920,
54:935, 54:985, 54:995, 54:1044,
54:1054, 54:2420, 54:2884, 54:2885,
54:2889, 54:2927, 54:2928, 54:2932,
54:2956, 54:2957, 54:2963, 02:424, 1388
- `\hexnumber@` . 28:887, 28:895, 28:930,
28:938, 28:959, 28:969, 28:995,
28:1003, 28:1011, 28:1020, 28:1023,
28:1032, 28:1033, 28:1072, 28:1080,
28:1126, 28:1134, 28:1153, 28:1154,
28:1164, 28:1165, 28:1170, 28:1196,
28:1204, 28:1209, 28:1211, 29:756
- `\hgl@` 02:379, 02:380
- `\hmode@bgroup` 21:85, 21:93, 21:347,
21:376, 21:411, 21:417, 21:448,
21:459, 21:466, 21:499, 21:506,
21:509, 21:511, 21:519, 21:535,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- 21:754, 21:784, 21:790, 21:822,
21:829, 21:857, 21:860, 21:917,
21:1281, 32:7, 33:589, 33:596, 1402
- \hmode@start@before@group
21:86, 21:169, 21:171, 21:177, 21:202
- \hyper@nopatch@longtable 55:461
- \if@afterindent 44:124, 44:131
- \if@compatibility 50:2, 50:711
- \if@endpe 37:340,
37:355, 37:372, 37:382, 39:140, 917
- \if@eqnsw 38:415, 38:516
- \if@fcolmade
54:73, 54:269, 54:399, 54:408,
54:446, 54:456, 54:1130, 54:1150,
54:1168, 54:1197, 54:1277, 54:2846,
54:2863, 54:2897, 54:2939, 54:2978
- \if@filesw 20:5, 20:36,
20:106, 20:164, 20:317, 20:329,
20:350, 20:379, 20:391, 20:406,
20:426, 20:438, 20:445, 20:457,
37:22, 37:61, 37:89, 37:126, 37:151,
37:165, 44:153, 47:4, 47:8, 47:39,
47:48, 47:56, 47:67, 47:84, 50:1305,
50:1322, 50:1436, 50:1455, 53:413, 1383
- \if@firstamp 41:328
- \if@firstcolumn ... 54:73, 54:251,
54:284, 54:401, 54:449, 54:2392,
54:2877, 54:2906, 54:2947, 1067
- \if@font@series@context
29:588, 29:609, 29:619, 754
- \if@forced@series
25:2789, 29:166, 29:238, 657
- \if@ignore 28:317, 28:356,
37:4, 37:341, 37:358, 37:374, 37:383
- \if@in@minipage@env .. 40:486, 40:508
- \if@includeinrelease
03:73, 03:76, 03:132, 06:964
- \if@inlabel .. 39:28, 39:65, 39:102,
39:191, 39:214, 54:140, 54:167, 54:193
- \if@inmath 1397
- \if@insert 54:73,
54:1408, 54:1521, 54:1555, 54:1670,
54:1700, 54:1833, 54:1868, 54:1943,
54:2024, 54:2112, 54:2239, 54:2367
- \if@listfiles@hashes
20:762, 20:797, 20:804, 20:808
- \if@listfiles@sizes
20:762, 20:798, 20:802
- \if@minipage
16:26, 18:262, 18:277, 18:294,
18:313, 18:346, 18:380, 37:530,
37:552, 39:180, 40:483, 41:79, 45:20
- \if@mparswitch 54:73, 54:2394
- \if@multiplelabels 35:93
- \if@mypkg@draft 1150
- \if@negarg 42:168, 42:209, 42:223, 42:284
- \if@newlist .. 37:572, 39:29, 39:33,
39:69, 39:78, 39:106, 39:197, 54:877,
54:941, 54:954, 54:999, 54:1012, 54:1058
- \if@nmbrlist 39:33, 39:232
- \if@no@font@opt . 27:16, 27:110, 27:129
- \if@nobreak 18:121, 18:150,
18:171, 18:315, 18:348, 18:382,
20:203, 20:218, 20:232, 39:198,
39:223, 40:446, 40:467, 44:47,
44:128, 45:180, 45:373, 48:270,
49:48, 49:58, 49:71, 49:79, 49:94,
49:102, 54:144, 54:171, 54:197,
54:340, 54:1500, 54:1649, 54:1810, 1385
- \if@noitemarg 39:32, 39:230
- \if@noparitem 39:30, 39:188
- \if@noparlist 39:31, 39:114
- \if@noskipsec 18:150, 18:171,
39:58, 40:447, 40:468, 44:38, 44:40,
44:97, 45:374, 54:134, 54:161, 54:187
- \if@ovb 42:438,
42:505, 42:536, 42:561, 42:572, 42:585
- \if@ovhline 42:470, 42:600
- \if@ovl 42:438,
42:503, 42:532, 42:557, 42:602, 42:611
- \if@ovr 42:438,
42:502, 42:531, 42:556, 42:599, 42:609
- \if@ovt 42:438,
42:504, 42:535, 42:560, 42:577, 42:589
- \if@ovvline 42:470, 42:575
- \if@partsw . 20:5, 20:321, 20:383, 20:430
- \if@pboxsw 40:415, 40:436, 40:592
- \if@reversemargin 54:73, 54:2397
- \if@rjfield 41:19, 41:33
- \if@skipping@module
03:137, 03:149, 03:152
- \if@specialpage
54:73, 54:886, 54:961, 54:1019
- \if@tempswa .. 12:9, 20:327, 20:389,
20:436, 21:1570, 24:100, 24:786,
28:543, 28:618, 28:692, 28:773,
28:1297, 37:63, 37:128, 37:167,
37:537, 37:558, 47:95, 50:1334,
50:1467, 50:1556, 52:292, 52:293,
52:315, 52:316, 52:336, 52:337,
54:1341, 54:1377, 54:2203, 54:2328,
02:156, 01:62, 01:63, 01:64, 1405
- \if@test 54:8, 54:9,
54:1238, 54:1257, 54:1297, 54:1319,
54:1383, 54:1468, 54:1477, 54:1617,
54:1626, 54:1770, 54:1781, 54:1924,
54:2005, 54:2091, 54:2209, 54:2334

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \if@twocolumn 20:26,
20:96, 20:153, 45:32, 45:210, 45:235,
54:73, 54:118, 54:272, 54:283,
54:400, 54:448, 54:472, 54:1132,
54:1188, 54:2391, 54:2848, 54:2865
- \if@twoside
54:73, 54:117, 54:890, 54:964, 54:1022
- \ifdt@p 38:203, 38:205
- \IfFileExists@ 20:490, 20:517
- \IfFileExists@@ 20:517
- \ifG@refundefined 35:3, 35:4, 35:5
- \ifh@ 38:76, 38:114, 38:123
- \ifin@ 21:1590, 21:1593, 27:50, 27:52,
28:3, 28:23, 28:497, 28:639, 28:641,
28:702, 28:715, 28:785, 28:787,
28:815, 28:867, 28:881, 28:915,
28:927, 28:976, 28:992, 28:1061,
28:1064, 28:1085, 28:1115, 28:1118,
28:1181, 28:1184, 28:1187, 28:1254,
28:1256, 28:1285, 29:303, 29:307,
29:311, 50:271, 50:289, 50:583, 50:617
- \ifmath@fonts 24:267, 26:223
- \ifmaybe@ic 32:83, 32:92, 1405
- \ifnot@nil 26:360, 26:377, 26:398
- \ifrestore@version 1396
- \iftc@forced .. 33:832, 33:843, 33:1111
- \ifv@ 38:75, 38:113, 38:122
- \ifx 348
- \in@ 21:1588,
21:1591, 27:49, 27:51, 28:3, 28:22,
28:496, 28:638, 28:640, 28:698,
28:711, 28:784, 28:786, 28:813,
28:865, 28:876, 28:913, 28:924,
28:974, 28:988, 28:1059, 28:1062,
28:1082, 28:1113, 28:1116, 28:1178,
28:1182, 28:1185, 28:1252, 28:1255,
28:1283, 29:301, 29:305, 29:309,
50:270, 50:287, 50:580, 50:616, 1405
- \in@@ 25:2877, 25:2878,
25:2880, 25:2881, 28:6, 28:7, 28:8, 28:10
- \in@false 28:11
- \in@true 28:13
- \init@restore@glb@settings
26:266, 26:269, 26:271
- \init@restore@version 28:63,
28:92, 28:109, 28:124, 28:125, 1396
- \init@series@setup 25:3252,
25:3257, 29:121, 29:295, 29:322, 29:323
- \input@path 01:93,
01:115, 01:117, 01:123, 01:125,
01:131, 01:133, 01:138, 01:140,
01:150, 20:503, 20:557, 20:579,
20:606, 20:623, 01:217, 52:275, 1161
- \insc@unt 02:37, 02:47,
02:48, 02:49, 02:58, 02:149, 02:150,
02:151, 02:152, 02:153, 02:157, 02:159
- \install@mathalphabet
24:684, 24:701, 24:708,
28:516, 28:519, 28:645, 28:646,
28:743, 28:795, 28:798, 28:805,
28:820, 28:821, 28:828, 28:1268, 28:1270
- \is@range 26:393, 26:394
- \kernel@ifnextchar .. 03:90, 06:99,
06:118, 06:191, 50:475, 06:875, 06:890
- \kernel@make@fragile 18:25,
18:26, 18:27, 18:28, 18:29, 21:190,
21:191, 37:489, 37:490, 37:491,
38:90, 38:91, 38:92, 38:93, 38:179,
38:180, 38:181, 06:479, 41:160,
41:161, 41:162, 42:833, 42:834,
42:835, 42:836, 42:837, 42:838,
42:839, 42:840, 42:841, 42:842,
42:843, 42:844, 44:23, 44:24, 44:25,
44:26, 44:27, 49:85, 49:86, 06:991,
06:992, 06:993, 06:994, 06:995,
06:996, 06:997, 06:998, 06:999,
06:1000, 06:1001, 06:1002, 06:1003,
06:1004, 06:1005, 06:1006, 1092
- \l@ngrel@x 06:79, 06:80,
06:88, 06:89, 06:93, 06:144, 06:156,
06:208, 06:311, 06:312, 06:314, 86
- \l@nohyphenation
37:534, 37:726, 57:285, 1410
- \last@fontshape
24:617, 24:635, 24:652, 24:669
- \latexrelease@postltxcmd 07:3329
- \latexrelease@postltxexpl 05:146
- \leavevmode@ifvmode
18:559, 18:560, 18:568,
30:635, 30:637, 30:639, 30:641,
38:115, 38:154, 38:219, 38:239, 38:240
- \load@onefile@withoptions
50:988, 50:1030, 50:1142, 1124
- \load@onefile@withoptions
50:935, 50:1092, 50:1644, 1122
- \lower@bound ... 26:403, 26:404, 26:415
- \lst@vskip 453
- \LT@cols 1325
- \ltx@sh@ft .. 21:411, 21:418, 21:499,
21:507, 21:784, 21:791, 02:416, 1405
- \ltx@star@counter
35:140, 35:145, 35:158, 35:172
- \m@ne 02:39, 421
- \m@th . 19:13, 30:348, 30:470, 30:472,
30:473, 30:476, 30:517, 30:541,
30:544, 30:548, 30:551, 30:558,
30:561, 30:568, 30:571, 30:653,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 38:68, 38:71, 38:106, 38:140, 38:147,
 38:167, 38:169, 38:185, 38:204,
 38:429, 38:458, 38:563, 38:575,
 38:602, 38:613, 40:415, 40:436,
 40:660, 41:174, 41:227, 44:239,
 44:262, 45:400, 45:441, 45:447,
 45:500, 45:506, 02:396, 02:408, *1395*
`\makeph@nt` 38:101, 38:103
`\makesm@sh` 38:131, 38:133
`\mandatory@arg` 26:431,
 26:518, 26:522, 26:527, 26:534,
 26:536, 26:541, 26:543, 26:548,
 26:550, 26:563, 26:579, 26:586, 26:588
`\math@bgroup` 24:715,
 26:307, 26:313, 28:54, 28:82, 28:147,
 28:176, 28:185, 28:207, 28:215,
 28:246, 32:131, 32:132, 32:139, *708*
`\math@egroup` 24:715, 26:311,
26:312, 32:132, 32:133, 32:140, *1358*
`\math@famname` *1357*
`\math@fonts` 24:685,
 24:690, 26:233, 26:337, 28:61,
 28:90, 28:161, 28:192, 28:222, 28:254
`\math@fontsfalse`
 19:7, 21:322, 21:349,
 21:380, 21:1283, 24:78, 24:269,
 24:279, 24:302, 33:67, 33:637, 33:893
`\math@fontstrue` 24:267, 24:727
`\math@group` *1357*
`\math@version`
 24:8, 24:399, 24:689, 24:693,
 24:695, 24:696, 24:698, 26:231,
 28:57, 28:60, 28:65, 28:66, 28:70,
 28:85, 28:89, 28:94, 28:95, 28:99,
 28:112, 28:113, 28:114, 28:127,
 28:128, 28:129, 28:150, 28:152,
 28:154, 28:155, 28:160, 28:164,
 28:166, 28:168, 28:172, 28:188,
 28:191, 28:195, 28:197, 28:199,
 28:203, 28:218, 28:221, 28:225,
 28:227, 28:229, 28:233, 28:249,
 28:253, 28:257, 28:259, 28:261,
 28:265, 28:296, 28:300, 29:728, *707*
`\math@version.` *1357*
`\mathchar@type` 28:959,
 28:969, 28:1020, 28:1023, 28:1032,
 28:1048, 28:1153, 28:1164, 28:1227
`\mathph@nt` 38:99, 38:105
`\mathsm@sh`
 38:129, 38:138, 38:139, 38:145, 38:146
`\maybe@ic` 32:64, 32:65, 32:84
`\maybe@ic@` 32:84
`\maybe@icfalse` 32:98
`\maybe@ictrue` 32:88
`\maybe@load@fontshape`
 21:89, 25:2864,
 25:2903, 26:128, 29:186, 29:254, *516*
`\maybe@update@bfseries@defaults`
 29:39, 29:336, 29:345, 29:383
`\maybe@update@mdseries@defaults`
 29:40, 29:359, 29:368, 29:384
`\mb@b` 40:76, 40:87, 40:95, 40:105
`\mb@l` 40:76, 40:80, 40:86,
 40:95, 40:99, 40:104, 42:148, 42:152
`\mb@r` 40:76, 40:80, 40:86,
 40:95, 40:99, 40:104, 42:148, 42:152
`\mb@t` 40:77, 40:84, 40:96, 40:103
`\md@def@ult` 29:476
`\mddef@ult` 29:220, 29:279,
 29:372, 29:373, 29:374, 29:413,
 29:414, 29:415, 29:481, 29:522, *1415*
`\mddefault@previous` 29:369, 29:371,
 29:410, 29:412, 30:118, 30:128, *1418*
`\mdseries@..` *737*
`\mdseries@previous` *1418*
`\mdseries@rm` 29:112,
 29:128, 29:129, 29:145, 29:153,
 29:360, 29:372, 29:413, 29:418, 29:445
`\mdseries@sf` 29:113, 29:129, 29:154,
 29:361, 29:373, 29:414, 29:419, 29:446
`\mdseries@tt` 29:114, 29:129, 29:155,
 29:362, 29:374, 29:415, 29:420, 29:447
`\meaning` *332*
`\merge@font@series` 25:2796, 25:2813,
 25:2814, 25:2895, 25:2897, 26:134, *657*
`\merge@font@series@`
 25:2816, 25:2821, 25:2898
`\merge@font@series@without@substitution`
25:2848, 25:2863, 25:2900, 26:137, *658*
`\merge@font@series@without@substitution@`
 25:2848, 25:2901
`\merge@font@shape` 25:3146, 25:3159,
25:3160, 25:3231, 25:3236, 26:133
`\merge@font@shape@`
 25:3162, 25:3167, 25:3237
`\merge@font@shape@without@substitution`
 25:3186, 25:3239, 26:136
`\merge@font@shape@without@substitution@`
 25:3186, 25:3240
`\mv@{version}` *710*
`\mv@{version}@frozen` *707*
`\mv@{version}@reset` *710*
`\n@space`
 ... 30:636, 30:638, 30:640, 30:642,
 30:647, 30:648, 30:649, 30:650, 30:653
`\new@command` . 06:96, 06:95, 06:172,
 06:206, 06:225, 06:296, 06:317, 06:343
`\new@environment` 06:188, 06:187, 06:200

