

enumext

ENUMERATE EXERCISE SHEETS

V2.4 2026-11-01^{*}

©2024–2026 by Pablo González L[†]

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

Abstract

This package provides enumerated list environments compatible with *tagging* PDF for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the “*answers*” to these in memory using *multicol* package.

Contents

1	Introduction	2	5.7	Keys for multicol	12
1.1	Description and usage	3	5.8	Keys for minipage	13
1.2	The concept of left margin	3	5.8.1	The command <code>\miniright</code>	13
1.3	User interface	4	5.8.2	The key <code>mini-right</code>	13
1.3.1	Public counters	4	6	The storage system	13
1.3.2	Public dimension	4	6.1	Keys for storage system	13
1.3.3	Support for multicol	4	6.1.1	Keys for label and ref	14
1.3.4	Support for minipage	4	6.1.2	Keys for wrap and marks	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	5	6.1.3	Keys for debug and checking	15
1.3.6	Support for <code>\footnote</code>	5	6.2	The command <code>\anskey</code>	16
2	The environments provided	5	6.2.1	Keys for <code>\anskey</code>	16
2.1	The environment <code>enumext</code>	5	6.3	The environment <code>anskey*</code>	16
2.2	The environment <code>enumext*</code>	5	6.3.1	Keys for <code>anskey*</code>	17
2.3	The command <code>\item*</code>	6	6.4	The environment <code>keyans</code>	17
2.3.1	Keys for <code>\item*</code>	6	6.4.1	The <code>\item*</code> in <code>keyans</code>	18
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.5	The environment <code>keyanspic</code>	19
2.4.1	Items on the whole row: the <code>item-break</code> key	6	6.5.1	Keys for <code>keyanspic</code>	19
3	The command <code>\setenumext</code>	7	6.5.2	The command <code>\anspic</code>	19
4	The command <code>\setenumextmeta</code>	7	6.6	Printing stored content	20
5	The keyval system	7	6.6.1	The command <code>\getkeyans</code>	20
5.1	Keys for label and ref	8	6.6.2	The command <code>\foreachkeyans</code>	21
5.2	Keys for penalties	9	6.6.3	The command <code>\printkeyans</code>	21
5.3	Keys for spaces	9	7	Full examples	23
5.3.1	Vertical spaces	9	8	Tagged PDF examples	25
5.3.2	Horizontal spaces	10	9	The way of non-enumerated lists	26
5.4	Keys for add code	11	10	References	29
5.5	Keys for start, series and resume	11	11	Change history	30
5.6	Keys for reset	12	12	Index of Documentation	31
5.6.1	The command <code>\resetenumext</code>	12	13	Implementation	33
			14	Index of Implementation	172

Motivation and acknowledgments

Usually, it is enough to use the classic `enumerate` environment to generate “*simple exercise sheets*” or “*multiple choice questions*”, the basic idea behind `enumext` is to cover three points:

1. To have a simple interface to be able to write “*lists of exercises*” with “*answers*”.
2. To have a simple interface for writing “*multiple choice questions*”.
3. To have a simple interface for placing “*columns*” and “*drawings*” or “*tables*”.

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all $\text{\LaTeX}$ team for their great work and to the different members of the $\text{\TeX-SX}$ community who have provided great answers and ideas. Here are notes on the main ones:

1. Answer given by Alan Munn in `\topsep`, `\itemsep`, `\partopsep`, `\parsep` - what do they each mean (and what about the bottom)?

^{*}This file describes a documentation for v2.4, last revised 2026-11-01.

[†]E-mail: pablgonz@educarchile.cl.

2. Answer given by Enrico Gregorio in [Understanding minipages - aligning at top](#)
3. Answer given by Ulrich Diez in [Different mechanics of hyperlink vs. hyperref](#)
4. Answer given by Enrico Gregorio in [Minipage and multicols, vertical alignment](#)

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[1] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TE_X. It has been tested with the standard classes provided by L^AT_EX: `book`, `report`, `article` and `letter` on 10pt, 11pt and 12pt.

 The minimum requirement is L^AT_EX release 2026-11-01.

1 Introduction

In the L^AT_EX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like `exam`[2] and packages like `xsim`[3] do the job perfectly, but they don’t always fit the basic day to day needs.

In the work of many teachers it is common to use “simple exercise sheets” also known as “informal lists of exercises”, for example:

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
 - (a) $\alpha > \delta$
 - (b) L^AT_EXze is cool?
4. Related to Linux
 - (a) You use Linux?
 - (b) Usually uses the package manager?
 - (c) Rate the following package and class
 - i. `xsim-exam`
 - ii. `xsim`
 - iii. `exsheets`

Sometimes it is also useful to show the “answers” along with the questions:

1. Factor $x^2 - 2x + 1$
* $(x - 1)^2$
2. Factor $3x + 3y + 3z$
* $3(x + y + z)$
3. True False
 - (a) $\alpha > \delta$
* False
 - (b) L^AT_EXze is cool?
* Very True!
4. Related to Linux
 - (a) You use Linux?
* Yes
 - (b) Usually uses the package manager?
* Yes, dnf
 - (c) Rate the following package and class
 - i. `xsim-exam`
* doesn’t exist for now :(
 - ii. `xsim`
* very good
 - iii. `exsheets`
* obsolete

Or to refer to a specific question and its “answer”, for example:

The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.

Or to print all the “answers”:

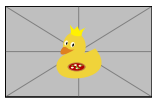
1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
(b) Very True!
4. (a) Yes
- (b) Yes, dnf
- (c) i. doesn’t exist for now :(
ii. very good
iii. obsolete

Another very common thing to use is “multiple choice questions”, for example:

1. First type of questions

A) value	C) value	A) value	D) value
B) correct	D) value	B) value	E) value
		C) value	
2. Second type of questions
 - I. $2\alpha + 2\delta = 90^\circ$
 - II. $\alpha = \delta$
 - III. $\angle EDF = 45^\circ$

A) I only	D) I and III only
B) II only	E) I, II, and III
C) I and II only	
- ★ 3. Third type of questions
 - (1) $2\alpha + 2\delta = 90^\circ$
 - (2) $\angle EDF = 45^\circ$
4. Question with image and label below:

		
A)	B)	C)
		
D)	E)	

5. Question with image on right side:

- A) value
- B) value
- C) value
- D) correct
- E) value



Here the `\label` and a “short note” left as an explanation are stored, and then printed:

1. B) $x = 5$

2. D)

3. C) some note
4. E) A duck

5. D) “other note”

The `enumext` package was created and designed to meet these small requirements in the creation of “simple worksheets” and “multiple choice questions”.

These “simple worksheets” or “multiple choice questions” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “looks simple” is not so simple.

1.1 Description and usage

The `enumext` package defines enumerated environments using the `list` environment provided by $\text{\LaTeX}$, but “does not redefine” any internal commands associated with it such as `\list`, `\endlist` or `\item` outside of the “scope” in which they are defined.

This package is NOT intended to replace the `enumerate` environment nor replace the powerful `enumitem`, the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex`»`dvips`»`ps2pdf` and is present in $\text{\TeX}$ Live and $\text{\MiKTeX}$, use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `luatex enumext.ins` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `arara enumext.dtx`.

```
enumext.sty  » TDS:tex/latex/enumext/
enumext.pdf  » TDS:doc/latex/enumext/
README.md   » TDS:doc/latex/enumext/
enumext.dtx  » TDS:source/latex/enumext/
enumext.ins  » TDS:source/latex/enumext/
```

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “extra space”, which makes it difficult to obtain the desired horizontal spaces in a `list` environment. Usually the `list` should not go beyond the left margin of the page, but since these four values are related, that causes a problem.

The `enumitem` package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

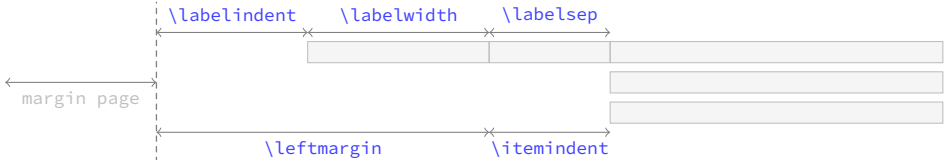


Figure 1: Representation of horizontal lengths in `enumitem`.

The `enumext` package does NOT provide a user interface to set the values for `\leftmargin` and `\itemindent`, instead it provides the keys `list-offset` and `list-indent` which internally set the values for `\leftmargin` and `\itemindent`. The concepts of `\leftmargin` and `\itemindent` are different in `enumext`. The figure 2 shows the visual representation of idea.

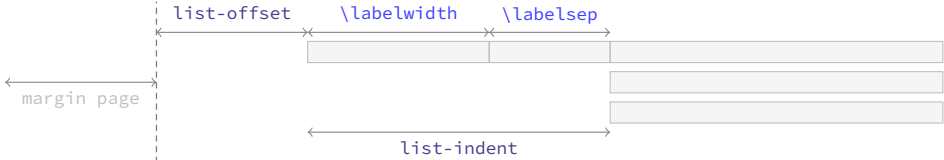


Figure 2: Representation of horizontal lengths concept in `enumext`.

In this way the number of parameters to pass is reduced a little. With the default values of keys `list-offset`, `list-indent`, `labelwidth` and `labelsep` the lists will have the (usually) expected output for “simple worksheets”. The figure 3 shows the visual representation.

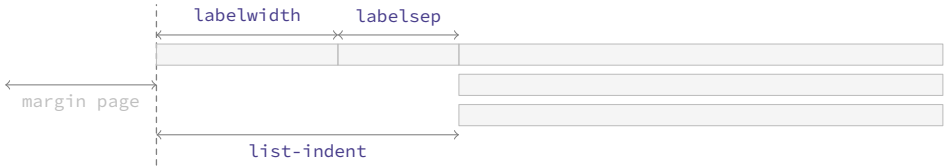


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to “store content” and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual *stored content*, `\printkeyans` to print all *stored content*, `\foreachkeyans` to print a range of *stored content*, `\miniright` for `minipage`, `\setenumext` to config [`<key = val`], `\setenumextmeta` to add a “meta-key” and `\resetenumext` to reset counters.

1.3.1 Public counters

The package `enumext` uses the `enumXi`, `enumXii`, `enumXiii`, `enumXiv` counters for the *four* nesting levels of the `enumext` environment, the `enumXv` counter for the `keyans` environment, the `enumXvi` counter for the `keyanspic` environment, the counter `enumXvii` for `enumext*` environment and the counter `enumXviii` for `keyans*` environment.

- If any package defines these *counters* or they are user-defined in the document, the package will return a “fatal error” and abort the load.

1.3.2 Public dimension

The package `enumext` only provides a *single public dimension* `\itemwidth` and is intended for user convenience only and is NOT for internal use as such. The dimension `\itemwidth` is a *rigid length* and contains the “width of the content” of each `\item` regardless of `labelwidth` and `labelsep`.

- If any package defines `\itemwidth` or the user defines `\itemwidth` in the document, the package will overwrite it without warning.

1.3.3 Support for multicol

The package provides direct support for using the `multicol[1]` package. This allows for a two-column output as shown in the figure 4.

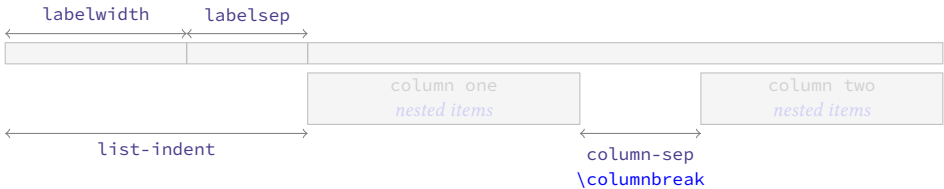


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (§5.7).

- The `\columnseprule` command is not available as a key and is set to “zero” for the inner levels and the `keyans` environment. If the value of this is set inside the document, it will affect “all environments” that use the `columns` key.

1.3.4 Support for minipage

The package provides direct support for `minipage` environment, this allows you to obtain an output like the one shown in figure 5.



Figure 5: Representation of the `mini-env` output for a nested level `enumext` environment.

The `minipage` environments on the “left side” and “right side” are always used with “aligned on top” [`t`]. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (§5.8).

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem`[5] package to customize the references which is activated by the `ref` key (§5.1), the standard $\LaTeX$ `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§6.1.1) when the key `save-ans` is active (§6.1).

1.3.6 Support for \footnote

The `enumext*` and `keyans*` environments and the `mini-env` key use the `minipage` environment in their implementation but in a transparent way for the user, it is only used for typesetting. The implementation supports `\footnote` and is compatible with `hyperref` and works the same way as if used anywhere in the document.

Unfortunately, if `hyperref` is loaded and *tagged* PDF is not active, it will not produce the “links”, the internal implementation uses `\footnotetext`[(*number*)] and `\footnotemark`[(*number*)]{(*text*)} and “links” for this are not supported by the `hyperref` package.

The best way to solve this if *tagged* PDF is NOT active is to use Jean-François Burnol’s `footnotehyper`[6] package. It will support keeping the “links” if `hyperref` is loaded with the `hyperfootnotes=true` option (default). Load it is as follows:

```
\IfDocumentMetadataF
{
  \usepackage{footnotehyper}
  \makesavenoteenv{enumext}
  \makesavenoteenv{enumext*}
}
```

At the moment the `footnotehyper` package v1.1f (2025/11/15) is not compatible with *tagged* PDF.

2 The environments provided

The package `enumext` provides two main list environments, the *vertical* environment `enumext` and the *horizontal* environment `enumext*`.

<code>enumext</code>	<code>\begin{enumext}[(keyval list)]</code>	<code>\begin{enumext*}[(keyval list)]</code>
<code>enumext*</code>	<code>\item <item content></code>	<code>\item <item content></code>
	<code>\item [(custom)] <item content></code>	<code>\item [(custom)] <item content></code>
	<code>\item* [(symbol)] [(offset)] <item content></code>	<code>\item* [(symbol)] [(offset)] <item content></code>
	<code>\end{enumext}</code>	<code>\end{enumext*}</code>

2.1 The environment enumext

The `enumext` is an environment that works in the same way as the standard `enumerate` environment provided by $\LaTeX$, `\item` and `\item[(custom)]` commands work in the usual way. The environment can be nested with at most “four levels” and the options can be configured globally using `\setenumext` command and locally using `[(key = val)]` in the environment.

Example with `columns=2`

1. This text is in the first level.

(a) This text is in the second level.

i. This text is in the third level.
- A. This text is in the fourth level.

X This text is in the first level.

★ 2. This text is in the first level.

2.2 The environment enumext*

The `enumext*` is a *horizontal list environment* similar to the `shortenumerate` or `tasks` environments provided by the `shortlst`[7] and `tasks`[8] packages, `\item` and `\item[(custom)]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[(key = val)]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “item content” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` does NOT include `labelwidth` and `labelsep`, only the *width of the content*. The only exception is an item that occupies the whole row when the key `item-break` is active (§2.4.1).
- You cannot have floating environments like `figure` or `table` or `\marginpar` but `\footnote` with `hyperref` is supported. If you want to activate links “without” *tagged* PDF active you must load `foot-notehyper` and `hyperref` packages (§1.3.6).

- The standard list environments `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse` and `list`, and the environments `center`, `flushleft`, `flushright`, `trivlist`, `tabbing`, `verbatim` and those created with `\newtheorem` can be used in the “*item content*”, also nested in themselves, because inside them `\item` is the standard command (§13.48.2).

Example with columns=2

1. This text is in the first level.
2. This text is in the first level.
- X This text is in the first level.
- ★ 4. This text is in the first level.

2.3 The command `\item*`

`\item*` `\item*[\langle symbol \rangle][\langle offset \rangle]`

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` work like the numbered `\item`, but places a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the `\langle offset \rangle` set by the *second optional argument*.

The *starred argument* ‘`*`’ cannot be separated by spaces ‘`\` ’ from the command, i.e. `\item*` and the *first optional argument* does NOT support *verbatim content*. It can be configured with the keys `item-sym*` and `item-pos*` locally in the environment or globally using `\setenumext` command (§3).

- The behavior of `\item*` in the `enumext` and `enumext*` environments is NOT the same as in the `keyans` and `keyans*` environments.

2.3.1 Keys for `\item*`

`item-sym*` = `{\langle symbol \rangle}` default: `\textborn`

Sets the *symbol* to be displayed to the “left” of the box containing the current `\langle label \rangle` set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in *text* or *math* mode, for example `item-sym*={\mathbb{N}}`.

`item-pos*` = `{\langle rigid length \rangle}` default: *by levels*

Sets the *offset* between the box containing the current `\langle label \rangle` defined by `labelwidth` key and the `\langle symbol \rangle` set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed, it will *offset to the left*; if negative values are passed it will *offset to the right*.

2.4 The command `\item` in `enumext*`

`\item` `\item(\langle columns \rangle)`

The `\item` command for the `enumext*` environment provides a “*first optional argument*” `\item(\langle columns \rangle)` which “*joins items*” between columns.

Let’s consider the following examples adapted directly from the `tasks` package:

```
\begin{enumext*}[widest=10,colums=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item The seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
    (\the\itemwidth)
  \item The ninth
  \item[Z] The tenth (\the\itemwidth)
\end{enumext*}
```

- | | | | |
|--|--|--------------|---------------|
| 1. The first | ★ 2. The second | 3. The third | 4. The fourth |
| ★ 5. The fifth item is way too long for this and needs three columns | 6. The sixth | | |
| 7. The seventh | X The eighth item is way too long for this and needs two columns (196.17749pt) | 9. The ninth | |
| Z The tenth (89.28171pt) | | | |

2.4.1 Items on the whole row: the `item-break` key

`item-break` = `{\langle true | false \rangle}` default: *false*

By default the “*item content*” is typeset in a `minipage`, so an item cannot be broken between two pages: if it does not fit in the rest of the page it moves entirely to the next one. When this key is *true* and an item occupies the *whole row*, the content is typeset directly in the paragraph, without the `minipage`, and it can continue on the next page.

An item occupies the whole row when it is `\item(<columns>)` with `<columns>` equal to the value of the `columns` key and it is the first item of a row, or when `columns=1`, where every item occupies the whole row. If `<columns>` is greater than the columns left in the row it is reduced, as explained in the *warnings* for joined items, so the item does NOT occupy the whole row and remains in a `minipage`.

The `<label>` is placed as always in the box set by `labelwidth` and `labelsep`, and the following lines of the item are indented by the same amount. The keys `listparindent` and `parsep` apply to the paragraphs of the item. The other items of the environment are not modified. The key works in `enumext*` and `keyans*`, locally and with `\setenumext` (§3).

- The vertical space around an item typeset without `minipage` can differ slightly from that of an item typeset in a box, because the item takes part in the paragraph and page breaking.

In the following example the third item occupies the whole row; if it were near the end of a page its text could continue on the next one:

```
\begin{enumext*}[columns=2,item-break=true]
  \item One
  \item Two
  \item(2) This item occupies the whole row and can be broken between pages.
  \item Four
\end{enumext*}
```

1. One
2. Two
3. This item occupies the whole row and can be broken between pages.
4. Four

3 The command `\setenumext`

<code>\setenumext</code>	<code>\setenumext{<key = val>}</code>	<code>\setenumext[<keyans*>]{<key = val>}</code>
<code>\setenumext[<enumext, level>]{<key = val>}</code>	<code>\setenumext[<print, level>]{<key = val>}</code>	
<code>\setenumext[<enumext*>]{<key = val>}</code>	<code>\setenumext[<print, *>]{<key = val>}</code>	
<code>\setenumext[<keyans>]{<key = val>}</code>	<code>\setenumext[<print*>]{<key = val>}</code>	

The command `\setenumext` sets the `<keys>` on a global basis for environments `enumext`, `enumext*`, `keyans`, `keyans*` and the `\printkeyans` command. It can be used both in the preamble and in the body of the document as many times as desired.

The `<keys>` set in the *optional argument* of environments and commands have the *highest precedence*, overriding both options passed by `\setenumext`. If the *optional argument* is not passed, the *first level* of the environment `enumext` will be taken by default.

- For security reasons the keys `resume` “with value”, `resume*`, `reset`, `reset*`, `series` and `save-ans` they can NOT be set by this command and are ignored. The key `save-ans` that activates the “storage system” must be passed directly in the *optional argument* of the “first level” of the environment in which they are executed.

4 The command `\setenumextmeta`

<code>\setenumextmeta</code>	<code>\setenumextmeta[<1>]{<key name>}{<key-one = val, key-two = val, ...>}</code>
<code>\setenumextmeta[<2>]{<key name>}{<key-one = val, key-two = val, ...>}</code>	
<code>\setenumextmeta[<3>]{<key name>}{<key-one = val, key-two = val, ...>}</code>	
<code>\setenumextmeta[<4>]{<key name>}{<key-one = val, key-two = val, ...>}</code>	
<code>\setenumextmeta[<*>]{<key name>}{<key-one = val, key-two = val, ...>}</code>	
<code>\setenumextmeta*{<key name>}{<key-one = val, key-two = val, ...>}</code>	

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the `{<key name>}` must be different from those defined by the package. The *optional argument* of the form `[1]`, `[2]`, `[3]`, `[4]` adds a new “meta-key” for levels 1, 2, 3 and 4 of the `enumext` environment, the form `[*]` adds a new “meta-key” for the `enumext*` environment. If it is run *without* the *optional argument*, it will add a new “meta-key” for the “first level” of the `enumext` environment.

The *starred argument* `*` will create the new “meta-key” for the environment `enumext*` and for “all levels” of the environment `enumext`. For example: `\setenumextmeta*{midsep}{topsep=3pt, partopsep=0pt}` will create a new key `midsep` available for all levels of the `enumext` environment and the `enumext*` environment and it can be used like any other key so `\begin{enumext}[midsep]` and `\begin{enumext*}[midsep]` will be valid.

5 The keyval system

The `<key = val>` system used by the `enumext` package is implemented using `l3keys` so it must be noted that `<keys>` marked as “value forbidden”, that is `<key>` is different from `<key=`.

All *⟨keys⟩* described in this section are available for the `enumext`, `enumext*`, `keyans` and `keyans*` environments with the exception of the keys `series`, `resume`, `resume*`, `reset` and `reset*` which are only available for the `enumext` and `enumext*` environments; and the keys `mini-right`, `mini-right*` which are only available for the `enumext*` and `keyans*` environments.

All *⟨keys⟩* related to vertical or horizontal spacing accept a “*skip*” or “*dim*” expression if passed between braces, i.e. you do not need to use `\dimeval` or `\dimexpr` to perform calculations.

- It should be kept in mind that using any *⟨key⟩* that sets *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inner levels* and `keyans`, `keyans*` and `keyanspic` environments.

5.1 Keys for label and ref

`mode-box` *⟨value forbidden⟩*

default: *not used*

This is a “*switch-key*” that does not receive an argument and is “only” available for the “*first level*” of the `enumext` environment and the `enumext*` environment. When this is set the `label`, `font`, `wrap-label` and `wrap-label*` keys are executed within `\makebox` for the `enumext` and `keyans` environments.

- This key is intended for compatibility with *tagged* PDF and is forcibly “*enabled*” when `\DocumentMetadata[9]` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.
- In the `enumext*` and `keyans*` environments `\makelabel` is redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

`label` = { *⟨\alph* | \Alph* | \arabic* | \roman* | \Roman*⟩* }

default: *by levels*

Sets the *⟨label⟩* that will be printed at the *current level* and default value for `labelwidth` key. The default value for the first level of the environments `enumext` and `enumext*` is `\arabic*`, for second level are *⟨\alph*⟩*, for third level are `\roman*` and for fourth level is `\Alph*`. For `keyans` and `keyans*` environments the default value is `\Alph*`.

- This key is intended to give the basic structure with which the *⟨label⟩* will be displayed, and the form in which it is used by standard “*label and ref*” and the “*internal label and ref*” system with the `save-ref` key. You cannot use commands with *⟨label⟩* as an argument, for example `\emph{⟨\alph*⟩}` will return an error. For full customization of how *⟨label⟩* is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

`labelsep` = { *⟨rigid length⟩* }

default: `0.3333em`

Sets the *horizontal space* between the box containing the current *⟨label⟩* defined by `label` key and the text of an item on the first line. Internally sets the value of `\labelsep` for the current level.

`labelwidth` = { *⟨rigid length⟩* }

default: *by label*

Sets the *width* of the label box containing the current *⟨label⟩* set by the `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter set by `label` key using ‘0’ for `\arabic*`, ‘M’ for `\Alph*`, ‘m’ for `\alph*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

`widest` = { *⟨integer | string⟩* }

default: *empty*

Sets the `labelwidth` key passing the *⟨integer⟩* or converting the *⟨string⟩* of the form `\Alph`, `\alph`, `\Roman` or `\roman` to a *value* for the current counter defined by `label` key, then calculating the *width* by means of a box. For example `widest={XXIII}` or `widest={23}` are equivalent. This key is useful when the default values of the `labelwidth` key are smaller than those actually used.

`font` = { *⟨font commands⟩* }

default: *empty*

Sets the *font style* for the current *⟨label⟩* defined by `label` key. You cannot use commands with an argument, for example `font={\small\textbf{#1}}` will return an error, but `font={\bfseries\small}` it will work. For full customization of how *⟨label⟩* is displayed use `wrap-label` and/or `wrap-label*` keys.

`align` = { *⟨left | right | center⟩* }

default: *left*

Sets the *aligned* of *⟨label⟩* defined by `label` key on the current level in the label box of width set by `labelwidth`.

`wrap-label` = { *⟨code {#1} more code⟩* }

default: *empty*

Wraps the *current* *⟨label⟩* defined by `label` key referenced by `{#1}` after executing the `align` and `font` keys. The *⟨code⟩* must be passed between braces and it does not modify the value set by the `labelwidth` key and is applied *only* on `\item` and `\item*`. When using it in the `\setenumext` command it is necessary to use the *double* ‘`{##1}`’. For example `wrap-label={\fbox{#1}}` or you can create a command:

```
\NewDocumentCommand \mywrap { s m }
{
  \IfBooleanTF{#1}
  {\textcolor{red}{\textbf{Q}}\textcolor{blue}{\textbf{.}}\textcolor{gray}{#2}}
  {\textcolor{blue}{\textbf{Q}}\textcolor{red}{\textbf{.}}\textcolor{gray}{#2}}
}
```

and then pass it through the key `wrap-label={\mywrap{#1}}` or `wrap-label={\mywrap*{#1}}`.

`wrap-label*` = { *⟨code {#1} more code⟩* }

default: *empty*

The same as the `wrap-label` key but also applies on `\item[⟨custom⟩]`.

`ref = {⟨code {⟨\alph*|⟨\Alph*|⟨\arabic*|⟨\roman*|⟨\Roman*⟩ more code⟩}⟩}` default: *empty*

Modifies the way *cross references* are displayed. The `label` key sets the default form of the *cross references*, by using this key you can define a different format, for example: `ref=\emph{⟨\alph*⟩}` is valid.

Internally it renews the command associated with each counter when it is executed, i.e., in the environment `enumext` the command `\theenumxi` is modified when the key is executed at the first level, `\theenumxii` when it is executed at the second level and `\theenumxiii` together with `\theenumxiv` when it is executed at the third and fourth levels.

- This must be kept in mind, since the values set by the `label` and `ref` keys are not cumulative by levels, so if you have used the `ref` key in the first level and then want to associate the counter with `label` or `ref` in the second level you must use the direct commands, i.e. `\arabic{enumxi}` to indicate the count of the first level instead of using `\theenumxi`.

5.2 Keys for penalties

Page breaks in the provided environments are controlled by the following three parameters, which work together to ensure they look good, avoiding unsightly page breaks that could distort the output.

`beginpenalty = {⟨integer⟩}` default: *-51*

Set the *page breaking* penalty for breaking at the beginning of the `enumext`, `enumext*`, `keyans` and `keyans*` environments. Internally sets the value of `\@beginparpenalty`.

`midpenalty = {⟨integer⟩}` default: *-51*

Set the *page breaking* penalty for breaking between items of the `enumext` and `keyans` environments. Internally sets the value of `\@itempenalty`. In the `enumext*` and `keyans*` environments the items are set inside the lines of a paragraph, so there is no vertical space between items where this penalty could be inserted and the key has no effect.

`endpenalty = {⟨integer⟩}` default: *-51*

Set the *page breaking* penalty for breaking at the end of the `enumext`, `enumext*`, `keyans` and `keyans*` environments. Internally sets the value of `\@endparpenalty`.

- The values passed to these *⟨keys⟩* affect the nested environments in which they were set and cannot be reset. $\text{\LaTeX}$ default is `-\@lowpenalty`, that is, *-51*. Because it is negative, it somewhat encourages a page break at each spot. Change it with, e.g., `\@beginparpenalty=9999`; a value of `10000` prohibits a page break. Please, refer to your $\text{\LaTeX}$ or $\text{\TeX}$ manual about how penalties control page breaks.

5.3 Keys for spaces

`show-length = {⟨true|false⟩}` default: *false*

Displays on the terminal the values for *all list parameters* at the current level. For *vertical spaces* show the values of `\topsep`, `\itemsep`, `\parsep` and `\partopsep`. For *horizontal spaces* show the values of `\labelwidth`, `\labelsep`, `\itemindent`, `\listparindent` and `\leftmargin`.

5.3.1 Vertical spaces

`topsep = {⟨rubber length|rigid length⟩}` default: *by levels*

Set the *vertical space* added to both the top and bottom of the list. Internally sets the value of `\topsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `8.0pt plus 2.0pt minus 4.0pt`, for second level are `4.0pt plus 2.0pt minus 1.0pt`, for third and fourth level are `2.0pt plus 1.0pt minus 1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt plus 2.0pt minus 1.0pt`.

`parsep = {⟨rubber length|rigid length⟩}` default: *by levels*

Set the *vertical space* between paragraphs within an item. Internally sets the value of `\parsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `4.0pt plus 2.0pt minus 1.0pt`, for second level are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `0pt`. For `keyans` and `keyans*` environments the default value is `2.0pt plus 1.0pt minus 1.0pt`.

- In the `enumext*` and `keyans*` environments this value is passed to `\parskip` within the `minipage` environment where “item content” is placed.

`partopsep = {⟨rubber length|rigid length⟩}` default: *by levels*

Set the *vertical space* added, beyond `topsep`, to the “top” and “bottom” of the entire environment if the environment instance is preceded by a “blank line” or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level in environment `enumext` are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`. For the `keyans` environment the default value is `2.0pt plus 1.0pt minus 1.0pt`, and for the `keyans*` and `enumext*` environments it is available but *without* effect.

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the

source code of document. T_EX will enter *(vertical mode)* and apply this value to the “top” and “bottom” the environment or nested level.

`itemsep = { (rubber length | rigid length) }` default: *by levels*

Set the *vertical space* between items, beyond the `parsep`. Internally sets the value of `\itemsep` for the current level. The default value for the first level of the environments `enumext` and `enumext*` are `4.0pt plus 2.0pt minus 1.0pt`, for the rest of the levels are `2.0pt plus 1.0pt minus 1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt plus 2.0pt minus 1.0pt`.

🔗 In the `enumext*` and `keyans*` environments this value corresponds to the separation between rows.

`noitemsep` *(value forbidden)* default: *not used*

This is a “meta-key” that does not receive an argument. Set `itemsep` and `parsep` equal to `0pt` the entire level of environment.

`nosep` *(value forbidden)* default: *not used*

This is a “meta-key” that does not receive an argument. Sets all keys for vertical spacing equal to `0pt` the entire level of environment.

`base-fix` *(value forbidden)* default: *not used*

This is a “switch-key” that does not receive an argument available *only* for the “first level” of environment `enumext`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` and there is no material between the `\item` and the start of the environment for example `\item \begin{enumext}` within the environment `enumext*`. Internally sets the keys `topsep`, `above` and `above*` at `0pt`.

🔗 This key is provided as a way to work around this minor issue, but you should be aware that if for some reason you have the `itemindent` key set in the `enumext*` environment it will be lost and you will need to adjust it using the `list-offset` key in the `enumext` environment.

Extra vertical spaces

🔗 The following *(keys)* should be used with “caution”, they are intended to be used at the “top” and “bottom” of the environment when the `columns` or `mini-env` keys do not provide adequate *vertical spaces*. The values passed can be *rubber* or *rigid* lengths, the way they are applied is the way you differ, using the star `*` *(keys)* applies `\vspace*` so that T_EX does *not discard* this space at page break.

`above = { (rubber length | rigid length) }` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discordable”.

`above* = { (rubber length | rigid length) }` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the top of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “not discordable”.

`below = { (rubber length | rigid length) }` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the bottom of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “below” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discordable”.

`below* = { (rubber length | rigid length) }` default: *not used*

Set the *extra vertical space* added, beyond `topsep`, to the bottom of the entire level of environment. This key is intended to give a “fine adjustment” of the vertical space on the “below” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “not discordable”.

5.3.2 Horizontal spaces

`list-offset = { (rigid length) }` default: `0pt`

Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent = { (rigid length) }` default: `labelwidth + labelsep`

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=0pt` is set in the environments `enumext` and `keyans` the *(label)* will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”.

🔗 The `enumext*` and `keyans*` environments are implemented using `\makebox` and `minipage` which causes “list indent” to always be equal to the value passed to `labelwidth` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

`itemindent = { (rigid length) }` default: `0pt`

Sets the extra *horizontal indentation*, beyond `labelsep`, of the “first line” of each `\item` that is not followed by a “blank line” or the `\par` command. This value must be greater than or equal to `opt` and is applied internally using `\hspace` without modifying the value of `\itemindent`.

- This key is intended for the `enumext*` and `keyans*` environments where, by their implementation, it is not possible to adjust `labelwidth` and `list-indent` without modifying the output. If you use `enumext` or `keyans` and want to get around the effect of a *blank line* or of the `\par` command followed by `\item` you can modify `labelwidth` and `list-indent` and get the same effect.

`rightmargin = {⟨rigid length⟩}` default: `opt`

Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to `opt`. Internally sets the value of `\rightmargin` for the current level.

`listparindent = {⟨rigid length⟩}` default: `opt`

Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.

- In the `enumext*` and `keyans*` environments this value is passed to `\parindent` within the `minipage` environment where “item content” is placed.