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\new@fontshape</code>	27:2, 27:4, 27:22, 27:24	29:167, 29:170, 37:254, 37:275,	
<code>\new@label@record</code>		37:293, 37:303, 40:194, 50:996,	
...	36:57 , 36:150 , 36:301, 37:26, 1430	50:1131, 55:150, 55:163, 55:177,	
<code>\new@mathalphabet</code>	28:696, 28:717, 28:728	08:538, 08:629, 08:1500, 08:1535, 1442	
<code>\new@mathgroup</code>		<code>\operator@font</code>	30:654 , 38:3, 38:4,
04:27, 24:15 , 28:546, 02:74, 02:76, 1390		38:5, 38:6, 38:7, 38:8, 38:9, 38:10,	
<code>\new@mathversion</code>		38:11, 38:12, 38:13, 38:14, 38:15,	
. 28:21 , 28:479, 28:487, 28:492, 28:495		38:16, 38:17, 38:18, 38:19, 38:20,	
<code>\new@module@skip</code>	03:138, 03:150, 03:152	38:21, 38:22, 38:23, 38:24, 38:25,	
<code>\new@moduledate</code>		38:26, 38:27, 38:28, 38:29, 38:30,	
..... 03:82, 03:101, 03:104, 03:152		38:31, 38:32, 38:33, 38:34, 38:37, 38:40	
<code>\new@modulename</code>	03:96, 03:152	<code>\optional@arg</code>	
<code>\new@symbolfont</code>	28:547, 28:589	26:432, 26:511, 26:513, 26:585, 26:588	
<code>\newcommand</code>	337	<code>\outer@nobreak</code>	45:181,
<code>\NewCommandCopy</code>	337	45:251, 45:255 , 45:346, 45:364, 1385	
<code>\NewDocumentCommand</code>	329	<code>\outputbox@append</code>	1242
<code>\newlinechar</code>	348	<code>\p@</code>	 02:181
<code>\newmathalphabet@</code>	27:14	<code>\p@...</code>	1413
<code>\newmathalphabet@@</code>	27:109	<code>\p@enum</code>	922
<code>\newmathalphabet@@@</code>	27:15, 27:109	<code>\p@equation</code>	38:426, 38:455, 38:611
<code>\nfss@catcodes</code>	24:20, 24:121,	<code>\p@renwd</code>	1368
24:484, 24:485, 24:492 , 24:539,		<code>\p@selectfont</code>	26:120, 1358
30:41, 30:47, 30:57, 30:146, 1393		<code>\par</code>	191
<code>\nfss@text</code>	21:335, 21:337, 29:759 ,	<code>\par@deathcycles</code>	
32:5, 32:123 , 35:17, 35:34, 35:51, 1380		39:56, 39:77, 39:79, 39:80	
<code>\no@alphabet@error</code>	24:5,	<code>\patch@level</code>	03:1,
28:515, 28:517, 28:733, 28:734,		03:39, 03:44, 03:46, 03:48, 03:52,	
28:748, 28:757, 28:843, 28:844, 1372		03:61, 57:740, 57:752, 57:754, 1408	
<code>\no@alphabet@help</code>	1358	<code>\patchcmd</code>	329
<code>\no@version@warning</code>	1358	<code>\pdfannot@link@off@@</code>	
<code>\noaccents@</code>	24:730 , 30:140	54:846, 54:915, 54:930	
<code>\noexpand</code>	350	<code>\pdfannot@link@on@@</code>	
<code>\non@alpherr</code>	24:709,	54:845, 54:923, 54:938	
24:711, 28:73, 28:102, 28:118 ,		<code>\ph@nt</code>	38:81, 38:82, 38:83, 38:97
28:175, 28:206, 28:236, 28:268, 28:1305		<code>\pickup@font</code>	21:199,
<code>\not@base</code>	29:876, 29:880, 29:881,	24:258, 24:450 , 24:644, 24:678,	
29:882, 29:883, 29:884, 29:885,		26:144, 26:171, 26:332, 26:334, 26:336	
29:886, 29:887, 29:888, 29:889, 29:890		<code>\pictur@</code>	 42:21
<code>\not@math@alphabet</code>	25:2915,	<code>\pkgcls@arg</code>	50:1655, 50:1787
25:2920, 25:2925, 25:3203, 25:3207,		<code>\pkgcls@candidate</code>	
25:3210, 25:3213, 25:3216, 25:3219,		50:1642, 50:1664, 50:1741,	
25:3222, 25:3225, 29:6, 29:9, 29:12,		50:1745, 50:1749, 50:1818, 50:1821	
29:15, 29:18, 29:21, 29:24, 29:27,		<code>\pkgcls@debug</code>	50:1631, 50:1648,
29:30, 29:334, 29:357, 29:387,		50:1649, 50:1650, 50:1651, 50:1652,	
29:408, 29:432, 29:443, 29:460,		50:1717, 50:1718, 50:1719, 50:1720,	
29:463, 29:485, 29:490, 29:495,		50:1729, 50:1734, 50:1752, 50:1761,	
29:528, 29:533, 29:538, 29:554,		50:1777, 50:1811, 50:1812, 50:1813	
29:557, 29:560, 29:563, 29:566, 29:708		<code>\pkgcls@ext</code>	50:1656, 50:1705, 1432
<code>\now@and@everyjob</code>		<code>\pkgcls@innerdate</code>	
04:214, 04:220, 21:1035		50:140, 50:1636 , 50:1689, 50:1692,	
<code>\o@lign</code>	21:411, 21:418,	50:1693, 50:1699, 50:1839, 1142	
21:499, 21:507, 21:784, 21:791, 02:410		<code>\pkgcls@mindate</code>	50:1669,
<code>\on@line</code>	08:2163,	50:1678, 50:1695, 50:1700, 1141	
08:2177, 10:25, 14:8, 14:15, 14:214 ,		<code>\pkgcls@name</code>	50:1654, 50:1705

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\pkgcls@outerdate</code>	40:583, 44:60, 45:534, 45:552, 45:570, 50:574, 50:957, 50:1224, 1427
<code>\pkgcls@parse@date@arg</code>	<code>\protected@file@percent</code>
..... 50:1663, 50:1674	 37:199, 37:206, 37:221, 37:229, 37:230, 44:165, 44:172, 870
<code>\pkgcls@parse@date@arg@</code>	<code>\protected@wlog</code>
..... 50:1680, 50:1683	 50:421, 50:423, 50:437, 50:451
<code>\pkgcls@parse@date@arg@version</code> ..	<code>\protected@write</code> .. 20:202, 20:207, 35:101, 35:119, 35:128, 36:55, 44:189, 44:199, 46:14, 46:31, 1383
..... 50:1690, 50:1712	<code>\protected@xdef</code>
<code>\pkgcls@releasedate</code>	 06:357, 44:11, 45:515, 45:583, 45:599, 50:413, 50:432, 50:485, 50:854, 1383
.. 50:1642, 50:1746, 50:1750, 50:1822	<code>\provide@command</code>
<code>\pkgcls@rollbackdate@error</code>	 06:220, 06:219
..... 50:1742, 50:1801, 50:1819	<code>\ps@empty</code>
<code>\pkgcls@show@selection</code>	 49:10, 57:158
.. 50:1769, 50:1775, 50:1825, 50:1830	<code>\ps@plain</code>
<code>\pkgcls@targetdate</code>	 49:13
..... 50:1636, 50:1676, 50:1684, 50:1687, 50:1688, 50:1693, 50:1701, 50:1702, 50:1715, 50:1723, 50:1738, 50:1740, 50:1770, 50:1782, 50:1784, 50:1809, 50:1815, 50:1817, 1141	<code>\qcurr@file</code> 50:1252, 50:1296, 50:1298, 50:1303, 50:1329, 50:1427, 50:1429, 50:1434, 50:1462, 1417
<code>\pkgcls@targetlabel</code>	<code>\quote@@name</code>
..... 50:1636, 50:1677, 50:1698, 50:1713, 50:1725, 50:1757, 50:1791, 50:1829, 50:1832, 1141	 20:481, 20:497
<code>\pkgcls@use@this@release</code>	<code>\quote@name</code>
..... 50:1726, 50:1743, 50:1745, 50:1758, 50:1768, 50:1821	 20:481, 20:494, 20:496, 20:607, 20:609, 20:618, 50:1427
<code>\pr@@@s</code>	<code>\r@@t</code>
..... 38:259, 38:267	 38:66
<code>\pr@@@t</code>	<code>\reenable@package@load</code>
..... 38:262, 38:268	 52:477, 53:517, 1164
<code>\pr@m@s</code>	<code>\reinstall@nfss@defs</code>
..... 38:256, 38:257	 25:3205, 25:3246, 25:3250, 25:3252, 25:3256, 25:3257, 25:3260
<code>\protectedrel@x</code>	<code>\relax</code>
..... 06:77, 06:94, 06:143, 06:294, 86	 350
<code>\preload@sizes</code> 27:11, 27:94, 1357	<code>\rem@pt</code>
<code>\prepare@family@series@update</code> ...	 24:392
..... 29:118, 29:161, 29:162, 29:234, 29:237, 29:287, 29:290, 29:486, 29:491, 29:496, 29:529, 29:534, 29:539, 751	<code>\remove@angles</code>
<code>\pretocmd</code>	 26:364, 26:387
..... 328	<code>\remove@nil</code>
<code>\prim@s</code>	 28:37
..... 38:253, 38:255, 38:267	<code>\remove@star</code>
<code>\prime@s</code>	 26:364, 26:370
..... 38:254	<code>\remove@tlig</code> 21:1025, 21:1027, 21:1029, 21:1053, 21:1058, 21:1062, 21:1109, 21:1110, 21:1111, 1417
<code>\process@table</code>	<code>\remove@to@nnil</code>
..... 20:40, 20:110, 20:168, 28:367	 24:391, 26:364, 26:390, 26:503
<code>\propagate@doendpe</code>	<code>\renew@command</code>
..... 39:140, 917	 06:164, 06:163, 06:226, 06:234
<code>\protectd@edf</code>	<code>\renew@environment</code> ... 06:194, 06:193
..... 1426	<code>\requested@test@context</code>
<code>\protected</code>	 29:582, 29:601, 29:605, 754
..... 307	<code>\reserved@b</code>
<code>\protected@cmd</code>	 726
..... 1390	<code>\reserved@a</code>
<code>\protected@edef</code>	 03:13, 03:19, 03:34, 01:105, 01:109, 01:110, 03:181, 03:182, 13:33, 13:37, 14:234, 18:478, 18:481, 20:216, 20:217, 20:229, 20:230, 20:254, 20:263, 20:272, 20:325, 20:387, 20:434, 20:504, 20:506, 20:511, 20:513, 20:525, 20:528, 20:531, 20:534, 20:535, 20:536, 20:558, 20:560, 20:565, 20:567, 20:568, 20:569,
... 22:359, 29:629, 29:681, 35:148, 35:161, 35:168, 35:179, 35:187, 35:194, 35:202, 35:204, 35:213, 35:219, 36:40, 36:45, 36:83, 36:84, 06:263, 06:357, 40:549, 40:566,	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

20:580, 20:582, 20:587, 20:589,
 20:604, 20:610, 20:614, 20:621,
 20:627, 20:631, 20:660, 20:663,
 20:664, 20:666, 20:675, 20:678,
 20:685, 20:688, 20:710, 20:711,
 20:712, 20:716, 20:724, 20:741,
 20:742, 20:746, 20:752, 20:782,
 20:786, 20:789, 20:794, 21:99,
 21:100, 21:104, 21:107, 21:115,
 21:125, 21:128, 21:137, 21:156,
 21:161, 21:1019, 21:1023, 21:1071,
 21:1073, 21:1074, 21:1076, 21:1078,
 21:1086, 21:1095, 24:31, 24:32,
 24:33, 24:42, 24:45, 24:48, 24:64,
 24:67, 24:70, 24:106, 24:109, 24:111,
 24:148, 24:152, 24:486, 01:179,
 24:489, 01:180, 24:616, 24:617,
 24:632, 24:635, 24:640, 24:651,
 24:652, 24:665, 24:669, 24:674,
 24:701, 24:704, 24:705, 24:713,
 01:183, 01:201, 01:205, 25:2823,
 25:2824, 25:2827, 25:2828, 25:2843,
 25:2844, 25:2856, 25:2857, 25:3168,
 25:3169, 25:3172, 25:3173, 25:3194,
 25:3195, 26:197, 26:199, 26:201,
 26:211, 26:213, 26:216, 05:26,
 26:361, 26:362, 05:27, 05:28, 26:375,
 26:376, 05:29, 05:30, 05:52, 05:57,
 27:53, 27:57, 05:94, 05:100, 05:110,
 01:227, 28:532, 28:535, 28:607,
 28:610, 28:643, 28:652, 28:654,
 28:698, 28:701, 05:144, 28:711,
 28:714, 28:812, 28:814, 28:876,
 28:880, 28:924, 28:926, 28:987,
 28:990, 28:1082, 28:1084, 28:1178,
 28:1180, 01:234, 28:1282, 28:1284,
 28:1300, 28:1302, 28:1303, 28:1308,
 29:41, 29:42, 29:74, 29:75, 01:237,
 29:196, 29:197, 29:261, 29:262,
 01:239, 01:240, 29:597, 29:598,
 01:247, 01:250, 01:252, 01:253,
 32:48, 32:49, 32:54, 32:55, 32:66,
 32:69, 32:89, 32:96, 01:260, 06:140,
 01:263, 06:145, 06:153, 06:156,
 01:265, 06:174, 06:175, 06:176,
 06:178, 06:225, 06:226, 06:227,
 06:233, 06:234, 06:235, 06:236,
 06:239, 35:144, 35:145, 35:146,
 35:157, 35:158, 35:159, 35:202,
 35:203, 35:204, 35:205, 35:213,
 35:214, 35:219, 35:220, 36:40, 36:41,
 36:45, 36:46, 36:83, 36:85, 06:263,
 06:269, 37:197, 37:198, 37:252,
 37:253, 37:258, 37:273, 37:274,
 37:278, 37:291, 37:292, 37:296,
 37:301, 37:302, 37:306, 37:391,
 37:392, 06:330, 06:334, 06:392,
 06:396, 38:511, 38:512, 38:513,
 38:514, 38:516, 06:430, 06:434,
 06:458, 06:462, 40:78, 40:79, 40:82,
 40:97, 40:98, 40:101, 40:189, 40:195,
 41:318, 41:322, 41:327, 41:346,
 41:437, 41:438, 06:573, 42:210,
 42:212, 42:216, 06:581, 06:582,
 01:315, 01:316, 01:317, 42:488,
 42:489, 42:516, 42:517, 42:545,
 42:546, 06:614, 06:622, 06:623,
 06:624, 06:625, 06:665, 06:668,
 45:29, 45:30, 45:32, 45:33, 45:63,
 45:67, 45:72, 45:74, 45:76, 45:78,
 45:83, 45:84, 45:132, 45:136, 45:142,
 45:145, 45:148, 45:151, 06:683,
 06:685, 06:688, 06:697, 06:719,
 06:726, 06:751, 06:753, 50:166,
 50:169, 50:171, 50:265, 50:273,
 50:277, 50:283, 50:291, 50:295,
 50:483, 50:485, 50:486, 50:487,
 50:491, 50:506, 50:508, 50:509,
 50:510, 50:514, 06:872, 06:881,
 50:691, 50:695, 50:701, 50:705,
 50:783, 50:784, 50:787, 50:856,
 50:860, 50:872, 50:873, 50:875,
 50:884, 50:888, 50:900, 50:901,
 50:903, 50:911, 50:915, 50:927,
 50:928, 50:929, 50:931, 50:943,
 50:946, 50:947, 50:949, 50:1033,
 50:1082, 50:1099, 50:1140, 50:1263,
 50:1264, 50:1266, 50:1398, 50:1399,
 50:1401, 50:1443, 50:1444, 50:1446,
 50:1450, 50:1666, 50:1671, 50:1724,
 50:1725, 50:1756, 50:1757, 50:1828,
 50:1829, 50:1853, 50:1855, 52:161,
 52:167, 52:168, 52:169, 54:26, 54:35,
 54:37, 54:38, 54:1228, 54:1248,
 54:2574, 54:2576, 54:2577, 54:2676,
 54:2678, 54:2684, 54:2687, 57:226,
 57:243, 57:244, 57:245, 57:252,
 57:253, 57:254, 57:538, 57:569,
 57:575, 57:576, 57:587, 57:588,
 57:589, 57:596, 57:597, 57:598,
 57:623, 57:624, 57:631, 57:635,
 57:637, 57:639, 57:644, 57:647,
 57:655, 57:656, 57:657, 57:753,
 57:756, 57:757, 57:774, 02:116, 1385
 \reserved@b . 01:106, 01:107, 13:33,
 13:34, 13:37, 18:479, 18:480, 18:487,
 20:323, 20:325, 20:385, 20:387,
 20:432, 20:434, 20:605, 20:607,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