5.4 Keys for add code

The following *⟨keys⟩* should be used with “caution”, they are intended to inject `{⟨code⟩}` into different parts of the defined environments. It must be kept in mind that the defined environments are based on the `list` base environment provided by $\TeX$ which is defined (simplified) as plain form `\list{⟨arg one⟩}{⟨arg two⟩}`. Using the `before*` key does not allow access to the `list` parameters defined by `[⟨key = val⟩]`.

`before = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “before” the environment starts. The `{⟨code⟩}` must be passed between braces, is executed “after” all calculations related to the *list parameters* in the environment and the *⟨keys⟩* sets by `[⟨key = val⟩]` have been performed, that is, in the *second argument* of the list: `\begin{list}{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}`.

`before* = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “before” the environment starts. The `{⟨code⟩}` must be passed between braces, is executed “before” performing all calculations related to the *list parameters* and the *⟨keys⟩* sets in `[⟨key = val⟩]` of the environment that is, “before” the arguments defining the list environment are executed: `{⟨code⟩}\begin{list}{⟨arg one⟩}{⟨arg two⟩}`.

`first = {⟨code⟩}` default: *not used*

Executes `{⟨code⟩}` when “starting” the environment. The `{⟨code⟩}` must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of list, just before the first occurrence of `\item: \begin{list}{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}\item`.

- Keep in mind that the `{⟨code⟩}` set in this *⟨key⟩* will affect the entire “body” of the environment and therefore the inner levels of the list and the `keyans`, `keyans*` and `keyanspic` environments. It is recommended to set this *⟨key⟩* per level. In the `enumext*` and `keyans*` environments this *⟨key⟩* is executed “after” the `listparindent`, `parsep` and `itemindent` *⟨keys⟩* within the `minipage` environment in which the “item content” is placed. Making modifications to `\setcounter` in these environments using this *⟨key⟩* will not have the expected effect.

`after = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “after” finishing the environment. The `{⟨code⟩}` must be passed between braces.

5.5 Keys for start, series and resume

`start = {⟨integer | integer expression⟩}` default: `1`

Sets the *start value* of the numbering on the “current level”. The `{⟨integer expression⟩}` must be passed between braces, internally is evaluated and pass to the “counter” defined by `label` key on the current level, i.e. it is equivalent to enter `start={\dimeval{100*\value{chapter}}` or `start={100*\value{chapter}}`.

`start* = {⟨integer | string⟩}` default: *not used*

Sets the *start value* of the numbering on the “current level”. Internally *⟨string⟩* is converted and passed as value to the “counter” defined by `label` key on the current level, i.e. it is equivalent to enter `start*=5`, `start*=E` or `start*=v`.

- The *start value* in `enumext` and `keyans` are set in the *second argument* of the `list` environment and “before” the execution of the keys `before` and `first`, but in the `enumext*` and `keyans*` it is set much later: `\begin{list}{⟨arg one⟩}{⟨arg two⟩}\setcounter`.

- The following *⟨keys⟩* are available only for the `enumext` and `enumext*` environments.

`series = {⟨series name⟩}` default: *not used*

Stores the *keys* of the *optional argument* of the “current level” of the environment in which it is executed in `{⟨series name⟩}` which is used as an *argument* in the `resume` key. The *⟨keys⟩* stored in `{⟨series name⟩}` are NOT

cumulative and are overwritten if the same $\{\langle series\ name\rangle\}$ is used again at the “same level” at which the key was executed.

- For security reasons the `series` key will never save in $\{\langle series\ name\rangle\}$ the $\langle keys\rangle$ series, `resume`, `resume*`, `reset`, `reset*`, `save-ans`, `save-key`, `start*` and `start`.

`resume` = $\{\langle series\ name\rangle\}$

default: *not used*

Sets the *start value* and *options* for the “current level” continuing the numbering and *options* of the “same level” as the environment in which the `series`= $\{\langle series\ name\rangle\}$ key was executed, the *start value* will continue numbering according to the last execution of `resume`= $\{\langle series\ name\rangle\}$. If passed “without value” this will only set *start value* continue the numbering of the “same level” from the last environment and level in which `series`= $\{\langle series\ name\rangle\}$ or `resume`= $\{\langle series\ name\rangle\}$ is NOT present and if the `save-ans` key is active (on the left) it will continue the numbering from the “last” environment in which it was executed. The *start value* can be overwritten using `start` or `start*` keys.

- The `resume` key passed “without value” must be exactly “without value”, i.e. `resume`= cannot be used and if executed before `resume*` it will affect the *start value*.

`resume*` $\langle value\ forbidden\rangle$

default: *not used*

Sets the *start value* and *options* for the “current level” continuing the numbering and options of the “same level” as the last environment and level in which the `series`= $\{\langle series\ name\rangle\}$ or `resume`= $\{\langle series\ name\rangle\}$ keys are NOT present and if the `save-ans` key is active (on the left) it will continue the numbering and options from the “last” environment in which it was executed. The *start value* can be overwritten using `start` or `start*` keys.

- When using the key `resume`= $\{\langle series\ name\rangle\}$ or `resume*` you will have hierarchy in the $\langle keys\rangle$ that are stored in $\{\langle series\ name\rangle\}$ or in an internal version of $\{\langle series\ name\rangle\}$ in the case of `resume*`. If you want to *reset* the value of a $\langle key\rangle$ that is already stored in $\{\langle series\ name\rangle\}$ or in an internal version of $\{\langle series\ name\rangle\}$ this must be placed to the *right* of the key `resume`= $\{\langle series\ name\rangle\}$ or `resume*`.
- When the `resume*` key is executed consecutively, it does not rewrite the $\langle keys\rangle$ stored in the internal version of $\{\langle series\ name\rangle\}$ and if the environment that precedes it does not have the *optional argument*, it will just continue with the numbering.

5.6 Keys for reset

`reset` $\langle value\ forbidden\rangle$

default: *not used*

Resets the *start value* of the “counters” in the `enumext` and `enumext*` environments along with the “internal counters” used by the `resume` without value and `resume*` keys at the “level” at which it is executed. The *start value* can be overwritten using the `start` or `start*` keys.

`reset*` $\langle value\ forbidden\rangle$

default: *not used*

Resets the *start value* of the “counters” in the `enumext` and `enumext*` environments along with the “internal counters” used by the `resume` without value and `resume*` keys at the “level” at which it is executed and in the “levels below” it in the case of the `enumext` environment. The *start value* can be overwritten using the `start` or `start*` keys.

- These keys are intended to be used in cases where the `\resetenumext` command does not work, e.g. after an unnumbered chapter. It should preferably be set *only* on the *first level*, although it is available for all levels.

5.6.1 The command `\resetenumext`

<code>\resetenumext</code>	<code>\resetenumext[$\langle 1\rangle$]{$\langle some\ counter\rangle$}</code>	<code>\resetenumext[$\langle 4\rangle$]{$\langle some\ counter\rangle$}</code>
	<code>\resetenumext[$\langle 2\rangle$]{$\langle some\ counter\rangle$}</code>	<code>\resetenumext[$\langle *\rangle$]{$\langle some\ counter\rangle$}</code>
	<code>\resetenumext[$\langle 3\rangle$]{$\langle some\ counter\rangle$}</code>	<code>\resetenumext*{$\langle some\ counter\rangle$}</code>

The `\resetenumext` command “resets” the *start value* of the “counters” for the `enumext` and `enumext*` environments along with the “internal counters” used by the keys `resume` without value and `resume*` according to the value of $\{\langle some\ counter\rangle\}$. For example `\resetenumext{chapter}` will “reset” the numbering of “all levels” of the `enumext` environment for each execution of a “numbered” chapter.

The *optional argument* of the form $\langle 1\rangle$, $\langle 2\rangle$, $\langle 3\rangle$, $\langle 4\rangle$ “reset” the values for levels 1, 2, 3 and 4 of the `enumext` environment, the form $\langle *\rangle$ “reset” the values for the `enumext*` environment. If is run *without* the *optional argument*, it will “reset” the values for “all levels” of the `enumext` environment.

The *starred argument* ‘ $\ast$ ’ will “reset” the values for “all levels” of the `enumext` and `enumext*` environments.

5.7 Keys for multicol

`columns` = $\{\langle integer\rangle\}$

default: 1

Set the *number of columns* to be used by the `multicol` environment within the environments `enumext` and `keyans`. The value must be a positive integer less than or equal to 10. In the `enumext*` and `keyans*` environments they correspond to the default number of columns (without joining) and internally adjust the value of `\itemwidth`.

`columns-sep = {⟨rigid length⟩}`

default: *by level*

Set the *space between* columns used by the `multicols` environment within the environments `enumext` and `keyans`. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level. In the `enumext*` and `keyans*` environments they correspond to the *space between* columns (without joining) and internally adjust the value of `\itemwidth`.

5.8 Keys for minipage

`mini-env = {⟨rigid length⟩}`

default: *not used*

Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}`

default: `0.3333em`

Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

5.8.1 The command `\miniright`

```
\miniright \begin{enumext}[mini-env={⟨rigid length⟩}] ⟨items before⟩ \item \miniright ⟨content⟩ \end{enumext}
\begin{enumext}[mini-env={⟨rigid length⟩}] ⟨items before⟩ \item \miniright*⟨content⟩ \end{enumext}
```

The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”.

The *starred argument* “*” inhibits the use of `\centering` command i.e. the usual $\TeX$ justification is maintained in the `minipage` on the “right side”.

5.8.2 The key `mini-right`

In the *horizontal list environments* `enumext*` and `keyans*` it is not possible to use the `\miniright` command and the `mini-right` key must be used instead.

`mini-right = {⟨content⟩}`

default: *not used*

Set the *content* for the drawing or tabular to be placed in the `minipage` environment on the “right side” by starting it with `\centering`. The `{⟨content⟩}` must be passed between braces.

`mini-right* = {⟨content⟩}`

default: *not used*

Same as above, but *without* starting with `\centering`.

6 The storage system

The entire mechanism for “*storing content*” it is activated according to `save-ans` key on the “*first level*” of `enumext` or `enumext*` environments and it is ignored if they are established when they are nested inside each other. Only when this *⟨key⟩* is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
---	---

By executing the key `save-ans={⟨store name⟩}` the entire “*structure*” of the environment (excluding the *first level*) including the *optional argument* passed to the inner levels or the environment nested in it, along with the *⟨content⟩* passed to `\anskey` or `anskey*`, the current *⟨labels⟩* for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be “*stored*” in a *sequence* `{⟨store name⟩}` and at the same time will be “*stored*” (without the “*structure*” or *optional argument*) in a *prop list* `{⟨store name⟩}`.

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all *⟨keys⟩* related to the “*storage system*” (§6.1) along with the keys `mini-env`, `mini-sep`, `mini-right`, `mini-right*`, `series`, `resume` and `resume*` when storing in *sequence* `{⟨store name⟩}` set by `save-ans` key.

6.1 Keys for storage system

The only *⟨keys⟩* available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the *⟨keys⟩* described in this section must be passed directly in the *optional argument* of the “*first level*” of the environment in which the key `save-ans` is executed. The key `save-ans` should NOT be passed with the command `\setenumext`.

`save-ans = {⟨store name⟩}` default: *not set*

Sets the *name* of the *sequence* and *prop list* in which the {⟨contents⟩} will be “stored” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments and the current {⟨labels⟩} for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic`. If the *sequence* or *prop list* {⟨store name⟩} does not exist, it will be created globally and will not be *overwritten* if the key is used again.

`save-key = {⟨key list⟩}` default: *not set*

This key *overrides* the default “stored keys” of the *optional argument* of the inner levels or nested environment that will be passed to the *sequence* {⟨store name⟩} set by `save-ans` key. The {⟨key list⟩} passed to this key ignores any {⟨keys⟩} in the “stored structure” and must be passed between braces. For example, if executed at a second level:

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{enumext}[nosep, columns=2, save-key={columns=3}]
      ...
    \end{enumext}
\end{enumext}
```

The “stored keys” by default in the *sequence* {⟨store name⟩} would be `nosep` and `columns=2`, but using the key `save-key={columns=3}` will overwrite and the “stored key” in the *sequence* {⟨store name⟩} are only `columns=3` ignoring all the others.

`save-sep = {⟨text symbol⟩}` default: {,}

Sets the *text symbol* that will separate the current {⟨label⟩} to the *optional argument* passed to the `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` and storing them in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. The {⟨text symbol⟩} must always be passed between braces, whitespace ‘ ’ is preserved within the braces and only affects the “stored content” and not what is displayed when using the `show-ans` or `show-pos` keys.

`no-store` {value forbidden} default: *not used*

This is a “switch-key” that does not receive an argument and disables the “storing content” in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey` command or use `anskey*` environment and “without” interfering with the `check-ans` key.

6.1.1 Keys for label and ref

`save-ref = {⟨true | false⟩}` default: *false*

Activates the “internal label and ref” mechanism for referencing “stored content” in *prop list* {⟨store name⟩} set by `save-ans` key. To reference the location of the “stored content” within the environment you must use `\ref{⟨store name : position⟩}`, where {⟨position⟩} corresponds to the position occupied by the “stored content” in the *prop list* {⟨store name⟩} returned by the `show-pos` key. For example `\ref{test:4}` will return `3`. (b) which corresponds to the location of the “stored content” at position `4` in *prop list* `test` within the environment in which the key `save-ans=test` was set.

`mark-ref = {⟨symbol⟩}` default: `\textreferencemark`

Sets the *symbol* that will be displayed by the `\printkeyans` command only if the `hyperref` package is detected and the `save-ref` key are active. This “symbol” is used as a “link” between the environment in which the `save-ans` key was used and the place where the command is executed.

6.1.2 Keys for wrap and marks

The `enumext` package provides a set of {⟨keys⟩} to set and manipulate “symbol marks” associated with “answers” and how they are displayed and stored in the *sequence* and *prop list*.

The {⟨keys⟩} available for the `\anskey` command and the `anskey*` environment can be passed “only” in the *optional argument* in the “first level” of the `enumext` or `enumext*` environment.

The {⟨keys⟩} available for the `keyans` and `keyans*` environments can be passed locally in the *optional argument*, at the “first level” of the `enumext` or `enumext*` environment or via the `\setenumext` command with one minor difference, when {⟨keys⟩} are passed through the “first level” of the `enumext` or `enumext*` environment they are set in “both” environments, but when they are passed using the `\setenumext` command they are set “individually” in each environment.

`show-ans = {⟨true | false⟩}` default: *false*

Display the *symbol* set by the `mark-ans` key to the left of the *mandatory argument* {⟨content⟩} passed to the `\anskey` command and {⟨body⟩} for the `anskey*` environment using the `wrap-ans` key if set. For `\item*` and `\anspic*` the `keyans`, `keyans*` and `keyanspic` environments it will display the *symbol* set by the `mark-ans*` key to the left of the current {⟨label⟩} and *optional argument*. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

Keys for `\anskey` and `anskey*`

`mark-ans` = {`\symbol`} default: `\textasteriskcentered`

Sets the *symbol* to be displayed in the left margin for `\anskey` command and `anskey*` environment when using the key `show-ans`. The “*symbol*” is placed in a box of width equal to the value of `labelwidth` at the current level, separated by the value of the key `mark-sep` and aligned by the value of the key `mark-pos`. This key is not affected by the keys `font` or `wrap-label` so if you want to apply *styling* you have to do it directly, for example: `mark-ans={\textcolor{red}{\textbf{\textasteriskcentered}}}`.

`mark-pos` = {`\left` | `right` | `center`} default: `right`

Sets the *aligned* of the “*symbol*” defined by `mark-ans` key for `\anskey` command and `anskey*` environment. The “*symbol*” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `mark-sep` key.

`mark-sep` = {`\rigid length`} default: `labelsep`

Sets the *horizontal space* between the box containing the “*symbol*” defined by `mark-ans` key and the *mandatory argument* `\content` passed to the `\anskey` command and the `\body` in `anskey*` environment.

`wrap-ans` = {`\code {#1} more code`} default: `\fbox+\parbox{#1}`

Wraps the *mandatory argument* `\content` passed to the `\anskey` and the `\body` in `anskey*` environment referenced by `{#1}` when using the `show-ans` or `show-pos` keys. The `{\code}` must be passed between braces and only affects how the `\content` or `\body` is displayed and NOT the “*stored content*” in the *sequence* and *prop list* `{\store name}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.

Keys for `keyans`, `keyans*` and `keyanspic`

`mark-ans*` = {`\symbol`} default: `\textasteriskcentered`

Sets the *symbol* to be displayed in the left margin for `\item*` and `\anspic*` for the `keyans`, `keyans*` and `keyanspic` environments when using the key `show-ans`. The “*symbol*” is placed in a box of width equal to the value of `labelwidth` of the environment in which it is executed, separated by the value of the key `mark-sep*` and aligned by the value of the key `mark-pos*`. This key is not affected by the keys `font` or `wrap-label` so if you want to apply *styling* you have to do it directly, for example: `mark-ans*={\textcolor{red}{\textbf{\textasteriskcentered}}}`.

`mark-pos*` = {`\left` | `right` | `center`} default: `right`

Sets the *aligned* of the “*symbol*” defined by `mark-ans*` key for the `keyans`, `keyans*` and `keyanspic` environments. The “*symbol*” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key of the environment in which it is executed and separated by the value of the `mark-sep*` key.

`mark-sep*` = {`\rigid length`} default: `labelsep`

Sets the *horizontal space* between the box containing the “*symbol*” defined by `mark-ans*` key and the current `\label` for `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments.

`wrap-ans*` = {`\code {#1} more code`} default: `not used`

Wraps the *current* `\label` when using the `show-ans` key for `\item*` and `\anspic*` referenced by `{#1}` in the `keyans`, `keyans*` and `keyanspic` environments after executing the `align` and `font` keys. The `{\code}` must be passed between braces and *only* affects how the `\label` is displayed and NOT the “*stored label*” in the *sequence* and *prop list* `{\store name}` set by `save-ans` key. This key overwrites the key `wrap-label` and if is passed using `\setenumext` it is necessary to use double `{##1}`. For example, if you want the `\label` to be displayed in red when using `show-ans` you just set `wrap-ans*={\textcolor{red}{#1}}`.

`wrap-opt` = {`\code {#1} more code`} default: `[{#1}]`

Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by `{#1}` in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The `{\code}` must be passed between braces and only affects the current *optional argument* and NOT the “*stored content*” in the *sequence* and *prop list* `{\store name}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.

6.1.3 Keys for debug and checking

`show-pos` = {`\true` | `false`} default: `false`

Displays the *position* occupied by the “*stored content*” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* `{\store name}` set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans` = {`\true` | `false`} default: `false`

Enables the *checking answer* mechanism: a warning is shown on the terminal when the number of items and answers differ, and the result is written to the `.log` file. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “*only one answer*” or “*only one execution*” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.

6.2 The command \anskey

`\anskey` `\anskey[⟨keys⟩]{⟨content⟩}`

The command `\anskey` takes a mandatory non empty argument `{⟨content⟩}` and “stores” it in the *sequence* and *prop list* `{⟨store name⟩}` set by `save-ans` key. By design the command cannot be nested or passed *verbatim material* in the argument and it is assumed that each *numbered* `\item` or `\item*` within the environment in which it is active it has a “single execution” of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

If `save-ref` key are active and the `hyperref`[10] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by $\text{\LaTeX}$ will be used.

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

6.2.1 Keys for \anskey

By default the *mandatory argument* `⟨content⟩` passed to `\anskey` when “storing” in the *sequence* `{⟨store name⟩}` has the form `\item ⟨content⟩`, the following `⟨keys⟩` allow modifying the way in which it is “stored” in the *sequence*.

- `break-col` `⟨value forbidden⟩` default: *not used*
Stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\columnbreak \item ⟨content⟩`.
- `item-join` = `{⟨columns⟩}` default: *not set*
Set the *number of columns* to be used for `\item(⟨columns⟩)` and stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\item(⟨columns⟩) ⟨content⟩`.
- `item-star` `⟨value forbidden⟩` default: *not used*
Stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\item* ⟨content⟩`.
- `item-sym*` = `{⟨symbol⟩}` default: *not set*
Sets the *symbol* for `\item*` when using the key `item-star` and stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\item*[⟨symbol⟩] ⟨content⟩`. The *symbol* can be in text or math mode, for example `item-sym*={$\ast$}` stores `\item*[$\ast$] ⟨content⟩`.
- `item-pos*` = `{⟨rigid length⟩}` default: *not set*
Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\item*[⟨symbol⟩][⟨offset⟩] ⟨content⟩`.

Example