20:609, 20:622, 20:624, 20:626,
 20:710, 20:711, 20:712, 20:785,
 20:787, 20:788, 20:795, 21:108,
 21:115, 21:130, 21:137, 21:1022,
 21:1023, 21:1072, 21:1073, 21:1095,
 21:1104, 21:1106, 24:32, 24:33,
 24:39, 24:96, 24:98, 24:151, 24:152,
 24:702, 24:713, 25:2842, 25:2843,
 25:2845, 05:53, 27:47, 27:54, 05:59,
 27:71, 27:73, 28:533, 28:534,
 28:535, 28:539, 28:541, 28:608,
 28:609, 28:610, 28:614, 28:616,
 28:651, 28:652, 28:653, 28:688,
 28:690, 28:769, 28:771, 28:816,
 28:817, 28:818, 28:825, 28:985,
 28:989, 28:991, 29:203, 29:208,
 29:212, 29:213, 29:220, 29:221,
 29:266, 29:273, 29:279, 32:53,
 32:54, 32:67, 32:69, 32:96, 32:97,
 06:127, 06:129, 06:141, 06:154,
 06:176, 06:177, 36:84, 36:85, 06:331,
 06:332, 06:334, 06:393, 06:394,
 06:396, 06:431, 06:432, 06:434,
 06:459, 06:460, 06:462, 41:323,
 41:325, 41:327, 06:666, 06:668,
 45:43, 45:44, 45:112, 45:113, 06:684,
 06:688, 06:723, 06:726, 50:266,
 50:267, 50:268, 50:270, 50:285,
 50:288, 50:483, 50:506, 06:873,
 06:883, 50:864, 50:870, 50:873,
 50:892, 50:898, 50:901, 50:919,
 50:925, 50:929, 50:943, 50:950,
 50:1312, 50:1313, 50:1316, 50:1317,
 50:1352, 50:1353, 50:1355, 50:1382,
 50:1445, 50:1446, 50:1449, 50:1450,
 50:1485, 50:1486, 50:1488, 50:1514,
 50:1574, 50:1575, 50:1577, 50:1604,
 54:1137, 54:1140, 54:1154, 54:1157,
 54:1174, 54:1177, 57:229, 57:231,
 57:235, 57:541, 57:543, 57:547,
 57:627, 57:632, 57:673, 57:774, 572
 \reserved@c . 01:107, 01:112, 20:775,
 24:97, 24:98, 24:703, 24:706, 27:48,
 27:55, 27:61, 27:68, 28:34, 28:38,
 28:540, 28:541, 28:615, 28:616,
 28:689, 28:690, 28:770, 28:771,
 28:793, 28:802, 28:817, 28:831,
 28:1072, 28:1089, 28:1098, 28:1126,
 28:1137, 28:1195, 28:1208, 28:1210,
 29:205, 29:208, 29:218, 29:219,
 29:222, 29:223, 29:268, 29:278,
 29:280, 32:68, 32:70, 32:77, 06:878,
 06:881, 06:883, 06:886, 50:944,
 50:946, 50:947, 50:1297, 50:1302,
 50:1303, 50:1321, 50:1329, 50:1335,
 50:1357, 50:1365, 50:1428, 50:1433,
 50:1434, 50:1454, 50:1462, 50:1468,
 50:1490, 50:1497, 50:1545, 50:1546,
 50:1547, 50:1557, 50:1579, 50:1586,
 50:1614, 57:233, 57:238, 57:246,
 57:545, 57:566, 57:567, 57:568,
 57:570, 57:571, 57:572, 57:573,
 57:574, 57:582, 57:590, 57:776, 1399
 \reserved@d 01:110, 01:113, 20:773, 20:775,
 05:51, 05:56, 27:61, 27:68, 27:70,
 27:74, 28:1080, 28:1089, 28:1098,
 28:1134, 28:1137, 28:1203, 28:1208,
 28:1212, 29:210, 29:211, 29:216,
 29:217, 29:271, 29:272, 29:276,
 29:277, 06:871, 06:880, 57:777, 1415
 \reserved@e 18:58, 18:60, 18:70, 18:72, 18:101,
 18:108, 18:116, 27:39, 27:45, 27:70,
 27:73, 27:74, 28:35, 28:40, 57:778, 1392
 \reserved@f 18:59, 18:60, 18:71, 18:72, 18:116,
 21:1571, 21:1572, 21:1573, 21:1574,
 21:1576, 21:1587, 24:253, 24:255,
 24:261, 24:262, 26:399, 26:410,
 26:414, 26:418, 26:424, 26:427,
 26:466, 26:503, 26:506, 27:27, 27:38,
 27:45, 27:71, 27:73, 57:779, 1385
 \reset@font 29:312,
 29:764, 29:810, 29:826, 29:841,
 29:856, 35:17, 35:34, 35:51, 40:545,
 40:563, 40:580, 45:175, 45:371,
 45:528, 45:547, 45:565, 47:40,
 49:14, 54:901, 54:971, 54:1030, 1361
 \restglb@settings 26:269, 26:279
 \restore@mathversion 28:108, 28:111, 28:126, 28:134
 \restore@protect 06:357
 \rlh@ 30:475, 30:476
 \rm@def@ult 29:476
 \rmdef@ult 29:337, 29:360, 29:398, 29:418,
 29:434, 29:445, 29:477, 29:518,
 29:779, 29:780, 33:27, 33:48, 746
 \robust@command@act 09:106, 06:514, 06:515,
 06:517, 06:585, 06:642, 06:779, 1417
 \robust@command@act@chk@args 06:539, 06:560, 99
 \robust@command@act@do 06:525, 06:557, 99
 \robust@command@act@end . . 06:522,
 06:523, 06:535, 06:538, 06:558, 99

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\robust@command@act@loop</code>	<code>\set@display@protect</code>
..... 06:519, 06:525, 06:555, 99	 14:7, 14:14, 14:34,
<code>\robust@command@act@loop@aux</code> ...	14:61, 06:8, 06:16, 06:355, 50:424, 1383
..... 06:525, 06:556	<code>\set@fontsize</code>
<code>\robust@command@chk@safe</code>	24:380,
..... 09:153, 06:407, 06:518,	24:382, 26:122, 26:168, 26:178, 1375
06:539, 06:559, 06:699, 06:711, 105	<code>\set@mathaccent</code>
<code>\sfct@</code>	28:885, 28:893, 28:928, 28:936, 28:954
26:443, 26:507	<code>\set@mathchar</code>
<code>\sfct@alias</code>	28:1009, 28:1019
26:569	<code>\set@mathdelimiter</code>
<code>\sfct@fixed</code>	28:1086, 28:1095, 28:1147
26:582	<code>\set@mathradical</code>
<code>\sfct@gen</code>	28:467, 28:1205
26:519	<code>\set@mathsymbol</code> 28:993, 28:1001, 28:1022
<code>\sfct@genb</code>	<code>\set@simple@size@args</code>
26:524	26:365, 26:378, 26:385, 26:406, 26:420
<code>\sfct@sgen</code>	<code>\set@size@funct@args</code>
26:519	26:368, 26:370, 26:428
<code>\sfct@sgenb</code>	<code>\set@size@funct@args@</code>
26:524	26:428
<code>\sfct@sub</code>	<code>\set@target@series</code> 24:351, 24:363,
26:531	25:2825, 25:2829, 25:2832, 25:2835,
<code>\sfct@subf</code>	25:2858, 25:2860, 25:2869, 25:2904
26:574	<code>\set@typeset@protect</code> 06:355,
<code>\saved@reqcolroom</code>	06:374, 41:249, 41:272, 41:312,
.. 54:1454, 54:1911, 54:2600, 54:2728	53:145, 53:189, 54:883, 54:885,
<code>\saved@space@catcode</code> . 57:404, 57:473	54:958, 54:960, 54:1016, 54:1018, 1397
<code>\scan@fontshape</code> ... 27:7, 27:40, 27:43	<code>\SetMathAlphabet@</code> 28:705, 28:774, 28:783
<code>\scan@fontshape</code> ... 27:6, 27:26, 27:37	<code>\SetSymbolFont@</code> 28:565, 28:619, 28:637
<code>\scantokens</code>	<code>\SetSymbolFont@m@dropped</code>
332	28:607, 28:612, 28:632, 28:633
<code>\scriptfont@name</code> 26:334, 26:339	<code>\sf@def@ult</code>
<code>\section</code>	29:476
330	<code>\sf@size</code>
<code>\select@group</code> 24:686, 24:705, 28:49,	19:6,
28:407, 28:457, 28:520, 28:698,	21:322, 24:287, 24:306, 24:725,
28:751, 28:760, 28:798, 28:830, 707	26:329, 26:333, 33:636, 45:413,
<code>\series@change@debug</code>	45:441, 45:447, 45:472, 45:500, 45:506
29:138, 29:167, 29:170,	<code>\sfdef@ult</code> .. 29:338, 29:361, 29:399,
29:180, 29:183, 29:187, 29:199,	29:419, 29:435, 29:446, 29:478,
29:207, 29:212, 29:218, 29:221, 29:223	29:519, 29:783, 33:29, 33:50, 746
<code>\series@check@toks</code>	<code>\sh@ft</code>
25:2878, 25:2880, 25:2887	02:414, 1405
<code>\series@drop@one@m</code>	<code>\show@kernel@robust@command</code>
25:2884, 25:2888, 25:2906	 06:705, 06:768, 06:796
<code>\series@maybe@drop@one@m</code> .. 24:32,	<code>\show@release@info</code>
25:2871, 25:2873, 25:2905, 28:534,	03:38, 03:41, 03:46, 03:51, 37:47,
28:609, 29:190, 29:209, 29:215,	37:48, 37:50, 37:112, 37:113, 37:115, 36
29:255, 29:270, 29:275, 29:480, 29:481	<code>\ShowCommand</code>
<code>\series@maybe@drop@one@m@x</code>	337
25:2874, 25:2876	<code>\sixt@on</code> 04:30, 24:15, 28:85, 28:249,
<code>\seriesdefault@kernel</code>	28:871, 28:873, 28:919, 28:921,
29:313, 29:930, 765	28:980, 28:982, 28:1028, 28:1030,
<code>\set@mathdelimiter</code> . 28:1135, 28:1169	28:1068, 28:1070, 28:1076, 28:1078,
<code>\set@color</code>	28:1122, 28:1124, 28:1130, 28:1132,
40:109	28:1191, 28:1193, 28:1199, 28:1201,
<code>\set@curr@file</code> ... 20:245, 20:254,	42:289, 42:304, 42:306, 45:62, 45:80,
20:281, 20:289, 20:471, 20:489,	45:131, 45:153, 02:16, 02:60, 02:62,
50:975, 50:1295, 50:1426, 52:266, 1124	54:1356, 54:1402, 54:1542, 54:1687,
<code>\set@curr@file@aux</code>	54:1854, 54:2169, 54:2233, 54:2290,
52:272, 52:278, 52:279	
<code>\set@curr@file@nosearch</code> . 52:266, 1425	
<code>\set@current@meta@family</code> . 28:412,	
29:773, 29:778, 29:812, 29:867, 1439	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 54:2360, 54:2630, 54:2639, 54:2695,
54:2711, 54:2749, 54:2799, 01:55
`\sixt@n` [421](#)
`\size@update`
26:147, 26:173, 26:186, 26:205, [26:207](#)
`\sizefn@info`
26:369, 26:371, 26:379, 26:407, 26:421
`\skip@` 17:79,
17:85, 17:98, 17:104, 32:106, 32:109,
[02:41](#), 02:376, 02:378, 02:379, 02:381
`\space@skip` 18:84, 18:539, 18:540
`\sp@n` [41:456](#)
`\split@name` [24:454](#),
24:466, 24:580, 24:595, 26:536, 26:550
`\ssf@size` .. 21:348, 21:380, 21:1282,
24:288, 24:307, 24:726, 26:329, 26:335
`\string@makeletter` [06:898](#), 50:949,
50:984, 50:985, 50:986, 52:169, [1418](#)
`\strip@meaning` [1368](#)
`\strip@prefix`
... [01:95](#), [24:683](#), 01:212, 06:332,
06:394, 06:432, 06:460, 01:307, 06:895
`\strip@pt` ... 24:279, 24:285, 24:286,
24:287, 24:288, 24:301, 24:305,
24:392, 24:725, 24:726, 26:181, 02:418
`\strong@currfont` 29:686, 29:695
`\strongem@force`
..... 29:693, 29:698, 29:702, 29:703
`\strongfontdeclare@clist`
29:681, 29:683, 29:687, 29:691, 29:696
`\sub@sfcnt` 26:531, 26:532, 26:533, 26:560
`\subf@sfcnt` 26:574, 26:575, 26:576
`\subst@correction` 24:86, 24:92
`\subst@fontshape` 27:8, [27:80](#)
`\subst@size` [26:482](#)
`\sw@slant` [32:92](#), [32:102](#)
`\t@st@ic` [32:91](#), [32:95](#), [1363](#)
`\target@meta@family@value`
.. 29:243, 29:262, 29:267, 29:269, [1439](#)
`\target@series@value`
..... 29:172, 29:179, 29:182,
29:184, 29:188, 29:189, 29:190,
29:213, 29:219, 29:220, 29:222,
29:242, 29:250, 29:252, 29:255,
29:273, 29:278, 29:279, 29:280, [1415](#)
`\tc@check@accent`
... [33:73](#), 33:125, 33:127, 33:129,
33:131, 33:133, 33:135, 33:137,
33:139, 33:141, 33:143, 33:145,
33:147, 33:149, 33:151, 33:153,
33:155, [33:899](#), 33:975, 33:977, 33:979
`\tc@check@symbol` ... [33:73](#), 33:175,
33:177, 33:179, 33:181, 33:183,
33:185, 33:187, 33:189, 33:191,
33:193, 33:195, 33:197, 33:199,
33:201, 33:203, 33:205, 33:207,
33:209, 33:211, 33:213, 33:215,
33:217, 33:219, 33:221, 33:223,
33:225, 33:227, 33:229, 33:231,
33:233, 33:235, 33:237, 33:239,
33:241, 33:243, 33:245, 33:247,
33:249, 33:251, 33:253, 33:255,
33:257, 33:259, 33:261, 33:263,
33:265, 33:267, 33:269, 33:271,
33:274, 33:276, 33:278, 33:280,
33:282, 33:284, 33:286, 33:288,
33:290, 33:292, 33:294, 33:296,
33:298, 33:300, 33:302, 33:304,
33:306, 33:308, 33:310, 33:312,
33:314, 33:318, 33:320, 33:322,
33:324, 33:326, 33:328, 33:330,
[33:899](#), 33:969, 33:971, 33:973,
33:981, 33:983, 33:985, 33:987,
33:989, 33:991, 33:993, 33:995,
33:997, 33:999, 33:1001, 33:1003,
33:1005, 33:1007, 33:1009, 33:1011,
33:1013, 33:1015, 33:1017, 33:1019,
33:1021, 33:1023, 33:1025, 33:1027,
33:1029, 33:1031, 33:1033, 33:1035,
33:1037, 33:1039, 33:1041, 33:1043,
33:1045, 33:1047, 33:1049, 33:1051,
33:1053, 33:1055, 33:1057, 33:1059,
33:1061, 33:1063, 33:1065, 33:1067,
33:1069, 33:1071, 33:1073, 33:1075,
33:1077, 33:1079, 33:1081, 33:1083
`\tc@error` 33:74, [33:879](#), 33:900
`\tc@errorwarn` 33:21,
[33:40](#), 33:42, 33:787, 33:789, 33:791,
33:792, 33:838, 33:839, 33:840, 33:873
`\tc@fake@euro`
..... [33:61](#), 33:316, [33:887](#), 33:968
`\tc@forcedfalse` 33:832
`\tc@forcedtrue` 33:837
`\tc@oldstylesubst` 33:16, 33:20
`\tc@subst` .. [33:41](#), 33:73, [33:872](#), 33:899
`\tc@swap@accent` 33:76, 33:77
`\test@font@series@context`
..... 29:584, [29:596](#), 29:618, [754](#)
`\test@next` [1391](#)
`\text@command` 32:8, [32:47](#), [1380](#)
`\textfont@name` 26:332, 26:338
`\textsubscript@offset` [45:463](#)
`\textsubscript@space` [45:463](#)
`\textsuperscript@offset` [45:404](#)
`\textsuperscript@space` [45:404](#)
`\tf@size` 24:286,
24:306, 24:724, 26:329, 26:331, [1361](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

<code>\thr@@</code>	26:59, 26:255, 26:261, 26:274, 26:281, 26:288, 26:293, 38:438, 38:467, 38:632, 39:263, 39:274, 42:298, 42:299, 42:301, 42:302, 42:340, 42:366, 42:399, 42:422, 02:16, 54:855, 57:84, 57:92, 02:447, 02:481, 1389
<code>\toks@</code>	03:94, 03:106, 03:108, 03:115, 03:119, 03:122, 03:127, 18:477, 18:478, 18:483, 24:149, 24:153, 24:155, 24:158, 24:284, 24:289, 28:7, 28:8, 28:506, 28:510, 28:516, 28:519, 28:524, 28:590, 28:591, 28:593, 28:594, 28:644, 28:646, 28:650, 28:667, 28:670, 28:729, 28:741, 28:742, 28:743, 28:789, 28:791, 28:797, 28:805, 28:809, 28:821, 28:824, 28:827, 28:835, 28:837, 28:1258, 28:1260, 28:1262, 28:1265, 28:1267, 28:1270, 28:1273, 28:1305, 28:1306, 50:536, 50:537, 50:539, 50:540, 06:1012, 06:1013, 02:41, 54:2913, 54:2914, 54:2915, 54:2916, 660
<code>\topmark</code>	1065
<code>\topmark(s)</code>	1065
<code>\transform@scriptfont</code>	26:333, 26:335, 26:343, 1439
<code>\try@load@font@shape</code>	1400
<code>\try@load@fontshape</code>	24:469, 24:477, 24:523, 24:628, 25:2867, 26:537, 28:379, 28:396, 28:429, 28:446, 659
<code>\try@simple@size</code>	26:373, 26:498
<code>\try@simples</code>	26:456, 26:462, 26:466
<code>\try@size@range</code>	26:102, 26:373, 26:449
<code>\try@size@substitution</code>	26:104, 26:453
<code>\tryif@simple</code>	26:464, 26:465
<code>\tryis@simple</code>	26:465
<code>\tt@def@ult</code>	29:476
<code>\ttdef@ult</code>	29:339, 29:362, 29:400, 29:420, 29:436, 29:447, 29:479, 29:520, 29:786, 33:31, 33:52, 746
<code>\tw@</code>	02:16, 1380
<code>\two@digits</code>	03:11, 01:169, 01:170, 26:529, 06:2, 50:1248, 50:1341, 50:1474, 50:1563, 01:70
<code>\type@restoreinfo</code>	26:203, 26:208
<code>\unconditionally@reset@math@version</code>	24:405, 24:413
<code>\undeclare@...</code>	1210
<code>\undeclare@file@substitution</code>	52:247, 1164
<code>\underline@text</code>	40:615
<code>\underline@text@kern</code>	40:615
<code>\underline@text@thickness</code>	40:615
<code>\underline@text@vgap</code>	40:615
<code>\unexpandable@noexpand</code>	1394
<code>\unexpanded</code>	1060
<code>\unqu@tefilef@und</code>	52:143
<code>\unquote@name</code>	20:475, 20:481, 20:498, 52:349
<code>\unrestored@protected@xdef</code>	06:357, 45:520, 45:588, 45:604, 45:615, 48:253, 49:41, 49:67, 49:90, 49:123
<code>\unskip</code>	1071
<code>\update@series@target@value</code>	29:119, 29:174, 29:195, 29:245, 29:260, 29:291
<code>\update@uc@cl@with@cyrillic</code>	21:1533, 21:1561, 21:1595, 21:1603
<code>\upper@bound</code>	26:400, 26:401, 26:402, 26:415
<code>\use@mathgroup</code>	24:692, 24:710, 24:712, 26:300, 28:64, 28:93, 28:711, 28:813, 28:816, 28:1283, 28:1307, 1358
<code>\UTF@two@octets@noexpand</code>	1425
<code>\UTFviii@four@octets</code>	57:459, 57:464, 57:470
<code>\UTFviii@four@octets@@</code>	57:459, 57:470
<code>\UTFviii@four@octets@combine</code>	57:494
<code>\UTFviii@four@octets@noexpand</code>	57:500
<code>\UTFviii@four@octets@string</code>	57:497
<code>\UTFviii@invalid</code>	57:398, 57:491
<code>\UTFviii@invalid@err</code>	57:456, 57:461, 57:467
<code>\UTFviii@invalid@err@@</code>	57:456, 57:467
<code>\UTFviii@three@octets</code>	57:458, 57:463, 57:469
<code>\UTFviii@three@octets@@</code>	57:458, 57:469
<code>\UTFviii@three@octets@combine</code>	57:493
<code>\UTFviii@three@octets@noexpand</code>	57:499
<code>\UTFviii@three@octets@string</code>	57:496
<code>\UTFviii@two@octets</code>	57:457, 57:462, 57:468
<code>\UTFviii@two@octets@@</code>	57:457, 57:468
<code>\UTFviii@two@octets@combine</code>	57:492
<code>\UTFviii@two@octets@noexpand</code>	57:498
<code>\UTFviii@two@octets@string</code>	57:495
<code>\UTFviii@undefined@err</code>	57:455, 57:460, 57:466
<code>\UTFviii@undefined@err@@</code>	57:455, 57:466
<code>\v@false</code>	38:82
<code>\v@true</code>	38:81, 38:83
<code>\vbox</code>	1071
<code>\vcenter@text</code>	40:330, 40:380, 41:235
<code>\vcenter@text@auxi</code>	40:330
<code>\vcenter@text@auxii</code>	40:330
<code>\vcenter@text@axis</code>	40:330

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\ver@...` 507
- `\ver@...` 507
- `\ver@{file}.<ext>` 1123
- `\verb@balance@group` 37:656,
37:658, 37:673, 37:675, 37:687,
37:689, 37:702, 37:704, 37:710, 37:711
- `\verb@egroup` 37:656,
37:659, 37:673, 37:676, 37:687,
37:690, 37:702, 37:705, 37:711, 37:715
- `\verb@eol@error` 37:712, 37:724, 37:734
- `\verbatim@font` 37:544,
37:565, 37:573, 37:725, 37:735, 1363
- `\verbatim@nolig@list` . 37:740, 37:746
- `\verbatim@out` 1399
- `\version@elt` 28:19,
28:32, 28:33, 28:503, 28:504, 28:563,
28:593, 28:704, 28:742, 28:834, 28:1263
- `\version@list`
..... 28:17, 28:22, 28:33, 28:496,
28:504, 28:568, 28:599, 28:638,
28:709, 28:754, 28:784, 28:839, 28:1276
- `\vgl@` 02:376, 02:377
- `\voidb@x` .. 23:23, 23:29, 02:181, 02:407
- `\vsplit` 1070
- `\warn@rel@i` 27:5, 27:25, 27:29, 27:81,
27:85, 27:90, 27:95, 27:119, 27:140
- `\wrong@fontshape` . 24:473, 24:610, 1357
- `\x@protect` .. 06:335, 06:346, 06:397,
06:435, 06:463, 06:669, 06:689, 1383
- `\xe@alloc@` 57:56, 57:66
- `\xe@alloc@intercharclass` . 57:35, 1407
- `\xe@ch@ck` 57:57, 57:61
- `\XXX@argdef` 1370
- `\z@` 02:181, 421
- `\z@skip` 02:181, 471
- `\zap@space` .. 20:281, 20:301, 29:629,
29:681, 47:49, 50:266, 50:488,
50:511, 50:523, 50:690, 50:814,
50:854, 50:872, 50:883, 50:900,
50:910, 50:927, 50:959, 50:1261, 50:1396
- `\zref@labelbyprop` 855
- tex commands:
 - `\tex_afterassignment:D`
..... 53:110, 53:199, 07:2798
 - `\tex_aftergroup:D` 53:118, 53:205, 1195
 - `\tex_alignmark:D` 09:388
 - `\tex_currentgrouplevel:D`
.. 10:33, 53:109, 53:117, 53:198, 53:204
 - `\tex_deadcycles:D` 53:152
 - `\tex_endlinechar:D` 09:481, 07:1615,
07:2102, 07:2204, 07:2205, 07:2215,
07:2216, 07:2223, 07:2245, 07:2260, 186
 - `\tex_escapechar:D`
... 09:218, 09:293, 48:119, 48:120,
 - 48:122, 52:212, 52:226, 07:1137,
07:1148, 07:1356, 07:1366, 07:1614,
07:2101, 07:2672, 07:2698, 08:1186, 201
 - `\tex_everypar:D`
.. 16:20, 16:24, 16:29, 16:53, 16:57,
16:61, 16:76, 16:76, 16:118, 16:120,
16:130, 16:131, 16:182, 16:183, 449
 - `\tex_gdef:D` 09:208, 09:283, 342
 - `\tex_hskip:D` 16:104
 - `\tex_ignoreprimitiverror:D`
..... 48:101, 48:104, 48:105, 48:108
 - `\tex_immediate:D` 53:529
 - `\tex_indent:D` 16:125
 - `\tex_interactionmode:D`
..... 48:117, 48:118, 48:123
 - `\tex_lastnodetype:D` 16:103, 48:81, 454
 - `\tex_lastxpos:D` 36:266
 - `\tex_lastypos:D` 36:267
 - `\tex_lowercase:D` 07:2495, 07:2678
 - `\tex_marks:D` 48:261
 - `\tex_newlinechar:D` 09:482
 - `\tex_noindent:D` 16:27, 16:59, 16:135, 449
 - `\tex_par:D`
.... 16:18, 16:51, 16:106, 16:113,
16:137, 16:178, 16:179, 16:180, 452
 - `\tex_parskip:D` ... 16:25, 16:26, 16:58
 - `\tex_savepos:D` 850
 - `\tex_setbox:D` .. 48:124, 53:111, 53:200
 - `\tex_shipout:D` . 53:187, 53:433, 53:584
 - `\tex_splitbotmarks:D`
..... 48:144, 48:175, 1074
 - `\tex_splitfirstmarks:D`
..... 48:156, 48:194, 48:203, 1074
 - `\tex_splitmaxdepth:D` 48:69
 - `\tex_unskip:D` 16:99, 48:79, 453
 - `\tex_vbadness:D` 48:70
 - `\tex_vfuzz:D` 48:71
 - `\tex_vsplit:D` 48:124, 1074
 - `\tex_vss:D` 53:39, 53:59
 - `\tex_write:D` 53:529
 - `\tex_xdef:D` 09:214, 09:289, 342
 - `\text` 1360
 - text commands:
 - `\l_text_case_exclude_arg_tl` . 57:676
 - `\text_case_switch:nmmn` 57:679
 - `\text_declare_case_equivalent:Nn`
..... 57:681
 - `\text_declare_lowercase_exclusion:n`
..... 57:706
 - `\text_declare_lowercase_mapping:nm`
..... 57:685
 - `\text_declare_lowercase_mapping:nmm`
..... 57:686