```
\begin{enumext}[save-ans=test,show-ans=true]
  \item* Text containing our instructions or questions. \anskey{first answer}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.\anskey{second answer}
    \end{enumext}
  \item Text containing our instructions or questions. \anskey{third answer}
  \item Text containing our instructions or questions. \anskey{fourth answer}
\end{enumext}
```

★ 1. Text containing our instructions or questions.

* first answer

2. Text containing our instructions or questions.

(a) Question.

* second answer

3. Text containing our instructions or questions.

* third answer

4. Text containing our instructions or questions.

* fourth answer

6.3 The environment anskey*

`anskey*` `\begin{anskey*}[⟨key = val⟩] ⟨body content⟩ \end{anskey*}`

The environment `anskey*` takes a mandatory `{⟨body content⟩}` and “stores it” in the *sequence* and *prop list* `{⟨store name⟩}` set by `save-ans` key. If `save-ref` key are active and the `hyperref`[10] package is detected `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by $\text{\LaTeX}$ will be used.

By design the environment cannot be nested but full supports “verbatim material” in the `⟨body⟩` and it is assumed that “each numbered” `\item` or `\item*` within the environment in which it is active it has a “single execution” unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the new “collect code” `c`-type argument part of $\text{\LaTeX}$ release 2025-06-01[11]. `\begin{anskey*}` and `\end{anskey*}` must be in different lines and should not appear

within verbatim environments or commands. All $\langle keys \rangle$ must be passed separated by commas and “without separation” of the start of the environment.

Comments ‘%’ or *any character* after $\backslash\text{begin}\{\text{anskey}^*\}$ or $[\langle key = val \rangle]$ on the same line are NOT supported, $\text{\LaTeX}$ will return an “error” message if this happens. In a similar way comments ‘%’ or *any character* after $\backslash\text{end}\{\text{anskey}^*\}$ on the same line $\text{\LaTeX}$ will return a “warning” message.

6.3.1 Keys for anskey*

The anskey^* environment uses the same $\langle keys \rangle$ as the $\backslash\text{anskey}$ command next to the $\langle keys \rangle$ write-env , overwrite and force-eol . The environment is available for all levels of the enumext environment and the enumext^* environment, but it is disabled for the keyans , keyans^* and keyanspic environments.

$\text{write-env} = \{\langle file.ext \rangle\}$ default: *not used*

Sets the *name* of the $\langle external\ file \rangle$ in which the $\langle contents \rangle$ of the environment will be written. The $\langle file.ext \rangle$ will be created in the working directory, relative or absolute paths are not supported. If $\langle file.ext \rangle$ does not exist, it will be created or overwritten if the overwrite key is used.

$\text{overwrite} = \{\langle true \mid false \rangle\}$ default: *false*

Sets whether the $\langle file.ext \rangle$ generated by write-env from the anskey^* environment will be rewritten.

$\text{force-eol} = \{\langle true \mid false \rangle\}$ default: *false*

Sets if the *last end of line* for the $\langle stored\ content \rangle$ is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the fancyvrb package as $\backslash\text{end}\{\text{Verbatim}\}$ or another environment that does not support a comments ‘%’ after closing $\backslash\text{end}\{\text{Verbatim}\}\%$.

Example

```
\begin{enumext}[save-ans=test,show-pos=true,start=5]
  \item* Text containing our instructions or questions.
    \begin{anskey*}[item-star]
      \langle first answer \rangle
    \end{anskey*}

  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \langle second answer \rangle
        \end{anskey*}
    \end{enumext}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle third answer \rangle
    \end{anskey*}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle fourth answer \rangle
    \end{anskey*}
\end{enumext}
```

★ 5. Text containing our instructions or questions.

[5]

First answer with verbatim

6. Text containing our instructions or questions.

(a) Question.

[6]

second answer

7. Text containing our instructions or questions.

[7]

third answer

8. Text containing our instructions or questions.

[8]

fourth answer

6.4 The environments keyans and keyans*

keyans

$\backslash\text{begin}\{\text{keyans}\}[\langle key = val \rangle] \ \backslash\text{item} \ \backslash\text{item}[\langle custom \rangle] \ \backslash\text{item}^* \ \backslash\text{item}^*[\langle content \rangle] \ \backslash\text{end}\{\text{keyans}\}$

keyans*

$\backslash\text{begin}\{\text{keyans}^*\}[\langle key = val \rangle] \ \backslash\text{item} \ \backslash\text{item}[\langle custom \rangle] \ \backslash\text{item}^* \ \backslash\text{item}^*[\langle content \rangle] \ \backslash\text{end}\{\text{keyans}^*\}$

The keyans and keyans^* environments are “*enumerated list*” environments designed for “*multiple choice*” questions activated by the save-ans key.

This environments can NOT be nested and must always be at the “*first level*” of the enumext environment, the commands $\backslash\text{item}$ and $\backslash\text{item}[\langle custom \rangle]$ work in the usual and the command $\backslash\text{item}(\langle columns \rangle)$ is available for the keyans^* environment.

🟢 The behavior of $\backslash\text{item}^*$ in keyans and keyans^* environments is NOT the same as in the enumext or enumext^* environments.

```
\begin{enumext}[save-ans=test]
  \item <item content>
  \begin{keyans}[<key = val>]
    \item <item content>
    \item [<custom>] <item content>
    \item* <item content>
    \item* [<content>] <item content>
  \end{keyans}
\end{enumext}
```

```
\begin{enumext}[save-ans=test]
  \item <item content>
  \begin{keyans*}[<key = val>]
    \item <item content>
    \item [<custom>] <item content>
    \item* <item content>
    \item* [<content>] <item content>
  \end{keyans*}
\end{enumext}
```

The `<keys>` set in the *optional argument* of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have *higher precedence* than those set by `\setenumext[<keyans>]{<key = val>}` or `\setenumext[<keyans*>]{<key = val>}`. If the *optional argument* is not passed or the `<keys>` are not set by `\setenumext`, the default values will be the same as the “second level” of the `enumext` environment with the difference in the `<label>` which will be set to `label=\Alph*`.

The keys `mark-ans*`, `mark-pos*`, `mark-sep*`, `save-sep`, `wrap-opt`, `wrap-ans*`, `show-ans` and `show-pos` are available for both environments.

6.4.1 The `\item*` in `keyans` and `keyans*`

```
\item* \item*
\item* [<content>]
```

The `\item*` and `\item* [<content>]` command “store” the current `<label>` set by `label` key next to the *optional argument* `<content>` in *sequence* and *prop list* `{<store name>}` set by `save-ans` key in the “first level” of the `enumext` or `enumext*` environments.

The *starred argument* ‘`*`’ cannot be separated by spaces ‘`␣`’ from the command, i.e. `\item*` and the *optional argument* does “NOT” support *verbatim content*. By design it is assumed that the `\item*` will only appear “once” within the environment.

Example

```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.

  \begin{keyans*}[nosep,columns=2]
    \item Choice
    \item* Correct choice
    \item Choice
    \item Choice
    \item Choice
  \end{keyans*}

  \item Text containing a question and image.

  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item* [<note>] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}
    Some text
  \end{keyans}
\end{enumext}
```

1. Text containing a question.

A) Choice

* B) Correct choice

C) Choice

D) Choice

E) Choice

2. Text containing a question and image.

A) Choice

B) Choice

C) Choice

D) Choice

* E) [note] Correct choice



Some text

6.5 The environment keyanspic

```
keyanspic \begin{keyanspic}[\langle key = val \rangle] \anspic*[\langle content \rangle]{\langle drawing or tabular \rangle} \end{keyanspic}
```

The `keyanspic` environment is an “enumerated list” environment activated by the `save-ans` key that has the same configuration for “spacing” and `\label` as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing *drawings or tabular* with `\label` centered *above* or *below* in a *single line* or *upper* and *lower* layout style.

When the `keyanspic` environment is used *without keys* the `\label` are centered *below* the *drawings or tabular* in a *single line* layout style.

A representation of the output can be seen in the figure 6.

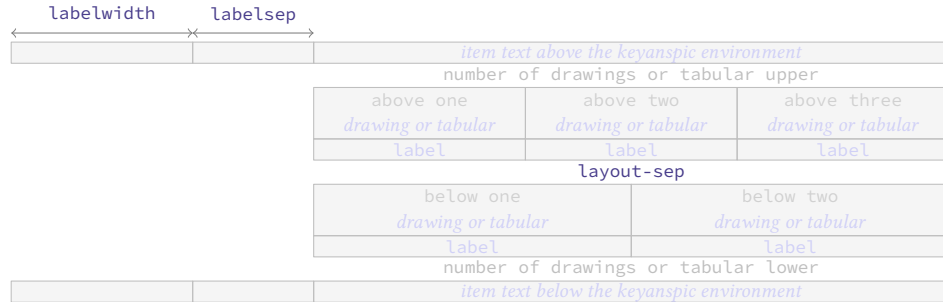


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

This environment cannot be nested and must *always* be at the “first level” of the `enumext` environment, the `\item` command is disabled and `\keys` cannot be set using `\setenumext`.

6.5.1 Keys for keyanspic

`label-pos = {\langle above | below \rangle}` default: *below*

Set the *position* of `\label` to be centered “above” or “below” *drawings or tabular* when the `\anspic` command is executed.

`label-sep = {\langle rubber length | rigid length \rangle}` default: *internal adjustment*

Set the *vertical spacing* between the `\label` centered “above” or “below” and *drawings or tabular* when running the `\anspic` command.

`layout-sty = {\langle n° upper , n° lower \rangle}` default: *not set*

Set the *number of drawings or tabular* that will be distributed “upper” and “lower” within the environment when executing the `\anspic` command. The value must be passed in braces and if not set or the `\langle n° lower \rangle` is omitted the *drawings or tabular* will be put on a *single line*.

`layout-sep = {\langle rubber length | rigid length \rangle}` default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings or tabular* placed at the “upper” and “lower” within the environment when executing the `\anspic` command. Internally adjusts the `parsep` value taken from the `keyans` environment.

`layout-top = {\langle rubber length | rigid length \rangle}` default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of `topsep` taken from `keyans` environment.

The keys `mark-ans*`, `mark-pos*`, `mark-sep*`, `save-sep`, `wrap-opt`, `wrap-ans*`, `show-ans` and `show-pos` are available for this environment.

6.5.2 The command \anspic

```
\anspic \anspic{\langle drawing or tabular \rangle}
\anspic*[\langle content \rangle]{\langle drawing or tabular \rangle}
```

The `\anspic` command takes three arguments, the *starred argument* ‘*’ store the current `\label` next to the *optional argument* `\langle content \rangle` in *sequence* and *prop list* `\langle store name \rangle` set by `save-ans` key.

The *starred argument* ‘*’ cannot be separated by spaces ‘ ’ from the command, i.e. `\anspic*` and the *optional argument* does “NOT” support *verbatim content*. By design it is assumed that the *starred argument* ‘*’ will only appear “once” within the environment.

Example

```
\begin{enumext}[save-ans=test,show-ans=true,nosep]
\item Question with images and labels below.

\begin{keyanspic}[layout-sty={3,2}]
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
```

```
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}

\item Question with images and labels above.

\begin{keyanspic}[label-pos=above, layout-sty={3,2}, layout-sep=0.25cm]
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}

\item Question with images and labels below on a single line.

\begin{keyanspic}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}

\end{enumext}
```

1. Question with images and labels below.



A)



B)



C)



D)



* E) [note]

2. Question with images and labels above.

A)



B)



C)



D)



* E) [note]



3. Question with images and labels below on a single line.



A)



B)



C)



D)



* E) [note]

Remember to pass the `alt={description}` key to the `\includegraphics` command when creating a tagged PDF.

6.6 Printing stored content

6.6.1 The command `\getkeyans`

`\getkeyans` `\getkeyans{<store name> : <position>}`

The command `\getkeyans` prints the “stored content” in *prop list* `{<store name>}` defined by `save-ans` key in the `<position>` returned by the `show-pos` key.

The “stored content” can only be accessed *after* it is stored, if `{<store name>}` does not exist the command will return an error.

The form taken by the argument $\{\langle store\ name\rangle : \langle position\rangle\}$ is the same as that used to generate the “internal label and ref” system when `save-ref` key are active, so to refer to a “stored content”. For example `\getkeyans{test:4}` will return the “stored content” at position 4 of the environment in which the key `save-ans=test` was set.

6.6.2 The command `\foreachkeyans`

```
\foreachkeyans <foreachkeyans>[<key = val>]{<store name>}
```

The command `\foreachkeyans` goes through and executes the command `\getkeyans` on the contents in *prop list* $\{\langle store\ name\rangle\}$. If you pass without options run `\getkeyans` on all contents in *prop list* $\{\langle store\ name\rangle\}$.

Options for command

`sep = {<code>}` default: {;}

Establishes the *separation* between “each” $\{\langle content\rangle\}$ stored in *prop list* $\{\langle store\ name\rangle\}$. For example, you can use `sep={\ \ [10pt]}` for vertical separation of stored contents.

`step = {<integer>}` default: 1

Sets the *step* (increment) applied to the value set by key `start` for “each” $\{\langle content\rangle\}$ stored in *prop list* $\{\langle store\ name\rangle\}$. The value must be a *positive integer*.

`start = {<integer>}` default: 1

Sets the *position* of the *prop list* $\{\langle store\ name\rangle\}$ from which execution will start. The value must be a *positive integer*.

`stop = {<integer>}` default: 0

Sets the *position* of the *prop list* $\{\langle store\ name\rangle\}$ from which execution will finish. The value must be a *positive integer*.

`before = {<code>}` default: empty

Sets the $\{\langle code\rangle\}$ that will be executed *before* each $\{\langle content\rangle\}$ stored in *prop list* $\{\langle store\ name\rangle\}$. The $\{\langle code\rangle\}$ must be passed between braces.

`after = {<code>}` default: empty

Sets the $\{\langle code\rangle\}$ that will be executed *after* each $\{\langle content\rangle\}$ stored in *prop list* $\{\langle store\ name\rangle\}$. The $\{\langle code\rangle\}$ must be passed between braces.

`wrapper = {<code> {#1} more code}` default: empty

Wraps the $\{\langle content\rangle\}$ stored in *prop list* $\{\langle store\ name\rangle\}$ referenced by `{#1}`. The $\{\langle code\rangle\}$ must be passed between braces. For example `\foreachkeyans[wrapper={\makebox[1em][l]{#1}}]{<store name>}`.

6.6.3 The command `\printkeyans`