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\text_declare_titlecase_exclusion:n</code>	<code>\textblank</code>
..... 57:711	... 21:885, 21:1257, 33:309, 33:310, 33:749, 33:984, 33:985, 33:1235
<code>\text_declare_titlecase_mapping:nn</code>	<code>\textborn</code>
..... 57:692	21:929, 33:220, 33:221, 33:351, 33:697, 33:1020, 33:1021, 33:1302
<code>\text_declare_titlecase_mapping:nnn</code>	<code>\textbraceleft</code>
..... 57:693	21:291, 21:328, 21:585, 21:745, 21:1118, 1382
<code>\text_declare_uppercase_exclusion:n</code>	<code>\textbraceright</code>
..... 57:716	21:292, 21:329, 21:586, 21:746, 21:1120, 1381
<code>\text_declare_uppercase_mapping:nn</code>	<code>\textbrokenbar</code>
..... 57:699	21:975, 21:1127, 33:92, 33:615, 33:944, 33:1356
<code>\text_declare_uppercase_mapping:nnn</code>	<code>\textbullet</code>
..... 57:700	.. 21:293, 21:747, 21:945, 21:1207, 33:85, 33:543, 33:607, 33:937, 33:1351
<code>\text_lowercase:n</code>	<code>\textcapitalcompwordmark</code>
..... 1425	21:877, 33:117, 33:648, 33:914, 33:1343
<code>\text_titlecase_all:nn</code>	<code>\textcelsius</code>
..... 57:658	... 21:946, 21:1226, 33:93, 33:317, 33:318, 33:616, 33:755, 33:938, 33:1229
<code>\text_titlecase_first:nn</code>	<code>\textcent</code>
..... 57:659	21:971, 21:1123, 33:94, 33:617, 33:941, 33:1353
<code>\textacutedbl</code>	<code>\textcentoldstyle</code>
..... 21:939, 21:1181, 33:198, 33:199, 33:685, 33:932, 33:1313	21:948, 33:222, 33:223, 33:356, 33:359, 33:698, 33:1036, 33:1037, 33:1316, 809
<code>\textascendercompwordmark</code>	<code>\textcircled</code>
..... 21:878, 33:118, 33:649, 33:915, 33:1344	... 21:299, 21:303, 21:320, 21:321, 21:754, 21:917, 33:119, 33:120, 33:619, 33:635, 33:651, 33:819, 33:1084, 33:1086, 33:1337, 833
<code>\textasciicircum</code>	<code>\textcircledP</code>
..... 21:989, 21:1142, 33:200, 33:201, 33:686, 33:956, 33:1332	.. 21:982, 21:1228, 33:224, 33:225, 33:699, 33:1078, 33:1079, 33:1331
<code>\textasciibreve</code>	<code>\textcolonmonetary</code>
..... 21:937, 21:1180, 33:202, 33:203, 33:687, 33:929, 33:1310	.. 21:950, 21:1219, 33:277, 33:278, 33:729, 33:1038, 33:1039, 33:1259
<code>\textasciicaron</code>	<code>\textcommaabove</code>
..... 21:938, 21:1179, 33:204, 33:205, 33:688, 33:930, 33:1311	... 21:373, 21:375, 21:389, 21:390, 21:480, 21:481, 21:723, 21:724, 1407
<code>\textasciicircum</code>	<code>\textcommabelow</code>
..... 21:306, 21:581, 21:1116, 1394	... 21:344, 21:346, 21:352, 21:353, 21:726, 21:727, 21:728, 21:729, 21:730, 21:731, 21:732, 21:733, 21:734, 21:735, 21:1280, 21:1483, 21:1484, 21:1485, 21:1486, 1407
<code>\textasciidieresis</code>	<code>\textcompsubstdefault</code>
.. 21:977, 21:1129, 33:206, 33:207, 33:689, 33:946, 33:1329	 33:34, 33:39, 33:55, 33:582, 33:877, 1414
<code>\textasciigrave</code>	<code>\textcompwordmark</code>
..... 21:928, 21:1110, 33:208, 33:209, 33:690, 33:927, 33:1301	.. 21:310, 21:311, 21:587, 21:1186, 523
<code>\textasciimacron</code>	<code>\textcopyleft</code>
... 21:984, 21:1137, 33:210, 33:211, 33:691, 33:951, 33:1312	... 21:980, 33:226, 33:227, 33:332, 33:700, 33:1076, 33:1077, 33:1330
<code>\textasciitilde</code>	<code>\textcopyright</code>
..... 21:307, 21:582, 21:1121, 1394	.. 21:303, 21:337, 21:978, 21:1130, 33:95, 33:618, 33:947, 33:1358, 1399
<code>\textasteriskcentered</code>	<code>\textcurrency</code>
... 21:287, 21:741, 21:888, 21:889, 21:1243, 22:331, 22:337, 33:84, 33:541, 33:606, 33:922, 33:1347, 1398	21:973, 21:1125, 33:299, 33:300, 33:742,
<code>\textbackslash</code>	
..... 21:288, 21:583, 21:742, 21:1115, 1394	
<code>\textbaht</code>	
21:963, 21:1184, 33:216, 33:217, 33:695, 33:1062, 33:1063, 33:1323	
<code>\textbar</code>	
..... 21:289, 21:584, 21:743, 21:1119, 1381	
<code>\textbardbl</code>	
..... 21:290, 21:744, 21:943, 21:1198, 22:336, 33:91, 33:329, 33:330, 33:542, 33:614, 33:761, 33:935, 33:1228, 1403	
<code>\textbf</code>	
..... 32:19, 36:261, 739	
<code>\textbigcircle</code>	
..... 21:753, 21:916, 21:1260, 33:218, 33:219, 33:696, 33:1014, 33:1015, 33:1298, 1404	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 33:814, 33:818, 33:972, 33:973, 33:1248
- `\textdagger`
 ... 21:295, 21:332, 21:749, 21:941,
 21:1205, 22:332, 22:338, 33:87,
 33:545, 33:609, 33:933, 33:1348, 1382
- `\textdaggerdbl`
 ... 21:294, 21:333, 21:748, 21:942,
 21:1206, 22:333, 22:339, 33:86,
 33:544, 33:608, 33:934, 33:1349, 1382
- `\textdblhyphen` . 21:900, 33:230, 33:231,
 33:333, 33:702, 33:986, 33:987, 33:1286
- `\textdblhyphenchar`
 ... 21:936, 33:228, 33:229, 33:334,
 33:701, 33:1032, 33:1033, 33:1309
- `\textdegree` 21:985,
 21:1138, 33:96, 33:620, 33:952, 33:1362
- `\textdied` 21:931, 33:232, 33:233, 33:352,
 33:703, 33:1024, 33:1025, 33:1304
- `\textdiscount`
 .. 21:965, 21:1218, 33:234, 33:235,
 33:704, 33:1066, 33:1067, 33:1324
- `\textdiv` 21:1002,
 21:1163, 33:97, 33:621, 33:966, 33:1371
- `\textdivorced`
 .. 21:930, 21:1263, 33:236, 33:237,
 33:705, 33:1022, 33:1023, 33:1303
- `\textdollar` 21:275, 21:327, 21:459,
 21:588, 21:822, 21:886, 21:1112,
 33:78, 33:79, 33:587, 33:589,
 33:920, 33:1092, 33:1094, 33:1345, 1381
- `\textdollaroldstyle` 21:947,
 33:238, 33:239, 33:357, 33:358,
 33:706, 33:1034, 33:1035, 33:1315, 809
- `\textdong` 21:959, 21:1223, 33:279, 33:280,
 33:730, 33:1056, 33:1057, 33:1262
- `\textdownarrow`
 .. 21:927, 21:1238, 33:281, 33:282,
 33:731, 33:1018, 33:1019, 33:1258
- `\texteightoldstyle`
 ... 21:910, 33:178, 33:179, 33:348,
 33:674, 33:1004, 33:1005, 33:1295
- `\textellipsis` 21:316, 21:341, 21:1208
- `\textemdash` 21:276, 21:428, 21:432,
 21:589, 21:593, 21:809, 21:1190, 1381
- `\textendash` 21:277, 21:429, 21:431,
 21:590, 21:592, 21:810, 21:1189, 1381
- `\textestimated`
 .. 21:966, 21:1234, 33:293, 33:294,
 33:738, 33:817, 33:970, 33:971, 33:1252
- `\texteuro`
 . 21:1000, 21:1224, 33:315, 33:316,
 33:753, 33:815, 33:967, 33:968, 33:1238
- `\textexclamdown` 21:278, 21:433,
 21:435, 21:594, 21:811, 21:1122, 1381
- `\textfiguredash`
 21:431, 21:592, 21:1188, 21:1194, 1420
- `\textfiveoldstyle`
 21:907, 33:180, 33:181,
 33:345, 33:675, 33:998, 33:999, 33:1292
- `\textfloatsep` 54:1079, 54:1092, 54:2730,
 54:2780, 54:2830, 54:3004, 1234
- `\textflorin` 21:949, 21:1177,
 33:297, 33:298, 33:741, 33:939, 33:1247
- `\textfont` 26:338, 38:251, 40:353,
 40:649, 40:650, 40:651, 45:428,
 45:429, 45:431, 45:487, 45:488, 45:490
- `\textfouroldstyle`
 21:906, 33:182, 33:183,
 33:344, 33:676, 33:996, 33:997, 33:1291
- `\textfraction` 54:2527, 54:2530, 54:2555,
 54:2558, 54:2717, 54:2998, 1400
- `\textfractionsolidus`
 .. 21:901, 21:1215, 33:301, 33:302,
 33:744, 33:923, 33:1241, 33:1473, 1400
- `\textgravedbl` 21:940, 21:1182,
 33:212, 33:213, 33:692, 33:931, 33:1314
- `\textgreater`
 21:301, 21:595, 21:764, 21:1114, 1390
- `\textguarani`
 ... 21:953, 33:240, 33:241, 33:355,
 33:707, 33:1044, 33:1045, 33:1318
- `\textheight` 20:22,
 20:23, 20:92, 20:93, 20:149, 20:150,
 45:257, 45:258, 45:261, 45:287,
 45:301, 53:433, 54:54, 54:230,
 54:231, 54:279, 54:404, 54:452,
 54:481, 54:942, 54:1000, 54:1059,
 54:1114, 54:1166, 57:156, 57:157, 1372
- `\texthorizontalbar`
 21:432, 21:593, 21:1191, 21:1196, 1420
- `\texthyphen`
 . 21:280, 21:438, 21:597, 21:813, 1381
- `\texthyphenchar`
 . 21:279, 21:437, 21:596, 21:812, 1387
- `\textindent` 1388
- `\textinterrobang`
 .. 21:957, 21:1214, 33:313, 33:314,
 33:751, 33:1052, 33:1053, 33:1236
- `\textinterrobangdown`
 .. 21:958, 21:1264, 33:311, 33:312,
 33:750, 33:1054, 33:1055, 33:1237
- `\textit` 32:21
- `\textlangle`
 .. 21:912, 21:1255, 33:273, 33:274,
 33:726, 33:1008, 33:1009, 33:1265
- `\textlbrace` 1381
- `\textlbrackdbl` 21:924, 33:174,
 33:175, 33:353, 33:671, 33:925, 33:1299

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\textleaf` 21:932, 33:242, 33:243, 33:337,
33:708, 33:1026, 33:1027, 33:1305
- `\textleftarrow` 21:883, 21:1235, 33:283,
33:284, 33:732, 33:980, 33:981, 33:1255
- `\textlegacyasteriskcentered`
..... 33:553, 33:764
- `\textlegacybardbl` 33:553, 33:765
- `\textlegacybullet` 33:553, 33:766
- `\textlegacydagger` 33:553, 33:768
- `\textlegacydaggerdbl` 33:553, 33:767
- `\textlegacyparagraph` 33:553, 33:769
- `\textlegacyperiodcentered` 33:553, 33:770
- `\textlegacysection` 33:553, 33:771
- `\textless`
..... 21:300, 21:598, 21:763, 21:1113, 1387
- `\textlira` 21:955, 21:1220, 33:285, 33:286,
33:733, 33:1048, 33:1049, 33:1261
- `\textlnot` 21:981,
21:1135, 33:98, 33:622, 33:949, 33:1360
- `\textlquill`
.. 21:969, 21:1216, 33:244, 33:245,
33:709, 33:1072, 33:1073, 33:1327
- `\textmacron` 1401
- `\textmarried`
.. 21:933, 21:1262, 33:246, 33:247,
33:710, 33:1028, 33:1029, 33:1306
- `\textmd` 32:19
- `\textmho` 21:915, 21:1233, 33:248, 33:249,
33:711, 33:1012, 33:1013, 33:1297
- `\textminus` ... 21:913, 21:1239, 33:307,
33:308, 33:747, 33:924, 33:1242, 813
- `\textmu` 21:990, 21:1143,
33:305, 33:306, 33:746, 33:957, 33:1244
- `\textmusicalnote`
.. 21:934, 21:1261, 33:250, 33:251,
33:712, 33:1030, 33:1031, 33:1307
- `\textnaira`
.. 21:952, 21:1221, 33:252, 33:253,
33:713, 33:1042, 33:1043, 33:1317
- `\textnineoldstyle`
... 21:911, 33:184, 33:185, 33:349,
33:677, 33:1006, 33:1007, 33:1296
- `\textnonbreakinghyphen`
21:430, 21:591, 21:1187, 21:1192, 1420
- `\textnormal` 32:15
- `\textnumero`
.. 21:964, 21:1227, 33:295, 33:296,
33:739, 33:1064, 33:1065, 33:1251
- `\textogonekcentered`
..... 21:510, 21:721, 21:722, 1404
- `\textohm` 21:923, 21:1232, 33:303,
33:304, 33:745, 33:816, 33:969, 33:1243
- `\textonehalf` 21:998,
21:1152, 33:99, 33:623, 33:963, 33:1368
- `\textoneoldstyle` 21:903, 33:186, 33:187,
33:341, 33:678, 33:990, 33:991, 33:1288
- `\textonequarter` 21:997,
21:1151, 33:100, 33:624, 33:962, 33:1367
- `\textonesuperior`
.. 21:994, 21:1146, 33:101, 33:319,
33:320, 33:625, 33:756, 33:960, 33:1232
- `\textopenbullet`
.. 21:967, 21:1259, 33:254, 33:255,
33:714, 33:1068, 33:1069, 33:1325
- `\textordfeminine` 21:325, 21:979,
21:1131, 33:102, 33:626, 33:948, 33:1359
- `\textordmasculine` ... 21:326, 21:995,
21:1147, 33:103, 33:628, 33:961, 33:1366
- `\TextOrMath`
... 22:328, 22:331, 22:332, 22:333,
22:334, 22:335, 22:336, 22:337,
22:338, 22:339, 22:344, 22:351, 1406
- `\textparagraph` . 21:296, 21:330, 21:750,
21:991, 21:1144, 22:335, 33:88,
33:546, 33:610, 33:958, 33:1364, 1382
- `\textperiodcentered`
.. 21:297, 21:751, 21:992, 21:1145,
33:89, 33:547, 33:611, 33:959, 33:1365
- `\textpermill` 1400
- `\textpertenmill` 1400
- `\textpertenthousand`
.. 21:515, 21:961, 21:1210, 33:270,
33:271, 33:272, 33:723, 33:1058,
33:1059, 33:1096, 33:1321, 1400
- `\textperthousand` 21:513,
21:944, 21:1209, 33:82, 33:83,
33:603, 33:936, 33:1095, 33:1350, 1400
- `\textpeso` 21:954, 21:1225, 33:256, 33:257,
33:715, 33:1046, 33:1047, 33:1319
- `\textpilcrow`
... 21:962, 33:258, 33:259, 33:350,
33:716, 33:1060, 33:1061, 33:1322
- `\textpm` 21:986,
21:1139, 33:104, 33:630, 33:953, 33:1363
- `\textquestiondown` ... 21:281, 21:434,
21:436, 21:599, 21:814, 21:1154, 1381
- `\textquotedbl` 21:602, 21:1111, 1381
- `\textquotedblleft` 21:282,
21:439, 21:600, 21:815, 21:1202, 1381
- `\textquotedblright` 21:283,
21:440, 21:601, 21:816, 21:1203, 1382
- `\textquoteleft` 21:284,
21:441, 21:603, 21:817, 21:1199, 1381
- `\textquoteright` 21:285,
21:442, 21:604, 21:818, 21:1200, 1382
- `\textquotesingle` 21:887,
21:1109, 33:105, 33:631, 33:921, 33:1346