```
\printkeyans <printkeyans>{<store name>}
\printkeyans[<keys>]{<store name>}
\printkeyans*[<keys>]{<store name>}
```

The command `\printkeyans` prints “all stored content” in *sequence* $\{\langle store\ name\rangle\}$ defined by `save-ans` key placing this inside the `enumext` or `enumext*` environment if the *starred argument* ‘*’ is used.

The “stored content” can only be accessed *after* it is stored in the *sequence*, if $\{\langle store\ name\rangle\}$ does not exist the command will return an error.

The *optional argument* allows managing the $\langle keys\rangle$ in the “first level” of the environment in which the “stored content” of the *sequence* $\{\langle store\ name\rangle\}$ will be printed, if the *starred argument* ‘*’ is used it will be `enumext*` otherwise `enumext`.

The default values for the “first level” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the *sequence* $\{\langle store\ name\rangle\}$ the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence* $\{\langle store\ name\rangle\}$ it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`. These are the default values of the package: the values set with `\setenumext` for the environments `enumext` and `enumext*` do not modify them.

Since the command encapsulates by default the `enumext` environment or the `enumext*` environment, some considerations must be taken into account:

- If `\printkeyans*{<store name>}` is executed and the *sequence* $\{\langle store\ name\rangle\}$ already contains any `enumext*` environment an error will be returned, since nesting is not possible.
- If `\printkeyans*{<store name>}` is executed and the *sequence* $\{\langle store\ name\rangle\}$ contains any `enumext` environments, they will start with the $\langle keys\rangle$ set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.

- If `\printkeyans{<store name>}` is executed and the *sequence* `{<store name>}` contains any environment `enumext*`, they will start with the `<keys>` set by default unless they are set in the *optional argument* or `save-key` is used to modify it.

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[<print , 1>]{<keys>}` and `\setenumext[<print*>]{<keys>}`.

To set the `<keys>` for the environment `enumext` “saved” in the *sequence* `{<store name>}`, use `\setenumext[<print , level>]{<keys>}`, and to set the `<keys>` for the environment `enumext*` “saved” in the *sequence* `{<store name>}`, use `\setenumext[<print , *>]{<keys>}`.

Example

```
\begin{enumext}[save-ans=sample,columns=1,show-pos=true,nosep,save-ref=true]
  \item Factor  $3x+3y+3z$ . \anskey{$3(x+y+z)$}
  \item True False

  \begin{enumext}[nosep]
    \item \LaTeXe is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

  \begin{enumext}[nosep]
    \item You use Linux? \anskey{Yes}
    \item Rate the following package and class
      \begin{enumext}[nosep]
        \item \texttt{xsim} \anskey{very good}
        \item \texttt{exsheets} \anskey{obsolete}
      \end{enumext}
    \end{enumext}
  \end{enumext}

  The answer to \ref{sample:4} is \getkeyans{sample:4} and the answers to
  all the worksheets are as follows:

  \printkeyans{sample}
```

1. Factor $3x + 3y + 3z$.

[1]

2. True False

(a) ~~LaTeX~~e is cool?

[2]

3. Related to Linux

(a) You use Linux?

[3]

(b) Rate the following package and class

i. `xsim`

[4]

ii. `exsheets`

[5]

The answer to 3.(b).i is very good and the answers to all the worksheets are as follows:

1. $3(x + y + z)$ ✖
2. (a) Very True! ✖
3. (a) Yes ✖
- (b) i. very good ✖
- ii. obsolete ✖

7 Full examples

Some adapted questions taken from [TeX-SX](#) are included as an example. The source files of the examples can be compiled with the excellent [arara](#)¹ tool.

Example 1

Adapted from the response given by Enrico Gregorio in [Squares for answer choice options and perfect alignment to mathematical answers](#). Source file [📄](#).

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

1. B
2. A
3. B
4. A

Example 2

Adapted from the response given by Florent Rougon in [Multiple choice questions with proposed answers in random order — addition of automatic correction \(cross mark\)](#). Source file [📄](#).

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

☒ A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

A

 36 km/h.

☒ B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

☒ A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

1. B
- ✖

 2. A

✖
3. B
- ✖

 4. A

✖

Example 3

A “simple multiple choice” test. Source file [📄](#).

1. First type of questions

A

 value

C

 value

B

 correct

D

 value
2. Second type of questions

I.

 $2\alpha + 2\delta = 90^\circ$

II.

 $\alpha = \delta$

III.

 $\angle EDF = 45^\circ$

¹The cool $\text{\TeX}$ automation tool: <https://www.ctan.org/pkg/arara>

- A

I only

B

II only

C

I and II only
- D

I and III only

E

I, II, and III

3. Third type of questions

(1) $2\alpha + 2\delta = 90^\circ$

(2) $\angle EDF = 45^\circ$

A

value

B

value

C

value

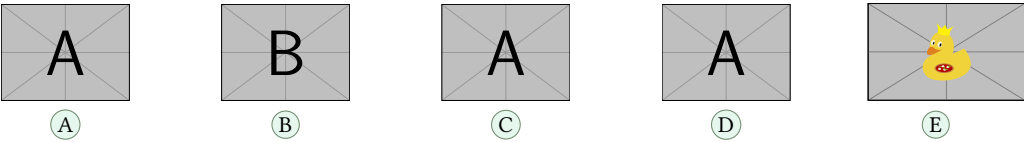
D

value

E

value

4. Question with image and label below:



5. Question with image on right side:
- A

value

B

value

C

value

D

correct

E

value



Test keys

1. B, $x = 5$

2. D

3. C, some note
- ⌘ 4. E, A duck

⌘ 5. D, other note

⌘
- ⌘

⌘

⌘

Example 4

A “simple worksheet” using ducks :) Source file

- Factor $x^2 - 2x + 1$
- Factor $3x + 3y + 3z$
- The following questions need to be quantified :)
- True False
- (a) $\alpha > \delta$

(b) ~~ETX~~ze is cool?
- Related to Linux
- (a) You use Linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

The answer to 1 is $(x - 1)^2$ and the answer to 3.(a) is False.

1. $(x - 1)^2$

2. $3(x + y + z)$

3. (a) False

(b) Very True!

4. (a) Yes
- ⌘ (b) Yes, dnf

⌘ (c) i. doesn't exist for now :(

⌘ ii. very good

⌘ iii. obsolete

⌘
- ⌘

⌘

⌘

Example 5

Adapted from the response given by Stephen in SAT like question format. Source file

1

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character

2

for pursuing an unexpected course of action.

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

- D) One character criticizes another character for pursuing an unexpected course of action.

3

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character

1. A)
2. C)
3. B)
4. D)

Example 6

Adapted from the response to Environment for enumerate environment. Source file .

- 8.5a, KSC 10. sample
- A sample
- ✓ B answer
- C sample
- D sample

- 9.5a, KSC 11. sample
- A sample
- B sample
- C sample
- ✓ D answer

12. sample
- A sample
- B answer
- C sample
- D sample

13. sample
- A sample
- B sample
- C sample
- D answer

10. B (8.5a, KSC)
11. D (9.5a, KSC)
12. B (10.5a, KSC)
13. D (11.5a, KSC)

8 Tagged PDF examples

This section is just to show the compatibility of enumext with tagged PDF using lualatex. The files described below are just for testing and are intended as examples and, in a way, to simplify the time of Matthew Bertucci (@mbertucci) when he sees this excellent package and adds it to The LaTeX Tagged PDF repository.

To compile the tests with lualatex-dev the packages multicol, unicode-math, geometry, graphicx, luaamll and hyperref are required along with the line:

```
\DocumentMetadata{lang = en-US, pdfversion = 2.0, pdfstandard = ua-2, tagging=on,}
```

- ◆ All examples have been checked using veraPDF together with ngpdf.
- The file enumext-01.tex contains the basic tests for the enumext and enumext* environments and the nesting between them plus the use of the label, labelwidth, labelsep, ref, align and wrap-label keys. Source file  and tagged PDF .

The file enumext-02.tex contains the tests for the enumext and enumext* environments and the support for minipage and multicol environments using the keys columns, columns-sep, mini-env, mini-right and \miniright command. Source file  and tagged PDF .

The file enumext-03.tex contains the tests for the enumext and keyanspic environments activated by the save-ans key together with the save-sep and save-ref keys and the \printkeyans command. Source file  and tagged PDF .

- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file [📄](#) and tagged PDF [📄](#).
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file [📄](#) and tagged PDF [📄](#).
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake itemize* and *description*. Source file [📄](#) and tagged PDF [📄](#).
- The file `enumext-07.tex` contains the tests for starting the environments with `\setenumext{resume}`, the `\resetenumext` command and the `series`, `resume`, `resume*` and `reset` keys. Source file [📄](#) and tagged PDF [📄](#).

9 The way of non-enumerated lists

It is possible to use (or abuse) the `enumext` and `enumext*` environments to mimic *non-enumerated* list environments such as `itemize` and `description`, clearly the `\keys` to “store answers”, the `keyans`, `keyans*` and `keyanspic` environments lose their sense and it is not the focus of `enumext` package, but, why not to do it?.

Other uses of the `enumext` environment that can be helpful for specific purposes are shown as examples. The *trick* to generate these “fake environments” is to set `label={}` or `label={\some}` and play with the `list-indent`, `list-offset`, `font` and `wrap-label` keys.

Fake itemize environment

The `label` key is set here using the default settings in $\TeX$ for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

- | | |
|---------------------|---------------------|
| • First level item | * First level item |
| – Second level item | ◊ Second level item |
| • Third level item | ◦ Third level item |
| · Fourth level item | ★ Fourth level item |
| • First level item | * First level item |

Fake description environment

Here `label={}` and `list-indent=2.5em`, `font=\bfseries` are set.

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Adding `list-indent=0pt` gives the *widest style*:

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

🔹 The small space at the beginning of the “unlabeled entry” corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.

🔹 When tagged PDF is active the default `description` style is NOT available due to the redefinition of `\makeLabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[content]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

Here `label={}` is set and a convenient value is given to `labelsep` and `labelwidth`, for example taking the *longest label* as reference and passing it as value using:

```
\newlength{\descitemwd}
\settowidth{\descitemwd}{\textbf{Something long}}
```

and then use `labelsep=4pt,labelwidth=\descitemwd,font=\bfseries`.

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

The environment can be translated so that the `<labels>` are on the left margin calculating the value passed to the `list-offset` key, in this case it will be equal to the sum of the values set by the `labelwidth` and `labelsep` keys finally resulting as `list-offset={-\descitemwd - 4pt}`.

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Adding `align=right` gives:

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

At this point `list-offset={-\descitemwd - 4pt}` instead of `list-offset={-\labelwidth - \labelsep}`, this is because the parameters `\labelwidth` and `\labelsep` take the default values, as if `label` had not been set.

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` and `wrap-label*` keys comes into play. Unlike the `enumitem` package, the `align` key only supports three options, so a command is created in the style `\parleft` of `enumitem` that allows placing *multiline labels* using `\parbox`.

```
\NewDocumentCommand \labelbx { s +m }
{%
  \SuspendTagging{\parbox}%
  \IfBooleanTF{#1}
  {%\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%
  {%\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%
  \ResumeTagging{\parbox}%
}
```

Now it is only necessary to set `wrap-label*={\labelbx{#1}}`.

SomeThing	A short one-line description. This is an entry <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
SoMeThInG	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum
LoNg	ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[12]` package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in [Create a fake label ref using list](#) and the answer given by David Carlisle in [Change the use of label ref by data save in an array \(list\)](#) I managed to create a more solid code than the original version, now using the `l3prop[13]` and `l3seq[13]` modules together with the `hyperref[10]` and `enumitem[5]` packages, which did the job, but with some shortcomings.

As time went by I took these shortcomings as a personal challenge which I called “*reinventing the wheel*”, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This “*reinventing the wheel*” finally ended up becoming `enumext`.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn't have in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with $\text{\LaTeX}$, that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it's something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that `enumext` serves this purpose and inspires more users and authors to follow this path.

Regression tests

The source repository includes a set of regression tests for `l3build`, kept in the `testfiles` directory. They are run with the Lua $\text{\TeX}$ engine and the development format `latex-dev`, and each test is compiled twice so that references and labels are resolved before the log is compared:

- `test-001` and `test-002`: lists without tagging, with the labels, the spacing, the nesting, the counters, the series and `enumext*`, and the independence between `\setenumext` and `\printkeyans`;
- `test-003`: the keys that set vertical and horizontal spaces, penalties, the symbol of `\item*`, the code keys `before`, `first` and `after`, and the `minipage` keys;
- `test-004`: the storage system with `\anskey`, `anskey*`, `\printkeyans`, `\getkeyans` and `\foreachkeyans`, and the keys that store and show the answers;
- `test-005`: the environments `keyans`, `keyans*` and `keyanspic`, their keys and the standard lists nested in the options of `keyans*`;
- `test-006`: the exact vertical spaces above, below and beside the `minipage` and `multicols` variants (`mini-env`, `mini-right`, `\miniright` and `multicols`) in `enumext`, `enumext*`, `keyans` and `keyans*`, measured with boxes, and the closing of the `minipage` when the content ends in a character;
- `test-010` and `test-011`: the same lists and the storage system with `tagging=on`, including the structure tree and the standard lists and the environments based on `trivlist` nested in the content of an item of `enumext*`;
- `test-012`: the structure tree with `tagging=on` of the `minipage` and `multicols` variants, where the position of the figures records the vertical spaces;
- `test-013`: the `base-fix` key, with the baseline of the nested list measured with boxes for different values of `parsep`, `itemsep` and `columns`;
- `test-014`: the vertical spaces of the environment `keyanspic` with the label below and above the figures, measured with boxes for the keys `layout-top`, `layout-sep` and `label-sep`;
- `test-015`: the standard lists (`itemize`, `enumerate` and `description`) nested in the content of an item of `enumext*`, measured with boxes for different values of `columns`, `item-break` and joined items;
- `test-016`: the values of `labelwidth` and `labelsep` inside the nested lists, and the environments `quote`, `quotation`, `verse`, `list`, `center`, `trivlist`, `tabbing`, `verbatim` and the theorems nested in the content of an item of `enumext*`, measured with boxes;
- `test-100`: the error and warning messages of the package.

From the root of the repository, `l3build check` runs all the tests and `l3build check test-001` runs a single one. After a deliberate change in the package, `l3build save test-001` regenerates the reference file `test-001.tlg`, which must be reviewed before committing it. The target `l3build testSE` compiles twice the documents in `sources/test-pkg`, validates the tagged structure and copies the PDF files that are attached to this manual; it is also run by `l3build release`.

10 References

- [1] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2025.
- [2] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [3] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [4] BERRY, KARL. “ $\text{\LaTeX}$ 2_ε: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2026.
- [5] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2025.
- [6] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2025.
- [7] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [8] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.
- [9] The $\text{\LaTeX}$ Project. “ $\text{\LaTeX}$ for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2026.
- [10] The $\text{\LaTeX}$ Project. “Extensive support for hypertext in $\text{\LaTeX}$ ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2026.
- [11] The $\text{\LaTeX}$ Project. “ $\text{\LaTeX}$ News, Issue 41, June 2025”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [12] The $\text{\LaTeX}$ Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2025.
- [13] The $\text{\LaTeX}$ Project. “The $\text{\LaTeX}$ 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2026.
- [14] The $\text{\LaTeX}$ Project. “The $\text{\LaTeX}$ 2_ε sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2026.
- [15] The $\text{\LaTeX}$ Project. “latex-lab – $\text{\LaTeX}$ laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2026.
- [16] “indent – Environments that change margins”. Available from CTAN, <https://www.ctan.org/pkg/indent>, 1993.
- [17] GONZÁLEZ, PABLO. “scontents - Stores $\text{\LaTeX}$ contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2026.
- [18] FISCHER, ULRIKE. “tagpdf – $\text{\LaTeX}$ kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2026.
- [19] MITTELBACH, FRANK. “ $\text{\LaTeX}$ ’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2026.
- [20] The $\text{\LaTeX}$ Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2026.

11 Change history

- v2.4 (ctan), 2026-11-01**
 - New key `item-break` for `enumext*` and `keyans*`: an item that occupies the whole row is typeset without `minipage` and can be broken between pages.
 - The standard lists and the environments `quote`, `quotation`, `verse`, `center`, `tabbing`, `verbatim`, those created with `\newtheorem` and similar ones can be used in the items of `enumext*` and `keyans*`.
 - Fixed the duplicate error messages when the `save-ans` key is missing, the behavior of `\setenumext` with invalid levels and its interaction with `\printkeyans`, and `widest` for the labels `\Roman*` and `\roman*`.
 - Fixed `\miniright` and `mini-env`, `mini-right` and `mini-right*`: spurious errors, exact vertical spaces and the end of the `minipage`. Fixed `base-fix` with `parsep`, `itemsep` and `columns`.
 - Tagged PDF: balanced paragraph hooks, `TextAlign` attribute in `item-break`, and the `minipage` on the right side no longer leaves the tagging state active.
 - Error and information messages with a common layout.
 - Added regression tests, run with `l3build check`.
- v2.3 (ctan), 2026-10-23**
 - Update requirement to $\text{\LaTeX}$ release 2026-11-01.
 - Cleanup warnings and details returned by `expltools`.
 - Replace `\exp_after:wN` by `\exp_after:w`.
 - Update code for *tagged* PDF.
 - Adjustments in the documentation.
- v2.2 (ctan), 2026-06-01**
 - Update requirement to $\text{\LaTeX}$ release 2026-06-01.
 - Move `\setcounter` to `\begin{list}{\langle arg one \rangle}{\langle arg two \rangle\setcounter}` for `enumext*`.
 - Adjustments in the documentation.
- v2.1 (ctan), 2026-01-15**
 - Fixed bad interaction between `\setenumext` and the `ref` key.
- v2.0 (ctan), 2025-12-02**
 - Update requirement to `multicol` release 2025-10-21.
 - Adjustments in the documentation.
- v1.9 (ctan), 2025-11-01**
 - Update requirement to $\text{\LaTeX}$ release 2025-11-01.
 - Move `\setcounter` to `\begin{list}{\langle arg one \rangle}{\langle arg two \rangle\setcounter}`.
 - Update code for *tagged* PDF.
- v1.8 (ctan), 2025-10-04**
 - Replacing `\scantokens` (primitive) with `\tl_retokenize:n`.
 - Cleanup warnings and details returned by `expltools`.
- v1.7 (ctan), 2025-07-10**
 - Fixed `\setenumext{enumext*}{resume}`.
 - Fixed bad interaction between `\setenumext` and the `resume` key.
 - The behavior of the key `resume*` has been updated and documented.
- v1.6 (ctan), 2025-07-04**
 - Syntax simplification for `\setenumextmeta`.
 - Environments can be started with the key `resume` *without value*.
 - Add `\resetenumext`, `reset` and `reset*` keys.
 - The `resume`, `resume*` and `series` keys can now be set per level.
 - Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.
- v1.5 (ctan), 2025-06-11**
 - Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
 - Add keys `beginpenalty`, `midpenalty` and `endpenalty`.
- v1.4 (ctan), 2025-06-09**
 - Improved implementation of the `start` key for *tagged* PDF.
 - Improved implementation of the `ref` key.
 - Fixed the behavior of the `save-sep` key.
 - Fixed the behavior of the `resume*` key.
- v1.3 (ctan), 2025-06-01**
 - Removed dependency on the `scontents` package.
 - The `anskey*` environment has been rewritten using the new `c`-type argument.
- v1.2 (ctan), 2025-03-28**
 - Replace signature (prevent expansion for optional argument).
 - Solve Inconsistent local/global assignment.
- v1.1 (ctan), 2024-11-14**
 - Fixed implementation for `font` and `base-fix` keys.
 - Added new keys for symbol marks.
 - Update and improvements in the internal code.
 - Adjustments in the documentation.
- v1.0 (ctan), 2024-11-01**
 - First public release.

12 Index of Documentation

The italic numbers denote the pages where the corresponding entry is described.

C

Document class :

article 2

book 2

exam 2

letter 2

report 2

Commands provided by enumext:

\anskey 16

\anspic 19

\foreachkeyans 21

\getkeyans 20

\item* 5, 6, 18

\item 5, 6

\miniright 13

\printkeyans 21

\resetenumext 12

\setenumextmeta 7

\setenumext 7

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

D

Dimension defined by enumext:

\itemwidth 4

E

Environments :

minipage 28

multicols 28

Environments provided by enumext:

anskey* 16

enumext* 4

enumext 4

keyans* 17

keyanspic 19

keyans 17

I

\itemwidth 4

K

Keys for \anskey provided by enumext:

break-col 16

item-join 16

item-pos* 16

item-star 16

item-sym* 16

Keys for \foreachkeyans provided by enumext:

after 21

before 21

sep 21

start 21

step 21

stop 21

wrapper 21

Keys for anskey* provided by enumext:

break-col 16

force-eol 17

item-join 16

item-pos* 16

item-star 16

item-sym* 16

overwrite 17

write-env 17

Keys for environments provided by enumext:

above* 10

above 10

after 11

align 8

base-fix 10

before* 11

before 11

beginpenalty 9

below* 10

below 10

check-ans 15

columns-sep 13

columns 12

endpenalty 9

first 11

font 8

item-break 5, 6, 28

item-pos* 6

item-sym* 6

itemindent 10

itemsep 10

label-pos 19

label-sep 19, 28

labelsep 8

labelwidth 8

label 8

layout-sep 19, 28

layout-sty 19

layout-top 19, 28

list-indent 10

list-offset 10

listparindent 11

mark-ans* 15

mark-ans 15

mark-pos* 15

mark-pos 15

mark-ref 14

mark-sep* 15

mark-sep 15

midpenalty 9

mini-env 13

mini-right* 13

mini-right 13

mini-sep 13

mode-box 8

no-store 14

noitemsep 10

nosep 10

L
Labels provided by `enumext`:
`\Alph*` 8

enumerate	27
enumitem	3
fancyvrb	17
footnotehyper	5
geometry	25
graphicx	25
hyperref	5
l3keys	7
l3prop	27
l3seq	27
luamml	25
multicol	1
scontents	30
shortlst	5
tasks	5
unicode-math	25
xsim	2

13 Implementation

The most recent publicly released version of `enumext` is available at CTAN: <https://www.ctan.org/pkg/enumext>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/enumext/issues>.

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a TeXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

13.1 General conventions

Variables containing `i`, `ii`, `iii` and `iv` are associated by level with the `enumext` environment, variables containing `v` are associated with the `keyans` environment, variables containing `vi` are associated with the `keyanspic` environment, variables containing `vii` are associated with the `enumext*` environment and variables containing `viii` are associated with the `keyans*` environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital “X”.

The temporary function `\__enumext_tmp:n` is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

13.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

Identify the internal prefix (L^AT_EX3 DocStrip convention) for `l3doc` class.

```
2 (@@=enumext)
```

13.3 Declaration of the package

Checks for a minimum release version of L^AT_EX; otherwise it returns an “error”.

```
3 \NeedsTeXFormat{LaTeX2e}[2026-11-01]
4 \IfFormatAtLeastF { 2026-11-01 }
5 {
6   \PackageError { enumext }
7     { LaTeX~kernel~too~old~need~2026-11-01~or~later }
8   {
9     enumext~requires~a~LaTeX2e~release~2026-11-01~or~
10    later.~Update~your~TeX~distribution.
11  }
12 }
```

Now declare the `enumext` package.

```
13 \ProvidesExplPackage {enumext} {2026-11-01} {2.4} {Enumerate exercise sheets}
```

Finally, check if the `multicol` package is loaded; if not, it is loaded.

```
14 \hook_gput_code:nnn {begindocument} {enumext}
15 {
16   \IfPackageLoadedTF { multicol }
17   {
18     \msg_info:nnn { enumext } { package-load } { multicol }
19   }
20   {
21     \msg_info:nnn { enumext } { package-not-load } { multicol }
22     \RequirePackage{multicol}[2025-10-21]
23   }
24 }
```

13.4 Definition of variables

This section describes the variables shared by several commands, keys or environments, in groups and by purpose. The variables that belong to a single command, key or environment are defined in the subsection where they are used, in the part “*Definition of variables*”. The variables that do not appear in either place are created by means of `\keys_define:nn` or some function described below.

13.4.1 Nesting levels and state of the environments

Integer variables will control the nesting levels of the environments.

```
\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

25 \int_new:N \l__enumext_level_int
26 \int_new:N \l__enumext_level_h_int
27 \int_new:N \l__enumext_keyans_level_int
28 \int_new:N \l__enumext_keyans_level_h_int
29 \int_new:N \l__enumext_keyans_pic_level_int
```

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

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_name_and_start:` (§13.5.3).

```
\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standard_bool
\g__enumext_standard_bool
\l__enumext_standard_first_bool
\l__enumext_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
\l__enumext_envir_name_tl

30 \bool_new:N \l__enumext_starred_bool
31 \bool_new:N \g__enumext_starred_bool
32 \bool_new:N \l__enumext_starred_first_bool
33 \bool_new:N \l__enumext_standard_bool
34 \bool_new:N \g__enumext_standard_bool
35 \bool_new:N \l__enumext_standard_first_bool
36 \bool_new:N \l__enumext_keyans_env_bool
37 \tl_new:N \g__enumext_start_line_tl
38 \tl_new:N \g__enumext_envir_name_tl
39 \tl_new:N \l__enumext_envir_name_tl
```

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

13.4.2 Counters, labels and numbering

The dimension `\l__enumext_current_widest_dim` stores the current label width. It is used by `widest` (§13.15) and `label` (§13.13) keys.

```
\l__enumext_current_widest_dim

40 \dim_new:N \l__enumext_current_widest_dim
```

(End of definition for `\l__enumext_current_widest_dim`.)

Internal variables shared by the `start` and `start*` keys (§13.15), the `resume`, `resume*` and `series` keys (§13.27) and the environments.

```
\l__enumext_resume_count_bool
\l__enumext_resume_count_X_bool

41 \bool_new:N \l__enumext_resume_count_bool
42 \cs_set_protected:Nn \__enumext_tmp:n
43 {
44   \bool_new:c { l__enumext_resume_count_ #1 _bool }
45 }
46 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_resume_count_bool` and `\l__enumext_resume_count_X_bool`.)

The integer `\l__enumext_start_X_int` is used by the `start` and `start*` keys (§13.15) and the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.19.1).

```
\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl

47 \cs_set_protected:Nn \__enumext_tmp:n
48 {
49   \int_new:c { l__enumext_start_ #1 _int }
50   \tl_new:c { l__enumext_fake_item_indent_ #1 _tl }
51 }
52 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_start_X_int` and `\l__enumext_fake_item_indent_X_tl`.)

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13). The variables `\l__enumext_label_fill_left_X_tl`, `\l__enumext_label_fill_right_X_tl` and `\l__enumext_align_label_pos_X_str` are used by the `align` key (§13.13).

```
\l__enumext_wrap_label_X_bool
\l__enumext_wrap_label_opt_X_bool
\l__enumext_label_fill_left_X_tl
\l__enumext_label_fill_right_X_tl
\l__enumext_align_label_pos_X_str

53 \cs_set_protected:Nn \__enumext_tmp:n
54 {
55   \bool_new:c { l__enumext_wrap_label_ #1 _bool }
56   \bool_new:c { l__enumext_wrap_label_opt_ #1 _bool }
57   \tl_new:c { l__enumext_label_fill_left_ #1 _tl }
58   \tl_new:c { l__enumext_label_fill_right_ #1 _tl }
59 }
60 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

```

61 \cs_set_protected:Nn \__enumext_tmp:n
62 {
63   \str_new:c { l__enumext_align_label_pos_ #1 _str }
64 }
65 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

13.4.3 Horizontal and vertical spaces

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.19). The variable `\l__enumext_leftmargin_X_dim` is used and set by the function `\__enumext_calc_hspace:NNNNNNN` (§13.42.4).

```

66 \cs_set_protected:Nn \__enumext_tmp:n
67 {
68   \bool_new:c { l__enumext_leftmargin_tmp_ #1 _bool }
69   \dim_new:c { l__enumext_leftmargin_tmp_ #1 _dim }
70   \dim_new:c { l__enumext_leftmargin_ #1 _dim }
71 }
72 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_leftmargin_tmp_X_bool`, `\l__enumext_leftmargin_tmp_X_dim`, and `\l__enumext_leftmargin_X_dim`.)

Internal variables used by `columns` key (§13.24).

```

73 \cs_set_protected:Nn \__enumext_tmp:n
74 {
75   \skip_new:c { l__enumext_multicols_above_ #1 _skip }
76   \skip_new:c { l__enumext_multicols_below_ #1 _skip }
77 }
78 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip`.)

Internal variables used by `\miniright` command (§13.25.7) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.23, §13.25).

```

79 \int_new:N \g__enumext_minipage_stat_int
80 \skip_new:N \g__enumext_minipage_right_skip
81 \cs_set_protected:Nn \__enumext_tmp:n
82 {
83   \dim_new:c { l__enumext_minipage_left_ #1 _dim }
84   \bool_new:c { l__enumext_minipage_active_ #1 _bool }
85 }
86 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

Internal variables used by `enumext*` and `keyans*` environments. The boolean `\l__enumext_item_break_X_bool` is set by the key `item-break` (§13.21).

```

87 \cs_set_protected:Nn \__enumext_tmp:n
88 {
89   \bool_new:c { l__enumext_item_starred_ #1 _bool }
90   \bool_new:c { l__enumext_item_break_ #1 _bool }
91   \int_new:c { l__enumext_item_column_pos_ #1 _int }
92   \int_new:c { g__enumext_item_count_all_ #1 _int }
93   \box_new:c { l__enumext_item_text_ #1 _box }
94   \dim_new:c { l__enumext_joined_width_ #1 _dim }
95   \tl_new:c { g__enumext_item_symbol_aux_ #1 _tl }
96   \str_new:c { l__enumext_align_label_ #1 _str }
97   \bool_new:c { g__enumext_minipage_center_ #1 _bool }
98   \dim_new:c { g__enumext_minipage_right_ #1 _dim }
99 }
100 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

13.4.4 Storing answers, checking and printing

```
\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_current_label_tl
\l__enumext_store_current_opt_arg_tl
```

The variable `\l__enumext_store_active_bool` set by `save-ans` key (§13.30.3) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` saves the `{⟨store name⟩}` set by the `save-ans` key of the *sequence* and *prop list* where the content is stored. The variable `\g__enumext_store_name_tl` is just a global copy of `{⟨store name⟩}` used by different functions.

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.41) and `\anspic*` (§13.46.4) for the `keyans`, `keyans*` and `keyanspic` environments.

```
101 \bool_new:N \l__enumext_store_active_bool
102 \tl_new:N \l__enumext_store_name_tl
103 \tl_new:N \g__enumext_store_name_tl
104 \tl_new:N \l__enumext_store_current_label_tl
105 \tl_new:N \l__enumext_store_current_opt_arg_tl
```

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

```
\l__enumext_check_answers_bool
\g__enumext_check_ans_key_bool
\l__enumext_check_start_line_env_tl
\l__enumext_item_wrap_key_bool
\g__enumext_check_starred_cmd_int
\g__enumext_item_anskey_int
\g__enumext_item_number_int
\l__enumext_item_number_bool
\g__enumext_item_answer_diff_int
```

Internal variables used by “*internal check answer*” mechanism (§13.30.5) used by the `check-ans`, `no-store`, `wrap-ans*` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
106 \bool_new:N \l__enumext_check_answers_bool
107 \bool_new:N \g__enumext_check_ans_key_bool
108 \tl_new:N \l__enumext_check_start_line_env_tl
109 \bool_new:N \l__enumext_item_wrap_key_bool
110 \int_new:N \g__enumext_check_starred_cmd_int
111 \int_new:N \g__enumext_item_anskey_int
112 \int_new:N \g__enumext_item_number_int
113 \bool_new:N \l__enumext_item_number_bool
114 \int_new:N \g__enumext_item_answer_diff_int
```

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

```
\l__enumext_print_keyans_star_bool
\l__enumext_print_keyans_cmd_bool
\l__enumext_print_keyans_X_tl
\l__enumext_mark_position_str
\l__enumext_mark_position_v_str
\l__enumext_mark_position_viii_str
```

The boolean `\l__enumext_print_keyans_cmd_bool` is true while the command `\printkeyans` (§13.51) is running and the environments consult it, the boolean `\l__enumext_print_keyans_star_bool` is true when the command receives the *starred* argument and the token lists `\l__enumext_print_keyans_X_tl` store the keys passed to each environment. The strings `\l__enumext_mark_position_str`, `\l__enumext_mark_position_v_str` and `\l__enumext_mark_position_viii_str` store the position of the marks set by the `mark-pos` key (§13.31).

```
115 \bool_new:N \l__enumext_print_keyans_star_bool
116 \bool_new:N \l__enumext_print_keyans_cmd_bool
117 \str_new:N \l__enumext_mark_position_str
118 \str_new:N \l__enumext_mark_position_v_str
119 \str_new:N \l__enumext_mark_position_viii_str
120 \cs_set_protected:Nn \__enumext_tmp:n
121 {
122   \tl_new:c { l__enumext_print_keyans_ #1 _tl }
123 }
124 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

13.4.5 hyperref, labels and references

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or loaded in memory (§13.7). The boolean variable `\l__enumext_footnotes_key_bool` determines if `hyperref` is loaded with the key `hyperfootnotes=true`.

```
125 \bool_new:N \l__enumext_hyperref_bool
126 \bool_new:N \l__enumext_footnotes_key_bool
```

(End of definition for `\l__enumext_hyperref_bool` and `\l__enumext_footnotes_key_bool`.)

```
\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl
```

Internal variables used by `save-ref` key (§13.31). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the `⟨labels⟩` defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `\__enumext_newlabel:nn` (§13.7) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
127 \tl_new:N \l__enumext_newlabel_arg_one_tl
```

```

128 \tl_new:N \l__enumext_newlabel_arg_two_tl
129 \tl_new:N \l__enumext_write_aux_file_tl
130 \cs_set_protected:Nn \__enumext_tmp:n
131 {
132   \tl_new:c { l__enumext_label_copy_ #1 _tl }
133 }
134 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

13.4.6 Constants

`\c__enumext_all_envs_clist` An internal `clist-var` variable to run with `\__enumext_tmp:n`.