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\textquotestraightbase` 21:879,
33:106, 33:335, 33:632, 33:916, 33:1341
- `\textquotestraightdblbase` 21:880,
33:107, 33:336, 33:633, 33:917, 33:1342
- `\textrangle`
.. 21:914, 21:1256, 33:275, 33:276,
33:727, 33:1010, 33:1011, 33:1266
- `\textrbrace` 1382
- `\textrbrackdbl` 21:925, 33:176,
33:177, 33:354, 33:672, 33:926, 33:1300
- `\textrecipe`
.. 21:956, 21:1229, 33:260, 33:261,
33:717, 33:1050, 33:1051, 33:1320
- `\textreferencemark`
.. 21:993, 21:1213, 33:262, 33:263,
33:718, 33:1080, 33:1081, 33:1333
- `\textregistered`
.. 21:320, 21:321, 21:983, 21:1136,
33:108, 33:634, 33:950, 33:1361, 1394
- `\textrightarrow` 21:884, 21:1237, 33:287,
33:288, 33:734, 33:982, 33:983, 33:1256
- `\textrm` 32:15
- `\textrquill`
.. 21:970, 21:1217, 33:264, 33:265,
33:719, 33:1074, 33:1075, 33:1328
- `\textsc` 32:21, 1399
- `\textsection` 21:298,
21:331, 21:605, 21:752, 21:976,
21:1128, 22:334, 33:90, 33:548,
33:549, 33:612, 33:945, 33:1357, 1382
- `\textservicemark`
.. 21:968, 21:1230, 33:266, 33:267,
33:720, 33:1070, 33:1071, 33:1326
- `\textsevenoldstyle`
.. 21:909, 33:188, 33:189, 33:347,
33:679, 33:1002, 33:1003, 33:1294
- `\textsf` 32:15
- `\textsixoldstyle`
.. 21:908, 33:190, 33:191, 33:346,
33:680, 33:1000, 33:1001, 33:1293
- `\textsl` 32:21
- `\textssc` 25:2924, 32:25
- `\textsterling` .. 21:286, 21:339, 21:466,
21:606, 21:829, 21:972, 21:1124,
33:80, 33:81, 33:588, 33:596,
33:942, 33:1091, 33:1093, 33:1354, 1381
- `\textstyle` 19:15, 30:480, 38:63, 40:351,
40:645, 40:646, 40:647, 45:420,
45:421, 45:423, 45:479, 45:480, 45:482
- `\textsubscript` 45:450
- `\textsuperscript` 21:323, 21:325, 21:326,
33:627, 33:629, 33:643, 45:402, 1388
- `\textsurd` 21:996, 21:1254, 33:268, 33:269,
33:721, 33:1082, 33:1083, 33:1334
- `\textsw` 25:2919, 32:25
- `\TextSymbolUnavailable` 21:3, 21:782, 1394
- `\textthreeoldstyle`
.. 21:905, 33:192, 33:193,
33:343, 33:681, 33:994, 33:995, 33:1290
- `\textthreequarters` 21:999,
21:1153, 33:110, 33:639, 33:964, 33:1369
- `\textthreequartersemdash`
.. 21:882, 33:109, 33:321, 33:322,
33:339, 33:638, 33:757, 33:919, 33:1227
- `\textthreesuperior`
.. 21:988, 21:1141, 33:111, 33:323,
33:324, 33:640, 33:758, 33:955, 33:1231
- `\texttildelow` 21:935, 21:1183,
33:214, 33:215, 33:693, 33:928, 33:1308
- `\texttimes` 21:1001,
21:1157, 33:112, 33:641, 33:965, 33:1370
- `\texttrademark` 21:323, 21:960, 21:1231,
33:113, 33:642, 33:940, 33:1352, 1394
- `\texttt` 32:15
- `\texttwelveudash`
.. 21:881, 33:114, 33:325, 33:326,
33:338, 33:644, 33:759, 33:918, 33:1226
- `\texttwooldstyle` 21:904, 33:194, 33:195,
33:342, 33:682, 33:992, 33:993, 33:1289
- `\texttwosuperior`
.. 21:987, 21:1140, 33:115, 33:327,
33:328, 33:645, 33:760, 33:954, 33:1230
- `\textulc` 25:2914, 32:25
- `\textunderline` 1381
- `\textunderscore`
21:308, 21:335, 21:607, 21:1117, 1390
- `\textup` 32:21, 663
- `\textuparrow`
.. 21:926, 21:1236, 33:289, 33:290,
33:735, 33:1016, 33:1017, 33:1257
- `\textvisiblespace`
.. 21:312, 21:608, 21:1258, 1390
- `\textwidth` 20:24, 20:94, 20:151,
40:507, 45:266, 54:55, 54:123,
54:206, 54:223, 54:920, 54:935,
54:985, 54:995, 54:1044, 54:1054,
54:2884, 54:2927, 54:2956, 57:157, 1372
- `\textwon` 21:951, 21:1222, 33:291, 33:292,
33:736, 33:1040, 33:1041, 33:1260
- `\textyen` 21:974,
21:1126, 33:116, 33:646, 33:943, 33:1355
- `\textzerooldstyle`
.. 21:902, 33:196, 33:197,
33:340, 33:683, 33:988, 33:989, 33:1287
- `\TH` 21:558, 21:1159, 57:720, 1377
- `\th` 21:609, 21:1165, 57:720, 1377
- `\thanks` 1012
- `\thanks` 44:10, 44:26, 1383

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\the</code>	451	07:354, 07:382, 07:450, 07:451,
<code>\the...</code>	1431	07:671, 07:791, 07:804, 07:805,
<code>\the#1</code>	560	07:806, 07:808, 07:990, 07:1058,
<code>thebibliography (env.)</code>	1053	07:1252, 07:1491, 07:1492, 07:1564,
<code>\theenum</code>	922	07:1585, 07:1613, 07:1874, 07:1875,
<code>\theequation</code>		07:1891, 07:2103, 07:2483, 07:2561
38:410, 38:426, 38:455, 38:550, 38:611		
<code>\thefootnote</code> 45:396, 45:583, 45:588, 45:608		<code>\tl_const:Nn</code>
<code>\theH</code>	558	09:11, 09:12, 11:48, 11:49, 11:50, 11:51,
<code>\theHcounter</code>	561	11:52, 11:53, 11:54, 11:55, 48:62,
<code>\thempfn</code>	40:517,	53:372, 53:377, 07:1514, 07:2833,
45:515, 45:520, 45:599, 45:604, 45:607		07:3282, 08:35, 08:36, 08:104,
<code>\thempfootnote</code>	40:517, 45:398	08:107, 08:941, 08:942, 08:944,
<code>\thepage</code>	20:213,	08:945, 08:946, 08:947, 08:950,
20:226, 34:6, 35:18, 35:35, 35:52,		08:951, 08:953, 08:974, 08:975, 08:1319
35:105, 35:120, 35:129, 36:233,		<code>\tl_count:N</code> ... 52:546, 07:400, 07:2507
36:245, 36:250, 36:257, 44:164,		<code>\tl_count:n</code> . 07:602, 07:607, 07:893,
44:171, 44:177, 46:15, 46:32, 47:43,		07:1011, 07:1065, 07:1292, 07:1442
49:14, 54:249, 54:280, 54:2404, 854		<code>\tl_gclear:N</code> 48:163, 48:164, 306
<code>\Theta</code>	30:310	<code>\tl_gclear_new:N</code> . 36:25, 36:167, 08:264
<code>\theta</code>	30:286	<code>\tl_gput_left:Nn</code>
<code>\thetotalpages</code>	53:397, 53:495, 1206	51:265
<code>\thicklines</code>	42:135	<code>\tl_gput_right:Nn</code>
<code>\thickmuskip</code> 30:657, 38:228, 38:230, 38:243		16:69, 16:78, 28:336, 48:188, 48:199,
<code>\thickspace</code>	38:214	07:1167, 07:1484, 07:1486, 08:455, 289
<code>\thinlines</code>	42:135, 42:826, 42:843	<code>\tl_gset:Nn</code> .. 09:140, 16:15, 16:48,
<code>\thinmuskip</code>		17:128, 17:132, 17:152, 17:156,
30:655, 38:220, 38:222, 38:238, 38:244		28:285, 28:333, 36:26, 36:160,
<code>\thinspace</code>	18:558, 18:564,	36:168, 48:143, 48:154, 48:174,
18:565, 38:189, 38:214, 38:251, 1415		48:238, 48:239, 48:240, 52:51,
<code>\thispagestyle</code>	49:6	55:327, 55:328, 55:348, 55:352,
<code>\tilde</code>	18:494, 18:505, 30:530	55:396, 55:404, 55:450, 08:176,
<code>\time</code>	01:163, 01:167, 02:258	08:188, 08:295, 08:313, 08:411,
<code>\times</code>	30:407	08:423, 08:445, 08:489, 08:771,
<code>\title</code>	1012	08:825, 08:859, 08:863, 08:886, 08:890
<code>\title</code>	44:6, 44:7, 44:21, 44:23, 44:31	<code>\tl_gset_eq:NN</code>
<code>title</code>	36:260, 854	48:58, 48:59, 48:60, 48:141, 48:142,
<code>tl commands:</code>		48:147, 48:149, 48:153, 48:216,
<code>\c_empty_tl</code>	08:2242, 08:59	48:218, 48:220, 48:227, 48:229,
<code>\c_noalue_tl</code> ..	07:3281, 07:3282, 258	48:231, 48:469, 48:471, 48:473,
<code>\c_space_tl</code>		48:475, 48:481, 48:485, 48:488,
... 09:398, 11:698, 11:702, 11:743,		55:427, 55:428, 08:57, 08:452, 08:463
11:1110, 11:1113, 16:42, 16:73,		<code>\tl_head:N</code>
16:82, 52:553, 52:556, 07:27, 07:124,		07:2735
07:140, 07:143, 07:157, 07:168,		<code>\tl_head:w</code>
07:169, 07:185, 07:193, 07:203,		11:162, 11:175
07:206, 07:208, 07:210, 07:212,		<code>\tl_if_blank:nTF</code> ... 11:42, 11:422,
07:220, 07:226, 07:294, 07:295,		11:426, 11:599, 11:602, 11:627,
57:653, 57:660, 57:665, 57:670,		11:630, 17:31, 20:540, 57:633,
07:1192, 07:1207, 07:1239, 07:1240,		07:598, 07:1625, 07:1749, 07:1857,
07:1417, 07:1424, 07:1556, 07:2813		07:2317, 07:2354, 07:2560, 07:3304,
<code>\tl_clear:N</code>		07:3305, 07:3306, 08:429, 08:435, 08:477
... 09:212, 09:287, 11:375, 11:414,		<code>\tl_if_blank_p:n</code>
11:897, 11:1017, 11:1252, 07:309,		... 11:1254, 07:120, 07:2557
		<code>\tl_if_empty:N</code>
		313
		<code>\tl_if_empty:NTF</code> . 08:2020, 08:2028,
		08:2225, 08:2227, 11:336, 11:349,
		11:402, 11:1242, 48:145, 48:176,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

48:264, 07:320, 07:321, 07:469,
 07:722, 07:1498, 08:229, 08:249,
 08:360, 08:362, 08:659, 08:1060, 08:1450
 \tl_if_empty:nTF
 08:2551, 09:136, 09:321,
 09:340, 11:517, 11:838, 11:1541,
 11:1546, 49:26, 49:30, 49:45, 49:55,
 52:40, 52:46, 52:58, 52:107, 52:114,
 52:116, 52:118, 52:416, 07:1650,
 07:2314, 07:2673, 07:2699, 07:2734,
 07:2968, 07:3058, 08:332, 08:347,
 08:350, 08:576, 08:846, 08:878, 08:932
 \tl_if_empty_p:N 08:2501,
 08:2502, 53:129, 53:175, 53:430
 \tl_if_empty_p:n 08:909
 \tl_if_eq:NNTF 48:293,
 48:301, 48:308, 48:377, 48:478, 07:343
 \tl_if_eq:NnTF 28:296, 36:173
 \tl_if_eq:nnTF 07:734, 07:1024, 07:2678
 \tl_if_exist:N 307
 \tl_if_exist:NNTF . 08:2270, 08:2290,
 08:2376, 08:2524, 08:2565, 36:106,
 36:111, 36:120, 36:129, 36:134,
 36:152, 36:194, 36:209, 48:375,
 07:1499, 08:129, 08:227, 08:247, 08:1297
 \tl_if_exist_p:N 08:270
 \tl_if_head_eq_charcode:nNTF ...
 20:542, 20:547
 \tl_if_head_eq_meaning:nNTF 07:2809
 \tl_if_head_is_group:nTF
 09:348, 07:2599, 07:2631
 \tl_if_head_is_group_p:n ... 07:2556
 \tl_if_head_is_N_type:nTF 09:345,
 07:2567, 07:2579, 07:2596, 07:2628
 \tl_if_in:NnTF 07:1888
 \tl_if_in:nnTF 11:371,
 11:395, 11:417, 11:1051, 07:11,
 07:614, 07:816, 07:1854, 07:1880, 192
 \tl_if_novalue:n 07:3284
 \tl_if_novalue:nTF 07:373,
 07:392, 07:439, 07:994, 07:2383,
 07:3283, 07:3296, 07:3297, 07:3298,
 07:3299, 07:3300, 07:3301, 08:326
 \tl_if_novalue_p:n .. 11:1255, 07:3283
 \tl_if_single:nTF
 11:640, 07:2472, 07:3263
 \tl_if_single_token:nTF
 09:331, 07:682, 07:2670, 07:3265, 214
 \tl_item:Nn 07:2815
 \tl_log:n 08:1843
 \tl_map_function:nN 07:101, 07:393,
 07:599, 07:600, 07:1009, 07:1059, 1436
 \tl_map_inline:Nn 07:732
 \tl_map_inline:nn 07:1527, 07:1778,
 07:1780, 07:1782, 07:1784, 07:1960
 \tl_map_tokens:nn 07:3320, 1436
 \tl_new:N 08:1572, 08:1573,
 08:1574, 09:6, 09:8, 09:9, 09:10,
 11:76, 11:77, 11:80, 11:81, 11:82,
 11:83, 11:84, 11:97, 11:98, 11:99,
 11:1631, 16:14, 36:159, 48:55, 48:56,
 48:57, 48:65, 48:66, 48:131, 48:211,
 48:212, 52:8, 52:9, 52:10, 52:11,
 52:59, 53:114, 07:12, 07:13, 07:14,
 07:15, 07:20, 53:266, 07:24, 07:31,
 07:32, 07:33, 07:36, 07:41, 07:42,
 07:44, 07:45, 07:54, 07:55, 55:126,
 55:308, 55:324, 55:325, 55:329,
 55:330, 55:331, 55:332, 07:1869,
 07:1870, 07:2086, 07:2469, 07:3315,
 08:25, 08:26, 08:27, 08:29, 08:32,
 08:72, 08:90, 08:118, 08:132, 08:135,
 08:159, 08:160, 08:293, 08:311, 08:1316
 \tl_put_left:Nn 07:855, 07:864, 07:1583
 \tl_put_right:Nn
 09:184, 09:198, 09:259, 09:273,
 11:390, 11:397, 11:846, 11:932,
 11:1092, 11:1100, 11:1127, 11:1138,
 11:1155, 11:1180, 11:1187, 07:368,
 07:394, 57:676, 07:535, 07:568,
 07:591, 07:605, 07:622, 07:627,
 07:764, 07:884, 07:897, 07:924,
 07:932, 07:946, 07:948, 07:955,
 07:983, 07:999, 07:1005, 07:1074,
 07:1099, 07:1267, 07:1275, 07:1291,
 07:1298, 07:1559, 07:1562, 07:1649,
 07:1652, 07:1677, 07:1687, 07:1694,
 07:1703, 07:1724, 07:1729, 07:1735,
 07:1878, 07:1879, 07:1886, 07:1896,
 07:1964, 07:1998, 07:2243, 07:2255,
 07:2271, 07:2282, 07:2485, 07:2530
 \tl_replace_all:Nnn .. 11:370, 07:2498
 \tl_rescan:nn .. 09:496, 09:572, 09:600
 \tl_set:Nn 08:1581, 08:1601, 08:1602,
 08:1607, 08:1608, 08:1623, 08:1630,
 08:1632, 08:1663, 08:1684, 08:1685,
 08:1691, 08:1693, 08:1712, 08:1719,
 08:1721, 08:1768, 08:1775, 09:182,
 09:195, 09:196, 09:199, 09:201,
 09:222, 09:257, 09:270, 09:271,
 09:274, 09:276, 09:297, 09:406,
 09:411, 09:493, 09:538, 09:569,
 09:597, 11:346, 11:369, 11:413,
 11:424, 11:428, 11:580, 11:697,
 11:701, 11:742, 11:814, 11:841,
 11:878, 11:880, 11:916, 11:976,
 11:977, 11:1032, 11:1058, 11:1076,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 11:1077, 11:1198, 11:1210, 11:1238,
48:103, 48:115, 52:81, 52:82, 52:83,
52:84, 53:108, 53:197, 53:271,
53:297, 53:322, 07:25, 53:349, 07:92,
07:123, 07:310, 07:311, 07:312,
07:313, 07:341, 55:309, 55:317,
07:391, 55:334, 55:335, 55:336,
55:337, 55:378, 55:379, 07:903,
07:1111, 07:1129, 07:1331, 07:1555,
07:1590, 07:1611, 07:1665, 07:1682,
07:1756, 07:1851, 07:1957, 07:1963,
07:1984, 07:1991, 07:2004, 07:2010,
07:2016, 07:2022, 07:2028, 07:2034,
07:2040, 07:2046, 07:2077, 07:2099,
07:2122, 07:2465, 07:2466, 07:2492,
07:2516, 07:2547, 07:2558, 07:2586,
07:2652, 07:2654, 07:2773, 07:2774,
08:579, 08:1170, 08:1177, 08:1181
`\tl_set_eq:nN` 08:1611,
08:1635, 08:1696, 08:1724, 09:213,
09:288, 10:148, 11:857, 48:168,
48:169, 07:331, 07:337, 07:345,
07:387, 55:380, 55:381, 07:1584, 07:1890
`\tl_show:n` 08:1850,
08:1857, 08:1961, 08:2062, 07:1373,
07:1386, 07:1395, 07:1431, 07:1453
`\tl_tail:N` 07:2156
`\tl_to_str:N`
... 11:471, 07:2227, 07:2234, 07:2841
`\tl_to_str:n`
... 08:2009, 08:2022, 08:2093,
08:2165, 08:2179, 09:129, 09:134,
09:491, 09:510, 09:539, 09:550,
09:586, 11:58, 11:136, 11:393,
11:505, 11:581, 11:792, 11:1198,
11:1296, 11:1297, 11:1298, 11:1307,
11:1308, 36:10, 36:19, 36:51, 36:64,
36:92, 36:93, 36:100, 36:101, 36:120,
36:122, 36:124, 36:129, 36:131,
36:134, 36:167, 36:168, 36:209,
48:259, 48:274, 51:122, 51:151,
51:153, 52:580, 07:74, 07:85, 07:252,
07:256, 07:464, 07:472, 07:485,
07:496, 07:500, 07:532, 07:563,
07:648, 07:677, 07:685, 07:700,
07:713, 07:725, 07:744, 07:786,
07:787, 07:817, 07:818, 07:1265,
07:1339, 07:1345, 07:1559, 07:1563,
07:2228, 07:2235, 07:2425, 07:2448,
07:2526, 07:2527, 07:2687, 07:2691,
08:378, 08:540, 08:568, 08:629, 1155
`\tl_trim_spaces:n`
... 09:323, 11:392, 11:413,
11:425, 11:581, 11:605, 11:633,
11:792, 11:841, 11:1198, 50:549,
51:112, 51:143, 51:178, 51:192,
52:41, 52:42, 07:248, 07:420, 07:535,
07:569, 07:817, 07:1573, 07:1575,
07:2424, 07:2447, 07:2547, 07:3143,
07:3147, 07:3151, 07:3155, 08:407, 343
`\tl_trim_spaces_apply:nN` . 07:682,
07:2550, 07:2574, 07:2664, 08:328
`\tl_use:N` 08:1834, 11:1261,
24:415, 28:291, 28:295, 28:343,
28:347, 36:94, 36:107, 36:124,
07:1410, 07:1504, 08:1078, 08:1141
`\l_tmpa_tl` 373
tl internal commands:
`\c__mark_empty_tl`
... 48:54, 48:228, 48:230, 48:232, 48:377
`\g__mark_first_marks_tl`
... 48:163, 48:168, 48:199, 48:211, 1076
`\g__mark_last_marks_tl`
... 48:164, 48:169, 48:188, 48:211, 1076
`\g__mark_new_top_tl`
... 48:63, 48:141, 48:142, 48:148, 48:150
`\l__mark_saved_parameters_tl` ...
48:103, 48:110, 48:115, 48:128, 48:131
`\g__mark_tmp_tl` 48:63,
48:143, 48:145, 48:153, 48:174,
48:176, 48:186, 48:189, 48:253,
48:259, 48:264, 48:266, 48:267, 1079
`\__tl_if_empty_if:n` 07:3287
`\__tl_if_noalue_aux:w`
... 07:3288, 07:3294
`\tmspace` 38:214, 895
`\to` 30:451, 30:453
`toc/contentsline/after` (tag socket) 55:229
`toc/contentsline/before` (tag socket) 55:229
`toc/leaders/after` (tag socket) ... 55:233
`toc/leaders/before` (tag socket) .. 55:233
`toc/starttoc/after` (tag socket) .. 55:231
`toc/starttoc/before` (tag socket) .. 55:231
`\today` 01:168,
01:172, 01:180, 01:183, 44:33, 1348
token commands:
`\c_math_subscript_token` 07:1842
`\c_math_toggle_token` 07:1840
`\c_space_token` 07:1675, 07:2763
`\token_case_charcode:NnTF`
... 07:1671, 07:1710
`\token_if_active:NnTF` 07:2257, 07:2273
`\token_if_cs:NnTF` 07:2474, 202
`\token_if_eq_catcode:NnTF`
... 09:333, 07:1908
`\token_if_eq_charcode:NnTF`
... 07:415, 07:2144, 07:2176, 07:2187