```

135 \clist_const:Nn \c__enumext_all_envs_clist
136 {
137   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
138   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
139 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

13.5.1 Definition of messages for the internal hooks

These messages are not error messages, they only write information to the `.log` file with the standard layout of the information messages.

`prop-seq-int-hook` Messages written to the `.log` file by the internal hooks that show the values of the global variables of the storage.

`item-answer-hook`

```

140 \msg_new:nnn { enumext } { prop-seq-int-hook }
141 {
142   Storage~'#1':~
143   \c_backslash_str g__enumext_ #1 _prop~has~#2~elements,~
144   \c_backslash_str g__enumext_ #1 _seq~has~#3~elements,~
145   \c_backslash_str g__enumext_resume_ #1 _int~==~#4.
146 }
147 \msg_new:nnn { enumext } { item-answer-hook }
148 {
149   Items~==~#1,~answers~==~#2,~difference~==~#3.
150 }

```

(End of definition for `prop-seq-int-hook` and `item-answer-hook`.)

13.5.2 General utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§13.51) and `\foreachkeyans` command (§13.54).

`\seq_use:NV`

```

151 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
152 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

(End of definition for `\keys_precompile:neN` and `\seq_use:NV`.)

`\__enumext_at_begin_document:n` A internal “hook” function used for copying plain `list` and `minipage` environments definition and `hyperref` detection.

```

153 \cs_new_protected:Nn \__enumext_at_begin_document:n
154 {
155   \hook_gput_code:nnn {begindocument} {enumext} {#1}
156 }

```

(End of definition for `\__enumext_at_begin_document:n`.)

`\__enumext_after_env:nn` An internal “hook” function to execute code `mini-right` and `mini-right*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

```

157 \cs_new_protected:Nn \__enumext_after_env:nn
158 {
159   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
160 }

```

(End of definition for `\__enumext_after_env:nn`.)

`\__enumext_level:` Function for check current level in `enumext`.

```

161 \cs_new:Nn \__enumext_level:
162 {
163   \int_to_roman:n { \__enumext_level_int }
164 }

```

(End of definition for `\__enumext_level:`.)

`\__enumext_if_is_int:nT` A conditional function to determine whether the variable being passed is an integer used by `start` and `widest`
`\__enumext_if_is_int:nF` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function](#)
`\__enumext_if_is_int:nTF` [argument is an integer expression?](#).

```

165 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
166 {
167   \regex_if_match:nnTF { ^[\+|-]?[\d]+\$ } {#1} % $
168   { \prg_return_true: }
169   { \prg_return_false: }
170 }

```

(End of definition for `\__enumext_if_is_int:nT`, `\__enumext_if_is_int:nF`, and `\__enumext_if_is_int:nTF`.)

`\__enumext_show_length:nnn` Internal function used by `show-length` key to show “all lengths” calculated and used in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

171 \cs_new:Nn \__enumext_show_length:nnn
172 {
173   *~#2
174   \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
175   =~\use:c { #1 _use:c } { \__enumext_ #2 _ #3 _ #1 } \
176 }

```

(End of definition for `\__enumext_show_length:nnn`.)

`\__enumext_unskip_unkern:` The function `\__enumext_unskip_unkern:` will remove the last `⟨skip⟩` or `⟨kern⟩` at execution time using the values `11` and `12` of `\lastnodetype` to apply `\unskip` or `\unkern` depending on the case. In the *outer* vertical mode $\text{T}_{\text{E}}\text{X}$ can’t take things from the current page, because `\unskip` and `\unkern` are errors there, so a negative `\vskip` or `\kern` is used instead, as $\text{E}_{\text{T}}\text{X}$ does in `\removeatlastskip`.

```

177 \cs_new_protected:Nn \__enumext_unskip_unkern:
178 {
179   \int_case:nn { \lastnodetype }
180   {
181     { 11 }
182     {
183       \mode_if_vertical:TF
184       {
185         \mode_if_inner:TF
186         { \unskip }
187         { \skip_vertical:n { -\lastskip } }
188       }
189       { \unskip }
190     }
191     { 12 }
192     {
193       \mode_if_vertical:TF
194       {
195         \mode_if_inner:TF
196         { \unkern }
197         { \kern -\lastkern \scan_stop: }
198       }
199       { \unkern }
200     }
201   }
202 }

```

(End of definition for `\__enumext_unskip_unkern:`.)

13.5.3 Utilities for environments and levels

`\__enumext_is_not_nested:` The function `\__enumext_is_not_nested:` sets the variables `\g__enumext_standard_bool` and
`\__enumext_is_on_first_level:` `\g__enumext_starred_bool` to true only if the environments `enumext` and `enumext*` are NOT nested in each other and saves the environment name in `\l__enumext_envir_name_tl`.

```

203 \cs_new_protected:Nn \__enumext_is_not_nested:

```

```

204 {
205   \str_case:en { \@currenvir }
206   {
207     {enumext}
208     {
209       \tl_set:Nx \l__enumext_envir_name_tl { enumext }
210       \bool_lazy_and:nnT
211         { \bool_not_p:n { \g__enumext_standard_bool } }
212         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
213         {
214           \bool_gset_true:N \g__enumext_standard_bool
215         }
216     }
217     {enumext*}
218     {
219       \tl_set:Nx \l__enumext_envir_name_tl { enumext* }
220       \bool_lazy_and:nnT
221         { \bool_not_p:n { \g__enumext_starred_bool } }
222         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
223         {
224           \bool_gset_true:N \g__enumext_starred_bool
225         }
226     }
227   }
228 }

```

The function `\__enumext_is_on_first_level:` sets the variables `\l__enumext_standard_first_bool` (§13.30.3), `\l__enumext_starred_first_bool` (§13.30.3) to `true` only if the environment is not nested and it is at the “first level”. They also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name* of each environment in the variable `\g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

229 \cs_new_protected:Nn \__enumext_is_on_first_level:
230 {
231   \bool_lazy_all:nT
232   {
233     { \bool_if_p:N \g__enumext_standard_bool }
234     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
235     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
236   }
237   {
238     \bool_set_true:N \l__enumext_standard_first_bool
239     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
240     \tl_gset:Nx \g__enumext_start_line_tl
241       {
242         on~line~\exp_not:V \inputlineno
243       }
244   }
245   \bool_lazy_all:nT
246   {
247     { \bool_if_p:N \g__enumext_starred_bool }
248     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
249     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
250   }
251   {
252     \bool_set_true:N \l__enumext_starred_first_bool
253     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
254     \tl_gset:Nx \g__enumext_start_line_tl
255       {
256         on~line~\exp_not:V \inputlineno
257       }
258   }
259 }

```

(End of definition for `\__enumext_is_not_nested:` and `\__enumext_is_on_first_level:`.)

`\__enumext_keyans_name_and_start:`

The function `\__enumext_keyans_name_and_start:` will save the start line number and name of the environments `keyans`, `keyans*` and `keyanspic` in the variables `\l__enumext_check_start_line_env_tl` and `\l__enumext_envir_name_tl` to use in the `\__enumext_check_starred_cmd:n` function.

```

260 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
261 {

```

```

262     \str_case:en { \@currentvir }
263     {
264         {keyans}
265         {
266             \tl_set:Nc \l__enumext_envir_name_tl { keyans }
267             \tl_set:Nc \l__enumext_check_start_line_env_tl
268                 {
269                     in~'keyans'~start~on~line~\exp_not:V \inputlineno
270                 }
271         }
272     {keyans*}
273     {
274         \tl_set:Nc \l__enumext_envir_name_tl { keyans* }
275         \tl_set:Nc \l__enumext_check_start_line_env_tl
276             {
277                 in~'keyans*'~start~on~line~\exp_not:V \inputlineno
278             }
279     }
280 {keyanspic}
281 {
282     \tl_set:Nc \l__enumext_envir_name_tl { keyanspic }
283     \tl_set:Nc \l__enumext_check_start_line_env_tl
284         {
285             in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
286         }
287 }
288 }
289 }

```

(End of definition for `\__enumext_keyans_name_and_start:`.)

13.5.4 Utilities for log and terminal

The function `\__enumext_reset_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and will return the global variables to their default values after being used.

```

290 \cs_new_protected:Nn \__enumext_reset_global_vars:
291 {
292     \__enumext_reset_global_int:
293     \__enumext_reset_global_bool:
294     \__enumext_reset_global_tl:
295 }
296 \cs_new_protected:Nn \__enumext_reset_global_int:
297 {
298     \int_gzero:N \g__enumext_item_number_int
299     \int_gzero:N \g__enumext_item_anskey_int
300     \int_gzero:N \g__enumext_item_answer_diff_int
301 }
302 \cs_new_protected:Nn \__enumext_reset_global_bool:
303 {
304     \bool_gset_false:N \g__enumext_check_ans_key_bool
305     \bool_gset_false:N \g__enumext_standard_bool
306     \bool_gset_false:N \g__enumext_starred_bool
307 }
308 \cs_new_protected:Nn \__enumext_reset_global_tl:
309 {
310     \tl_gclear:N \g__enumext_store_name_tl
311     \tl_gclear:N \g__enumext_start_line_tl
312     \tl_gclear:N \g__enumext_envir_name_tl
313 }

```

(End of definition for `\__enumext_reset_global_vars:` and others.)

The function `\__enumext_log_global_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of elements saved in the *prop list* and *sequence* created by the `save-ans` key along with the value of the integer variable created for the `resume` key.

```

314 \cs_new_protected:Nn \__enumext_log_global_vars:
315 {
316     \msg_info:nneeee { enumext } { prop-seq-int-hook }
317     { \g__enumext_store_name_tl }
318     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
319     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }

```

```

320     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
321   }

```

The function `\__enumext_log_answer_vars:` will be passed to the function `\__enumext_execute_after_env:` and write to the `.log` file the number of items and answers along with the difference between them.

```

322 \cs_new_protected:Nn \__enumext_log_answer_vars:
323 {
324   \msg_info:nneee { enumext } { item-answer-hook }
325   { \int_use:N \g__enumext_item_number_int }
326   { \int_use:N \g__enumext_item_anskey_int }
327   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
328 }

```

(End of definition for `\__enumext_log_global_vars:` and `\__enumext_log_answer_vars:`.)

13.6 Copying list and minipage environments

The `list` environment provided by $\text{\LaTeX}$ has the following plain form[14]:

```

\list{⟨arg one⟩}{⟨arg two⟩}
\item[⟨opt⟩]
\endlist

```

And `minipage` environment provided by $\text{\LaTeX}$ has the following (simplified) plain form:

```

\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
⟨internal implement⟩
\endminipage

```

As a precaution they are copied using `\__enumext_at_begin_document:n` in case any package redefines the `list` environment or a related command.

◆ For compatibility with *tagged* PDF, `\NewCommandCopy` should be used and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `ltxcmd` (see `latex-lab-block`[15]).

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w
\__enumext_minipage:w
\__enumext_endminipage:

```

The functions `\__enumext_start_list:nn` and `\__enumext_stop_list:` correspond to copies of `\list` and `\endlist` from plain definition of `list` environment, the function `\__enumext_item_std:w` is a copy of the `\item` command.

```

329 \__enumext_at_begin_document:n
330 {
331   \cs_new_eq:NN \__enumext_start_list:nn \list
332   \cs_new_eq:NN \__enumext_stop_list: \endlist
333   \NewCommandCopy \__enumext_item_std:w \item
334 }

```

The functions `\__enumext_minipage:w` and `\__enumext_endminipage:` correspond to copies of `\minipage` and `\endminipage` from plain definition of `minipage` environment.

```

335 \__enumext_at_begin_document:n
336 {
337   \cs_new_eq:NN \__enumext_minipage:w \minipage
338   \cs_new_eq:NN \__enumext_endminipage: \endminipage
339 }

```

(End of definition for `\__enumext_start_list:nn` and others.)

13.7 Compatibility with hyperref and footnotehyper

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

First, the necessary rules are defined using “hooks” to determine if the `hyperref` package is loaded.

```

340 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
341 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

The function `\__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to `true` if the package is loaded. The public macro `\IfHyperBoolean` determines if the `hyperfootnotes=true` key is present, and if so, the boolean variable `\l__enumext_footnotes_key_bool` is set to `true`.

```

342 \cs_new_protected:Nn \__enumext_after_hyperref:
343 {
344   \IfPackageLoadedT { hyperref }
345   {
346     \msg_info:nnn { enumext } { package-load } { hyperref }
347     \bool_set_true:N \l__enumext_hyperref_bool

```

```

348     \IfHyperBoolean{hyperfootnotes}
349     {
350         \bool_set_true:N \l__enumext_footnotes_key_bool
351     }
352     { }
353 }

```

If the variable `\l__enumext_footnotes_key_bool` is true, it checks whether the package `footnotehyper` is loaded; if it is not, `\l__enumext_footnotes_key_bool` is set to false and `\footnote` is redefined.

```

354 \bool_if:NT \l__enumext_footnotes_key_bool
355 {
356     \IfPackageLoadedTF { footnotehyper }
357     {
358         \msg_info:nnn { enumext } { package-load } { footnotehyper }
359     }
360     {
361         \bool_set_false:N \l__enumext_footnotes_key_bool
362     }
363 }

```

The functions `\__enumext_hypertarget:nn` and `\__enumext_phantomsection:` correspond to the internal copies of `\hypertarget` and `\phantomsection`. If the boolean variable `\l__enumext_hyperref_bool` is false the functions `\__enumext_hypertarget:nn` and `\__enumext_phantomsection:` will be disabled.

```

364 \bool_if:NTF \l__enumext_hyperref_bool
365 {
366     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
367     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
368 }
369 {
370     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
371     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
372 }
373 }

```

(End of definition for `\__enumext_after_hyperref:`, `\__enumext_hypertarget:nn`, and `\__enumext_phantomsection:`.)

`\__enumext_newlabel:nn`

The function `\__enumext_newlabel:nn` writes the information to the `.aux` file when using the `save-ref` key. The arguments taken by the function are:

```

#1: \l__enumext_newlabel_arg_one_tl
#2: \l__enumext_newlabel_arg_two_tl

```

🔗 The trick here is to manage the number of arguments passed to `\newlabel{#1}{#2}` according to the presence of the `hyperref` package.

```

374 \cs_new_protected:Nn \__enumext_newlabel:nn
375 {
376     \protected@write \@auxout { }
377     {
378         \token_to_str:N \newlabel {#1}
379         {
380             {#2}
381             \bool_if:NT \l__enumext_hyperref_bool
382             { { \thepage } {#2} {#1} }
383             { }
384         }
385     }
386     \__enumext_hypertarget:nn {#1} { }
387     \__enumext_phantomsection:
388 }

```

(End of definition for `\__enumext_newlabel:nn`.)

13.8 Internal redefining `\footnote` command

To keep the correct numbering of `\footnote` and to make it work correctly in the `enumext*` and `keyans*` environments and `mini-env` key it is necessary to redefine the `\footnote` command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in `footnotes in boxes compatible with hyperref`.

```

\g__enumext_footnote_standard_int
\g__enumext_footnote_starred_int
\g__enumext_footnote_standard_arg_seq
\g__enumext_footnote_starred_arg_seq

```

Internal variables used for redefinition of `\footnote` (§13.8).

```

389 \int_new:N \g__enumext_footnote_standard_int

```

```

390 \int_new:N \g__enumext_footnote_starred_int
391 \seq_new:N \g__enumext_footnote_standard_arg_seq
392 \seq_new:N \g__enumext_footnote_starred_arg_seq
393 \seq_new:N \g__enumext_footnote_standard_int_seq
394 \seq_new:N \g__enumext_footnote_starred_int_seq

```

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

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:
  \__enumext_renew_footnote_mini:
  \__enumext_print_footnote_mini:

```

Redefinition of the `\footnote` command using `\footnotetext` and `\footnotemark` for the `mini-env` key in the `enumext` and `keyans` environments.

```

395 \cs_new_protected:Nn \__enumext_footnotetext:nn
396 {
397   \footnotetext[#1]{#2}
398 }
399 \cs_new_protected:Nn \__enumext_renew_footnote:
400 {
401   \RenewDocumentCommand \footnote { o +m }
402   {
403     \tl_if_novalue:nTF {##1}
404     {
405       \stepcounter{footnote}
406       \int_gset_eq:Nc \g__enumext_footnote_standard_int { c@footnote }
407     }
408     {
409       \int_gset:Nn \g__enumext_footnote_standard_int {##1}
410     }
411     \footnotemark [ \g__enumext_footnote_standard_int ]
412     \seq_gput_right:Nn \g__enumext_footnote_standard_arg_seq {##2}
413     \seq_gput_right:NV
414       \g__enumext_footnote_standard_int_seq \g__enumext_footnote_standard_int
415   }
416 }
417 \cs_new_protected:Nn \__enumext_print_footnote:
418 {
419   \seq_if_empty:NF \g__enumext_footnote_standard_int_seq
420   {
421     \seq_map_pairwise_function:NNN
422       \g__enumext_footnote_standard_int_seq
423       \g__enumext_footnote_standard_arg_seq
424       \__enumext_footnotetext:nn
425   }
426   \seq_gclear:N \g__enumext_footnote_standard_arg_seq
427   \seq_gclear:N \g__enumext_footnote_standard_int_seq
428 }

```

The `enumext*` and `keyans*` environments are implemented using `minipage`, so `\footnote` must also be redefined to keep this numbering as if it were part of the document.

```

429 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
430 {
431   \RenewDocumentCommand \footnote { o +m }
432   {
433     \tl_if_novalue:nTF {##1}
434     {
435       \stepcounter{footnote}
436       \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
437     }
438     {
439       \int_gset:Nn \g__enumext_footnote_starred_int {##1}
440     }
441     \footnotemark [ \g__enumext_footnote_starred_int ]
442     \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq {##2}
443     \seq_gput_right:NV
444       \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
445   }
446 }
447 \cs_new_protected:Nn \__enumext_print_footnote_mini:
448 {
449   \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
450   {
451     \seq_map_pairwise_function:NNN
452       \g__enumext_footnote_starred_int_seq

```

```

453         \g__enumext_footnote_starred_arg_seq
454         \__enumext_footnotetext:nn
455     }
456     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
457     \seq_gclear:N \g__enumext_footnote_starred_int_seq
458 }

```

(End of definition for `\__enumext_footnotetext:nn` and others.)

```

\__enumext_renew_footnote_standard:
\__enumext_print_footnote_standard:
\__enumext_renew_footnote_starred:
\__enumext_print_footnote_starred:

```

The redefinition of `\footnote` is encapsulated to pass it to the internal `\__enumext_mini_page` environment used by the `mini-env` key in the `enumext` and `keyans` environments. It is executed when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

459 \cs_new_protected:Nn \__enumext_renew_footnote_standard:
460 {
461     \bool_if:NT \g__enumext_standard_bool
462     {
463         \IfDocumentMetadataTF
464         {
465             \__enumext_renew_footnote:
466         }
467         {
468             \bool_if:NF \l__enumext_footnotes_key_bool
469             {
470                 \__enumext_renew_footnote:
471             }
472         }
473     }
474 }
475 \cs_new_protected:Nn \__enumext_print_footnote_standard:
476 {
477     \bool_if:NT \g__enumext_standard_bool
478     {
479         \IfDocumentMetadataTF
480         {
481             \__enumext_print_footnote:
482         }
483         {
484             \bool_if:NF \l__enumext_footnotes_key_bool
485             {
486                 \__enumext_print_footnote:
487             }
488         }
489     }
490 }

```

The redefinition of `\footnote` is encapsulated to pass it to the `enumext*` and `keyans*` environments. It is executed when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

491 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
492 {
493     \IfDocumentMetadataTF
494     {
495         \__enumext_renew_footnote_mini:
496     }
497     {
498         \bool_if:NF \l__enumext_footnotes_key_bool
499         {
500             \__enumext_renew_footnote_mini:
501         }
502     }
503 }
504 \cs_new_protected:Nn \__enumext_print_footnote_starred:
505 {
506     \IfDocumentMetadataTF
507     {
508         \__enumext_print_footnote_mini:
509     }
510     {
511         \bool_if:NF \l__enumext_footnotes_key_bool
512         {
513             \__enumext_print_footnote_mini:
514         }
515     }
516 }

```

```

515     }
516 }

```

In `enumext*` and `keyans*` environments it is necessary to use “hooks” to print `\footnote` with support for tagged PDF.

```

517 \__enumext_after_env:n { enumext* }
518 {
519     \__enumext_print_footnote_starred:
520 }
521 \__enumext_after_env:n { keyans* }
522 {
523     \__enumext_print_footnote_starred:
524 }

```

(End of definition for `\__enumext_renew_footnote_standard:` and others.)

13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
\__enumext_mini_page

```

The function `\__enumext_internal_mini_page:` creates an internal `\__enumext_mini_page` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to `false` to allow spaces at the “above” of the environment, plus `\skip_vertical:N \c_zero_skip` is added to maintain alignment on “top” in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces “below”. This environment will be used internally by the `mini-env` key, it is NOT documented in the user interface and is for internal use only. Within this environment `\footnote` is redefined to make the notes look the same as if they were elsewhere in the document. This implementation is adapted from the answer given by Max Chernoff (@MaxChernoff) in [Customize minipage to support spaces below it](#).

- This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§13.43) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.48).

```

525 \cs_new_protected:Nn \__enumext_internal_mini_page:
526 {
527     \int_compare:nNnT { \__enumext_level_int } = { 0 }
528     {
529         \DeclareDocumentEnvironment{\__enumext_mini_page}{ m }
530         {
531             \__enumext_renew_footnote_standard:
532             \__enumext_minipage:w [ t ] {##1}
533             \legacy_if_gset_false:n { @minipage }
534             \skip_vertical:N \c_zero_skip
535         }
536         {
537             \skip_vertical:N \c_zero_skip
538             \__enumext_endminipage:
539             \__enumext_print_footnote_standard:
540         }
541     }
542 }

```

(End of definition for `\__enumext_internal_mini_page:` and `\__enumext_mini_page`.)

13.10 Definition of public dimension

The package `enumext` only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. This dimension is set in all environments and is only used by the `wrap-ans` key at its default value.

```

543 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

13.11.1 Definition of variables

```

\__enumext_counter_i_tl
\__enumext_counter_ii_tl
\__enumext_counter_iii_tl
\__enumext_counter_iv_tl
\__enumext_counter_v_tl
\__enumext_counter_vi_tl
\__enumext_counter_vii_tl
\__enumext_counter_viii_tl

```

Variables to store the “name of the counters” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment. The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

The initial values of these variables are set by the function `\__enumext_define_counter:Nn` (§13.11) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§13.14).

```

544 \cs_set_protected:Nn \__enumext_tmp:n
545 {
546     \tl_new:c { \__enumext_counter_ #1 _tl }
547 }
548 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

13.11.2 Definition of error messages

counters Message used in the creation of counters by `enumext` package.

```

549 \msg_new:nnnn { enumext } { counters }
550 {
551   The~counter~'#1'~is~already~defined~by~some~package~or~macro,~
552   it~cannot~be~continued.
553 }
554 {
555   Another~package~or~macro~already~defines~the~counter~'#1'.~
556   Check~for~a~conflict~with~'enumext'.
557 }
```

(End of definition for counters.)

13.11.3 Internal functions for the counters

`\__enumext_define_counter:Nn` To create the necessary “counters”, the function first checks that they are not already defined by the user or a package such as `enumitem`; otherwise it returns an error and aborts the package loading. The arguments taken by the function are:

- `enumXi` #1: A token list `\l__enumext_counter_X_tl` to “store” the counter’s name.
- `enumXii` #2: The counter’s name.

```

enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii
558 \cs_new_protected:Nn \__enumext_define_counter:Nn
559 {
560   \cs_if_exist:cTF { c@ #2 }
561   { \msg_fatal:nnn { enumext } { counters } {#2} }
562   {
563     \tl_set:Nn #1 {#2}
564     \newcounter {#2}
565   }
566 }
```

The counters created here are `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment, `enumXvi` for `keyanspic` environment, `enumXvii` for `enumext*` and `enumXviii` for the `keyans*` environments.

```

567 \__enumext_define_counter:Nn \l__enumext_counter_i_tl { enumXi }
568 \__enumext_define_counter:Nn \l__enumext_counter_ii_tl { enumXii }
569 \__enumext_define_counter:Nn \l__enumext_counter_iii_tl { enumXiii }
570 \__enumext_define_counter:Nn \l__enumext_counter_iv_tl { enumXiv }
571 \__enumext_define_counter:Nn \l__enumext_counter_v_tl { enumXv }
572 \__enumext_define_counter:Nn \l__enumext_counter_vi_tl { enumXvi }
573 \__enumext_define_counter:Nn \l__enumext_counter_vii_tl { enumXvii }
574 \__enumext_define_counter:Nn \l__enumext_counter_viii_tl { enumXviii }
```

(End of definition for `\__enumext_define_counter:Nn` and others.)

In version 1.6 the command `\resetenumext` (§13.28) was added which internally uses `\counterwithin*` so for its correct operation, “real counters” are created instead of the “integer variables” for the keys `resume` and `resume*`.

```

\c@__enumext_resume_i_int
\c@__enumext_resume_ii_int
\c@__enumext_resume_iii_int
\c@__enumext_resume_iv_int
\c@__enumext_resume_vii_int
575 \cs_set_protected:Nn \__enumext_tmp:n
576 {
577   \cs_if_exist:cTF { c@ __enumext_resume_ #1 _int }
578   { \msg_fatal:nne { enumext } { counters } { __enumext_resume_ #1 _int } }
579   {
580     \newcounter { __enumext_resume_ #1 _int }
581   }
582 }
583 \clist_map_inline:nn {i,ii,iii,iv,vii} { \__enumext_tmp:n {#1} }
```

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

13.12 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

- Direct support for this is provided since L^AT_EX release 2025-06-01[11], but the original implementation is kept so as not to hinder the internal “label and ref” system.

```
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box
```

The variable `\g__enumext_counter_styles_tl` stores the default *⟨label style⟩* and the variable `\g__enumext_widest_label_tl` the label width. The box `\l__enumext_label_width_by_box` is used by the function `\__enumext_label_width_by_box:Nn` to measure the width of a label. These variables are used by *widest* (§13.15) and *label* (§13.13) keys.

```
584 \tl_new:N \g__enumext_counter_styles_tl
585 \tl_new:N \g__enumext_widest_label_tl
586 \box_new:N \l__enumext_label_width_by_box
```

(End of definition for `\g__enumext_counter_styles_tl`, `\g__enumext_widest_label_tl`, and `\l__enumext_label_width_by_box`.)

```
\__enumext_register_default_label_wd:Nn
```

These *⟨counters⟩* will be used as default *⟨labels⟩* if the *label* key is not used for the different levels of the *enumext*, *enumext**, *keyans* and *keyans** environments, so it is necessary to get a default value for *labelwidth* from these *⟨labels⟩* at the same time.

```
587 \cs_new_protected:Nn \__enumext_register_default_label_wd:Nn
588 {
589   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
590   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
591 }
592 \__enumext_register_default_label_wd:Nn \arabic { 0 }
593 \__enumext_register_default_label_wd:Nn \Alph { M }
594 \__enumext_register_default_label_wd:Nn \alph { m }
595 \__enumext_register_default_label_wd:Nn \Roman { VIII }
596 \__enumext_register_default_label_wd:Nn \roman { viii }
```

(End of definition for `\__enumext_register_default_label_wd:Nn`.)

```
\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv
```

The function `\__enumext_label_width_by_box:Nn` sets the default *⟨labelwidth⟩* using a box width if no *labelwidth* key is passed.

```
597 \cs_new_protected:Nn \__enumext_label_width_by_box:Nn
598 {
599   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
600   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
601 }
602 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

(End of definition for `\__enumext_label_width_by_box:Nn`.)

```
\__enumext_label_style:Nnn
\__enumext_label_style:cvn
```

The function `\__enumext_label_style:Nnn` is used by the *label* key to create the variables containing the *⟨label style⟩* and allows using *⟨arabic*⟩*, *⟨Alph*⟩*, *⟨alph*⟩*, *⟨Roman*⟩* and *⟨roman*⟩* as arguments.

It loops through the defined counter styles in `\g__enumext_counter_styles_tl` (*⟨arabic⟩*, *⟨alph⟩*, *⟨Alph⟩*, *⟨roman⟩* and *⟨Roman⟩*) for example, looking for *⟨roman*⟩* and replacing that by *⟨roman{⟨counter⟩}⟩*, and doing the same for the `\g__enumext_widest_label_tl` to keep both in sync.

```
603 \cs_new_protected:Nn \__enumext_label_style:Nnn
604 {
605   \tl_clear_new:N #1
606   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
607   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
608   \tl_map_inline:Nn \g__enumext_counter_styles_tl
609   {
610     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
611     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
612     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
613   }
614   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
615   { \tl_use:N \g__enumext_widest_label_tl }
616   \tl_set_eq:cn { the #2 } #1
617 }
618 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

(End of definition for `\__enumext_label_style:Nnn`.)

13.13 Setting keys associated with label

When *tagged* PDF is active *⟨make label⟩* is redefined using *⟨make box⟩* to work correctly (§13.38). From the user side it is convenient to have a key that allows using this redefinition with *⟨make box⟩* without having *⟨IfDocumentMetadataTF⟩* active.

`mode-box` Defines the key `mode-box` only for the “first level” of `enumext` and `enumext*` environments.

```

619 \cs_set_protected:Nn \__enumext_tmp:n
620 {
621   \keys_define:nn { enumext / #1 }
622   {
623     mode-box .bool_set:N = \l__enumext_mode_box_bool,
624     mode-box .initial:n = false,
625     mode-box .value_forbidden:n = true,
626   }
627 }
628 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for `mode-box`.)

`font` Definition of keys `font`, `labelsep`, `labelwidth`, `wrap-label` and `wrap-label*` keys for `enumext` and `keyans` environments.

```

629 \cs_set_protected:Nn \__enumext_tmp:nn
630 {
631   \keys_define:nn { enumext / #1 }
632   {
633     font .tl_set:c = { \l__enumext_label_font_style_ #2 _tl },
634     font .value_required:n = true,
635     labelsep .dim_set:c = { \l__enumext_labelsep_ #2 _dim },
636     labelsep .initial:n = {0.3333em},
637     labelsep .value_required:n = true,
638     labelwidth .dim_set:c = { \l__enumext_labelwidth_ #2 _dim },
639     labelwidth .value_required:n = true,
640     wrap-label .cs_set_protected:cp = { \__enumext_wrapper_label_ #2 :n } ##1,
641     wrap-label .initial:n = {##1},
642     wrap-label .value_required:n = true,
643     wrap-label* .code:n =
644     {
645       \bool_set_true:c { \l__enumext_wrap_label_opt_ #2 _bool }
646       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
647     },
648     wrap-label* .value_required:n = true,
649   }
650 }
651 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `font` and others.)

`align` The `align` key is implemented differently for “starred” and “non starred” environments. For compatibility with *tagged* PDF it is necessary to set `\l__enumext_align_label_pos_X_str`.

```

652 \cs_set_protected:Nn \__enumext_tmp:nn
653 {
654   \keys_define:nn { enumext / #1 }
655   {
656     align .choices:nn =
657     { left , right , center }
658     {
659       \int_case:nn { \l_keys_choice_int }
660       {
661         {1}
662         {
663           \tl_clear:c { \l__enumext_label_fill_left_ #2 _tl }
664           \tl_set:cn { \l__enumext_label_fill_right_ #2 _tl } { \hfill }
665           \str_set:cn { \l__enumext_align_label_pos_ #2 _str } { l }
666         }
667         {2}
668         {
669           \tl_set:cn { \l__enumext_label_fill_left_ #2 _tl } { \hfill }
670           \tl_clear:c { \l__enumext_label_fill_right_ #2 _tl }
671           \str_set:cn { \l__enumext_align_label_pos_ #2 _str } { r }
672         }
673         {3}
674         {
675           \tl_set:cn { \l__enumext_label_fill_left_ #2 _tl } { \hfill }
676           \tl_set:cn { \l__enumext_label_fill_right_ #2 _tl } { \hfill }
677           \str_set:cn { \l__enumext_align_label_pos_ #2 _str } { c }

```

```

678         }
679     },
680     },
681     align .initial:n = left,
682     align .value_required:n = true,
683     align / unknown .code:n =
684         \msg_error:nneee { enumext } { unknown-choice }
685         { align } { left,~right,~ center } { \exp_not:n {##1} },
686 }
687 }
688 \clist_map_inline:nn
689 {
690     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
691 }
692 { \__enumext_tmp:nn #1 }
693 \cs_set_protected:Nn \__enumext_tmp:nn
694 {
695     \keys_define:nn { enumext / #1 }
696     {
697         align .choices:nn =
698             { left , right , center }
699             {
700                 \int_case:nn { \l_keys_choice_int }
701                 {
702                     {1} { \str_set:cn { l__enumext_align_label_ #2 _str } { l } }
703                     {2} { \str_set:cn { l__enumext_align_label_ #2 _str } { r } }
704                     {3} { \str_set:cn { l__enumext_align_label_ #2 _str } { c } }
705                 }
706             },
707         align .initial:n = left,
708         align .value_required:n = true,
709         align / unknown .code:n =
710             \msg_error:nneee { enumext } { unknown-choice }
711             { align } { left,~right,~ center } { \exp_not:n {##1} },
712     }
713 }
714 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

13.14 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for `<label>`, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “*label and ref*” system.

13.14.1 Definition of variables

Internal variables used by `ref` key (§13.14).

```

\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_renew_counter_X_tl
\l__enumext_the_counter_X_tl
715 \tl_new:N \l__enumext_ref_key_arg_tl
716 \tl_new:N \l__enumext_ref_the_count_tl
717 \cs_set_protected:Nn \__enumext_tmp:n
718 {
719     \tl_new:c { l__enumext_renew_counter_ #1 _tl }
720     \tl_new:c { l__enumext_the_counter_ #1 _tl }
721     \tl_set:ce { l__enumext_the_counter_ #1 _tl } { \exp_not:c { theenumX #1 } }
722 }
723 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {##1} }

```

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

13.14.2 Definition of error messages

key-ref-empty

Messages used by `ref` key.

```

724 \msg_new:nnnn { enumext } { key-ref-empty }
725 {
726     Key~'ref'~needs~a~value~in~'#1'~\msg_line_context:.
727 }
728 {
729     Give~a~value~to~the~key~'ref'~or~remove~it.
730 }

```

(End of definition for key-