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\token_if_eq_meaning:NNTF</code>	09:203, 09:278, 09:497, 52:454, 52:466, 07:691, 07:694, 07:702, 07:1332, 07:1835, 07:2777, 07:3267, 07:3269
<code>\token_if_long_macro_p:N</code>	 07:1426, 07:2827
<code>\token_if_macro:NNTF</code>	 09:100, 08:1226, 08:1252, 336
<code>\token_if_math_toggle_p:N</code>	 07:2620, 07:2642
<code>\token_if_protected_long_macro_p:N</code>	 07:1427, 07:1436, 07:2828
<code>\token_if_protected_macro:NNTF</code>	 07:1195, 07:1220, 07:1387
<code>\token_if_protected_macro_p:N</code>	07:1435
<code>\token_to_meaning:N</code>	09:96, 09:494, 09:506, 09:570, 09:598, 16:41, 16:72
<code>\token_to_str:N</code>	 09:95, 09:96, 09:118, 09:556, 09:563, 09:591, 09:594, 11:370, 11:371, 11:388, 11:395, 11:398, 11:405, 11:811, 11:1459, 26:348, 36:57, 52:241, 07:74, 07:82, 07:85, 07:420, 07:833, 07:1106, 07:1260, 07:1409, 07:1433, 07:1455, 07:1921, 07:2067, 07:2262, 07:2274, 07:2424, 07:2447, 07:2674, 07:2687, 07:2691, 07:2700, 07:3110, 07:3111, 07:3124, 07:3125, 07:3229, 07:3230, 07:3243, 07:3244, 08:375, 201
<code>\toks</code>	 04:36, 16:40, 16:71, 16:80, 28:740, 28:741, 28:751, 28:760, 02:31, 02:59, 57:780, 451
<code>\toksdef</code>	 04:229, 02:46, 02:59
<code>\tokszero</code>	 04:229
<code>\tolerance</code>	 24:745, 24:791, 24:806, 49:131, 49:139, 02:187
<code>\top</code>	 30:331
<code>top:level (hook)</code>	 220, 220, 239
<code>\topfigrule</code>	 54:1078, 54:3026
<code>\topfraction</code>	 45:273, 54:2992
<code>\topmargin</code>	 54:47, 54:911, 54:979, 54:1038
<code>\TopMark</code>	 48:402, 48:524, 1064
<code>\topmark</code>	 54:2913, 54:2922, 1058
<code>\topsep</code>	 38:617, 39:1, 39:59, 1384
<code>\topskip</code>	 20:59, 20:128, 20:183, 54:106, 02:293, 489
<code>\totalheight</code>	 40:33, 40:34, 40:35
<code>totalpages</code>	 1189
<code>trace commands:</code>	
<code>trace_stack_levels</code>	 57:96
<code>\tracefloats</code>	 54:2506
<code>\tracefloatsoff</code>	 54:2506
<code>\tracefloatvals</code>	 54:2506, 1292
<code>\traceoff</code>	 437
<code>\tracelon</code>	 437
<code>\tracingall</code>	 02:432, 437
<code>\tracingassigns</code>	 02:449, 02:483, 02:500, 02:540
<code>\tracingcommands</code>	02:222, 02:447, 02:465, 02:481, 02:490, 02:503, 02:526, 02:543
<code>\tracingfonts</code>	 26:17, 26:55, 26:59, 26:87, 26:119, 26:154, 26:195, 26:225, 26:239, 26:255, 26:261, 26:274, 26:281, 26:288, 26:293, 26:302, 26:315, 26:323, 26:326, 1358
<code>\tracinggroups</code>	 02:439, 02:474, 02:512, 02:551
<code>\tracingifs</code>	 02:440, 02:475, 02:511, 02:550
<code>\tracinglostchars</code>	 24:741, 24:755, 02:216, 02:219, 02:433, 02:460, 02:472, 02:491, 02:534, 02:554, 1437
<code>\tracingmacros</code>	 02:211, 02:446, 02:464, 02:480, 02:492, 02:514, 02:533, 02:553
<code>\tracingnesting</code>	 02:442, 02:477, 02:509, 02:548
<code>\tracingnone</code>	 02:496, 1431
<code>\tracingoff</code>	 26:119, 26:323
<code>\tracingon</code>	 26:120, 26:324, 1359
<code>\tracingonline</code>	 02:572, 24:754, 54:2492, 02:210, 02:427, 02:502, 02:525, 02:542
<code>\tracingoutput</code>	 02:569, 02:215, 02:428, 02:506, 02:529, 02:546
<code>\tracingpages</code>	 02:214, 02:437, 02:459, 02:471, 02:491, 02:515, 02:535, 02:555
<code>\tracingparagraphs</code>	 02:213, 02:438, 02:461, 02:473, 02:492, 02:513, 02:532, 02:552
<code>\tracingrestores</code>	02:223, 02:448, 02:466, 02:482, 02:492, 02:501, 02:531, 02:541
<code>\tracingscantokens</code>	 02:441, 02:456, 02:476, 02:510, 02:523, 02:549
<code>\tracingstacklevels</code>	 02:224, 02:433, 02:444, 02:508, 22
<code>\tracingstats</code>	 57:2, 02:212, 02:436, 02:458, 02:470, 02:490, 02:516, 02:536, 02:556
<code>\triangle</code>	 30:333
<code>\triangleleft</code>	 30:369, 30:493
<code>\triangleright</code>	 30:370, 30:493
<code>\TrimSpaces</code>	 07:3316, 124
<code>trivlist (env.)</code>	 39:89
<code>\trivlist</code>	 37:446, 37:496, 37:498, 37:529, 37:551, 38:601, 39:89, 41:78, 43:66, 43:68, 43:73, 43:75, 916
<code>TS1 commands:</code>	
<code>\TS1:?</code>	 831
<code>\TS1:<family></code>	 831
<code>\tt</code>	 1363

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=lt page.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \ttdefault 29:13, 29:309, 29:479, 29:496,
29:520, 29:539, 29:567, 30:62, 740
- ttfamily (hook) 244
- \ttfamily 29:11,
29:12, 29:489, 29:537, 29:538,
29:565, 29:566, 32:17, 37:573, 127
- ttfamily 29:499
- \ttsubstdefault 30:20, 30:32, 33:32, 33:53
- \twocolumn 54:204, 1398
- \twocolumn[...] 1392
- \twocolumn[] 489
- \typein 82, 82
- \typein 06:31, 33:1584, 33:1589
- \typeout 82
- \typeout .. 03:21, 03:38, 01:100, 10:59,
10:62, 10:64, 10:66, 10:68, 10:71,
10:74, 14:74, 20:201, 20:692, 20:693,
20:699, 20:733, 20:779, 20:793,
20:814, 01:156, 24:463, 01:181,
01:183, 01:195, 01:210, 25:2866,
01:217, 01:228, 29:139, 01:241,
06:3, 29:646, 29:901, 29:911, 29:921,
30:9, 30:136, 06:36, 06:43, 01:254,
01:267, 33:1180, 33:1217, 33:1224,
33:1377, 33:1380, 33:1383, 33:1581,
33:1582, 33:1583, 33:1586, 33:1587,
33:1588, 33:1593, 33:1597, 01:305,
06:706, 06:707, 46:8, 46:25, 06:741,
06:742, 06:747, 48:360, 48:379,
06:801, 06:802, 06:817, 06:819,
50:419, 50:435, 50:447, 50:1632,
50:1841, 50:1844, 53:165, 53:176,
53:209, 54:590, 54:2495, 54:2507,
55:340, 57:278, 57:734, 57:741,
57:753, 57:754, 57:762, 01:57, 106
- \typeoutdetails 33:1180, 33:1181,
33:1184, 33:1190, 33:1192, 33:1196,
33:1197, 33:1203, 33:1221, 33:1222,
33:1376, 33:1591, 33:1593, 33:1597
- \typesetnormalchapter 117
- U**
- \u 21:257, 21:407, 21:495, 21:612,
21:619, 21:639, 21:646, 21:778,
21:1270, 21:1345, 21:1346, 21:1361,
21:1362, 21:1371, 21:1372, 21:1385,
21:1386, 21:1387, 21:1411, 21:1412,
21:1437, 21:1438, 33:127, 33:158
- \uccode 57:240, 57:248, 57:255,
57:257, 57:260, 57:262, 57:584,
57:592, 57:599, 57:601, 57:604, 57:606
- \Uchar 21:1062
- \Ucharcat 29:735
- \uchyph 02:245
- \ulcdefault 25:2914, 25:2936
- \ulcshape 25:2911, 25:2914,
25:2932, 25:2935, 25:3218, 25:3219,
29:632, 29:706, 32:29, 57:786, 661
- \Umathaxis 40:351
- \Umathcode 04:30, 18:574, 30:15,
30:55, 30:62, 30:77, 33:123, 33:331,
37:579, 02:89, 57:160, 57:400, 57:578
- \Umathsubshiftdown 45:479, 45:482
- \Umathsubtopmax 45:479, 45:480
- \Umathsupbottommin 45:420, 45:421
- \Umathsupshiftup 45:420, 45:423
- \Umathunderbarkern 40:647
- \Umathunderbarrule 40:646
- \Umathunderbarvgap 40:645
- \unboldmath 29:726
- \UndeclareTextCommand
..... 21:209, 33:79, 33:81, 33:83,
33:272, 33:549, 33:1091, 33:1092,
33:1093, 33:1094, 33:1095, 33:1096, 802
- \undefined 22:199, 22:200, 22:251,
22:252, 22:253, 22:298, 22:299,
22:300, 01:12, 01:14, 29:812, 30:127,
30:128, 35:40, 35:41, 35:57, 35:58,
35:59, 35:60, 01:20, 57:168, 57:180,
57:181, 57:201, 57:370, 01:41, 1393
- \undefinedpagestyle 49:4, 49:8
- \underbar 06:985, 06:1006, 02:397
- \underbrace 30:551
- \underline 926
- \underline .. 40:614, 40:615, 02:397, 1442
- \underscore 1399
- \unexpanded 22:214, 22:265,
32:48, 37:281, 37:317, 06:724,
06:735, 50:867, 50:1668, 50:1670, 338
- \unhbox 1360
- \unhcopy 41:422, 42:752, 42:808, 02:399, 943
- \unicodedataline
04:143, 04:146, 04:160, 04:161, 04:162
- \UnicodeEncodingName
..... 21:1005, 21:1009, 21:1066,
21:1072, 21:1076, 21:1087, 21:1091,
21:1107, 21:1108, 33:340, 33:341,
33:342, 33:343, 33:344, 33:345,
33:346, 33:347, 33:348, 33:349,
33:350, 33:351, 33:352, 33:353,
33:354, 33:355, 33:356, 33:357, 543
- \UnicodeFontFile 21:1064
- \UnicodeFontName 21:1065
- \UnicodeFontTeXLigatures
..... 21:1017, 21:1057, 21:1061
- \unicoderead 04:143,
04:157, 04:158, 04:159, 04:160, 04:165
- uninstall 04:1014

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfssscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\unitlength</code>	40:74, 40:85, 40:94, 40:104, 42:5, 42:40, 42:41, 42:43, 42:45, 42:53, 42:54, 42:55, 42:56, 42:71, 42:74, 42:84, 42:85, 42:95, 42:96, 42:104, 42:105, 42:118, 42:119, 42:130, 42:175, 42:187, 42:252, 42:267, 42:327, 42:329, 42:343, 42:351, 42:353, 42:368, 42:386, 42:388, 42:403, 42:408, 42:410, 42:425, 42:428, 42:433, 42:490, 42:491, 42:518, 42:519, 42:547, 42:548, 42:622, 42:638, 42:658, 42:664, 42:700, 42:701, 42:703, 42:704, 42:707, 42:708, 42:710, 42:711, 42:722, 42:723, 42:725, 42:726, 42:728, 42:729, 42:731, 42:732, 42:760, 42:761, 42:763, 42:764, 42:767, 42:768, 42:770, 42:771, 42:782, 42:784, 42:786, 42:788, 53:28, 53:53, 57:149, 1185	05:195, 36:73, 48:126, 48:144, 48:157, 48:175, 48:195, 48:204, 48:261, 48:380, 48:381, 48:382, 52:442, 52:446, 52:448, 52:465, 57:644, 07:829, 07:1501, 07:1837, 365	
<code>\unkern</code>	24:774	<code>\use:n</code>	08:1649, 08:2102, 08:2136, 08:2389, 08:2395, 08:2404, 09:121, 09:188, 09:191, 09:219, 09:228, 09:263, 09:266, 09:294, 09:300, 09:366, 09:417, 09:440, 09:459, 09:488, 09:518, 09:540, 09:547, 09:583, 11:10, 11:13, 11:385, 11:429, 11:789, 11:808, 20:542, 48:125, 51:210, 07:190, 07:692, 07:695, 07:891, 07:1135, 07:1262, 07:1354, 07:1462, 07:1643, 07:1659, 07:1802, 07:2607, 07:2649, 07:3264, 07:3266, 07:3271, 08:232, 08:382, 08:516, 08:571, 08:1174, 501
<code>\unless</code>	04:151, 04:159, 04:161	<code>\use:nn</code>	08:1755, 08:2152, 08:2727, 09:233, 09:305
<code>\unlhd</code>	29:888	<code>\use:nnn</code>	07:3110, 07:3124, 07:3229, 07:3243, 211
<code>\unpenalty</code>	24:777, 24:781, 32:117, 37:545, 37:567, 448	<code>\use_i:nn</code>	08:2254, 08:2599, 11:761, 07:986, 07:1879, 07:2702, 07:2755, 373
<code>\unrhd</code>	29:890	<code>\use_i:nnn</code>	08:2101, 07:2680, 287
<code>\unsetattribute</code>	41	<code>\use_i:nnnn</code>	09:364, 344
<code>\unsetattribute</code>	04:82, 04:239	<code>\use_i_delimit_by_q_recursion_</code>	
<code>\unskip</code>	448	<code>stop:nw</code>	09:447, 09:466
<code>\UnusedTemplateKeys</code>	11:784, 11:939, 11:1242, 11:1245, 11:1252, 11:1631, 374	<code>\use_i_delimit_by_q_stop:nw</code>	07:429
<code>\unvbox</code>	1227	<code>\use_ii:nn</code>	08:2605, 09:73, 09:155, 11:763, 55:38, 55:48, 07:1582, 07:1800, 07:2061, 07:2296, 07:2307, 07:2344, 07:2360, 07:2385, 07:2416, 07:2439, 07:2701, 07:3275, 08:659, 192
<code>\unvcopy</code>	38:193, 1066	<code>\use_ii:nnn</code>	07:1862, 07:2359, 07:2684, 07:3268
<code>\Uparrow</code>	30:595	<code>\use_ii:nnnn</code>	09:378, 09:386, 09:389
<code>\uparrow</code>	30:589	<code>\use_ii_i:nn</code>	09:232, 09:304
<code>\upbracefill</code>	30:554, 30:571	<code>\use_ii_iii:nnn</code>	52:216, 52:230, 52:242, 52:242
<code>\updefault</code>	25:3208, 29:22, 30:106, 30:113, 30:114, 30:122, 30:124, 772	<code>\use_iii:nn</code>	287
<code>\Updownarrow</code>	30:599	<code>\use_iii:nnn</code>	52:47, 07:2323, 07:2679, 07:2682, 07:2754, 07:3270
<code>\updownarrow</code>	30:593	<code>\use_iii:nnnn</code>	07:2322
<code>\uplus</code>	30:389	<code>\use_iv:nnnn</code>	07:415, 07:421
<code>\uppercase</code>	1373	<code>\use_none:n</code>	08:1753, 08:2256, 08:2345, 09:369, 09:447, 09:466, 10:7, 10:8, 48:340, 52:131, 52:134, 53:68, 07:99, 55:3, 55:4, 55:37, 55:47, 55:343, 55:345, 07:696, 07:1799, 07:1803, 07:1857, 07:1858, 07:1861, 07:1864, 07:1878, 07:1905, 07:2042, 07:2354, 07:2355,
<code>\upshape</code>	21:463, 21:756, 21:826, 21:919, 25:3206, 25:3207, 29:20, 29:21, 29:632, 29:643, 29:676, 29:706, 29:762, 32:24, 33:593, 57:786, 663		
<code>\Upsilon</code>	30:315		
<code>\upsilon</code>	30:297		
use commands:			
<code>\use:N</code>	08:1651, 08:1784, 08:1785, 08:1825, 08:2337, 08:2407, 08:2423, 10:158, 10:165, 11:615, 11:643, 11:826, 11:828, 11:898, 11:1078, 11:1234, 28:300, 28:350,		

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=lt page.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

07:2358, 07:2361, 07:2557, 08:7, 08:660, 08:1370, 08:1441, 08:1488, 307	\UseRawInputEncoding 57:430, 57:486, 57:533
\use_none:nn 08:2093, 08:2134, 11:508, 28:299, 28:349, 07:1152, 07:2036, 07:2317, 07:2318, 07:2321, 07:2324, 07:2586	\UseSocket 10:192, 10:219, 35:142, 35:150, 35:155, 54:498, 54:610, 54:614, 54:616, 54:638, 54:710, 54:714, 355
\use_none:nnn 08:2099, 11:488, 07:902, 07:2030, 07:2315, 07:2745, 07:3162, 07:3168, 07:3179, 07:3185, 08:766, 192	\UseStructureName 1310
\use_none:nnnn 08:2976, 07:2024	\UseStructureName . 55:132, 55:153, 55:304
\use_none:nnnnn 08:1872, 08:1990, 07:2018	\UseTaggingSocket 1309
\use_none:nnnnnn 07:2012	\UseTaggingSocket 17:135, 17:159, 35:151, 54:494, 54:618, 54:916, 54:931, 55:26, 55:182, 55:186, 55:194, 55:197, 55:393, 55:429, 55:444, 1309
\use_none:nnnnnnnn 07:2006	\UseTemplate 11:175, 11:1549, 410
\use_none:nnnnnnnnn 08:1872, 08:1990, 08:1370	\UseTextAccent 21:167, 21:168, 21:206, 33:74, 33:75, 33:77, 33:120, 33:122, 33:900, 33:1085, 33:1086, 33:1088, 33:1089, 1381
\use_none_delimit_by_q_recursion- stop:w 07:344	\UseTextSymbol 21:168, 21:204, 33:73, 33:316, 33:899, 33:968, 520
\use_none_delimit_by_q_stop:w 07:2521, 07:2540, 07:2544	\usetikzlibrary 224
\usebox 40:200	\ushape 1400
\usecounter 39:256, 39:269	
\UseExpandableTaggingSocket 1310	
\UseExpandableTaggingSocket . 55:26, 1322	
\usefont 21:1602, 24:81, 24:345, 24:750, 29:851, 33:7, 33:577, 37:583, 761	
\UseHook 08:2832, 08:2950, 09:306, 09:591, 09:594, 20:335, 20:341, 20:346, 20:357, 20:396, 20:400, 20:403, 26:146, 28:413, 29:342, 29:353, 29:365, 29:375, 29:396, 29:403, 29:416, 29:423, 29:482, 29:487, 29:492, 29:497, 29:775, 29:807, 29:822, 37:250, 37:262, 37:265, 37:353, 37:357, 37:370, 37:373, 50:1046, 50:1050, 50:1074, 50:1078, 52:170, 52:171, 52:174, 52:175, 53:179, 53:231, 53:435, 54:491, 54:505, 54:848, 54:880, 54:944, 1162	
\UseHookWithArguments 08:2832, 09:235, 09:556, 09:563, 35:100, 35:118, 330	
\UseInstance 11:1446, 11:1549, 377	
\UseLegacyTextSymbols ... 33:540, 33:763	
\UseName 77	
\UseName 05:191, 05:209	
\UseOneTimeHook 08:2832, 08:2951, 20:15, 20:57, 20:70, 20:336, 20:340, 20:345, 20:358, 20:397, 20:399, 20:402, 37:14, 37:19, 37:31, 37:32, 37:34, 37:82, 37:86, 37:96, 37:97, 37:99, 50:1047, 50:1051, 50:1073, 50:1077, 235	
\UseOneTimeHookWithArguments 08:2832, 220	
\usepackage .. 50:711, 50:773, 50:1648, 1190	
	V
	\v 21:258, 21:408, 21:494, 21:615, 21:616, 21:617, 21:621, 21:623, 21:626, 21:628, 21:630, 21:636, 21:642, 21:643, 21:644, 21:648, 21:650, 21:653, 21:655, 21:657, 21:663, 21:779, 21:1275, 21:1355, 21:1356, 21:1357, 21:1358, 21:1367, 21:1368, 21:1401, 21:1402, 21:1407, 21:1408, 21:1419, 21:1420, 21:1427, 21:1428, 21:1431, 21:1432, 21:1454, 21:1455, 21:1456, 21:1457, 21:1458, 21:1459, 21:1460, 21:1461, 21:1462, 21:1463, 21:1464, 21:1473, 21:1474, 21:1475, 21:1476, 21:1479, 21:1480, 33:129, 33:159
	\vadjust 18:38, 18:60, 18:72, 18:101, 18:108, 18:410, 18:426, 18:444, 18:460, 45:201, 45:223, 443
	\valign 33:65, 33:891
	\value 556
	\value 22:29, 22:31, 22:65, 22:85, 22:127, 22:139, 36:258, 47:9, 1360
	\varbigtriangledown 30:373, 30:376
	\varbigtriangleup 30:374, 30:375
	\varepsilon 19:15, 30:302
	\varphi 30:307
	\varpi 30:304
	\varrho 30:305
	\varsigma 30:306
	\vartheta 30:303

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\vbadness` 53:273, 53:275,
 53:324, 53:326, 54:2910, 02:189, 1202
`\vbox` 1227
 vbox commands:
 `\vbox_set:Nn` 48:76
 `\vbox_set_split_to_ht:NNn` ... 48:109
 `\vbox_set_to_ht:Nnn` .. 53:276, 53:327
 `\vbox_to_zero:n` 53:26, 53:51
 `\vbox_unpack:N` 48:84, 48:97,
 48:98, 48:421, 48:453, 53:287, 53:337
`\vcenter` 935
`\vdash` 30:415
`\vdots` 30:521
`\vec` 30:535
`\vector` 14:269, 42:244, 42:827, 42:844
`\vee` 30:378, 30:380
`\verb` ... 37:577, 37:593, 37:606, 37:620,
 37:632, 37:638, 37:644, 37:657,
 37:669, 37:674, 37:682, 37:688,
 37:698, 37:703, 37:716, 37:718, 126
`\verb*` 1412
`verbatim (env.)` 37:571
`\verbatim` 37:571
`verbatim* (env.)` 37:586
`\verbvisiblespace`
 ... 37:577, 37:579, 37:593, 37:598,
 37:608, 37:612, 37:622, 37:626, 1412
`\Vert` 30:582, 30:584
`\vert` 30:587
`\vfil` 33:66, 33:69,
 33:892, 33:895, 42:576, 42:588,
 53:446, 53:458, 54:154, 54:181,
 54:199, 54:417, 54:464, 54:610,
 54:914, 54:982, 54:1041, 02:386, 1249
`\vfill` 1250
`\vfilleq` 02:386
`\vfuzz` 49:134, 49:141, 53:272,
 53:274, 53:323, 53:325, 02:266, 1202
`\vglue` 02:376
`\vline` 41:443
`\voffset` 02:282
`\vphantom` 21:520, 21:536, 38:75
`\vrule` 18:550, 21:313,
 21:315, 21:525, 21:541, 26:191,
 30:569, 30:570, 30:572, 30:573,
 40:209, 40:211, 40:287, 40:294,
 40:613, 40:714, 40:721, 41:238,
 41:261, 41:296, 41:424, 41:443,
 42:238, 42:310, 42:313, 42:335,
 42:344, 42:361, 42:370, 42:394,
 42:402, 42:417, 42:424, 42:575,
 42:588, 42:736, 42:792, 54:2428,
 54:2887, 54:2930, 54:2960, 02:380, 1432
`\vsize` 1395
`\vskip` 1234
`\vspace` 18:399, 18:469, 18:470, 18:471
`\vsplit` 54:387, 54:434, 54:2912, 1075
- W**
- `\wedge` 30:377, 30:379
`\whatsit` 04:194, 41
`\widehat` 30:538
`\widetilde` 30:537
`\widowpenalty` 24:764, 02:196
`\width` 40:30, 1380
`\wlog` 04:6, 04:7, 04:8,
 04:54, 37:47, 37:112, 50:449, 02:40,
 57:60, 02:99, 57:789, 02:170, 01:84, 36
`\wp` 30:323
`\wr` 30:393
`\write` 1439
- X**
- `\x` 03:96, 03:99, 17:80, 17:90, 17:99,
 17:109, 24:396, 24:397, 57:382, 57:384
`\xdef` 1359
`\XeTeXcharclass` 24:737, 57:39,
 57:47, 57:54, 57:67, 57:73, 57:82, 57:89
`\XeTeXcharclassCL` 57:173
`\XeTeXcharclassCM` 57:177
`\XeTeXcharclassEX` 57:174
`\XeTeXcharclassID` 57:171
`\XeTeXcharclassIS` 57:175
`\XeTeXcharclassNS` 57:176
`\XeTeXcharclassOP` 57:172
`\XeTeXcharglyph` 21:1058
`\XeTeXdashbreakstate` 57:275
`\XeTeXglyph` 21:1058
`\XeTeXintercharclasses` .. 57:167, 57:200
`\XeTeXinterchartoks`
 57:168, 57:182, 57:183,
 57:184, 57:185, 57:186, 57:187,
 57:188, 57:189, 57:190, 57:191,
 57:192, 57:193, 57:194, 57:195,
 57:196, 57:201, 57:206, 57:207,
 57:208, 57:209, 57:210, 57:211,
 57:212, 57:213, 57:214, 57:215,
 57:216, 57:217, 57:218, 57:219, 57:220
`\XeTeXmathcode` 57:161, 57:579
`\XeTeXrevision` 57:41
`\XeTeXuseglyphmetrics` ... 57:272, 57:274
`\XeTeXversion` 21:1056, 29:734, 57:41
`\Xi` 30:312
`\xi` 30:292
`xpos` 36:266, 854
`\xpt` 1363
`\xspaceskip` 02:297

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\xtxHanGlue</code>	57:180, 57:204, 57:212, 57:213, 57:214, 57:215, 57:216, 57:217, 57:218, 57:219, 57:220	<code>\year</code>	03:11, 03:14, 01:169, 50:1341, 50:1474, 50:1563, 02:261
<code>\xtxHanSpace</code> . . .	57:181, 57:205, 57:206, 57:207, 57:208, 57:209, 57:210, 57:211	<code>ypos</code>	<u>36:266</u> , 854
Y		Z	
<code>\y</code>	03:97, 03:99	<code>\Z</code>	57:252, 57:575, 57:596
		<code>\z</code> . . .	04:858, 04:859, 57:243, 57:576, 57:587
		<code>\zeta</code>	30:284

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx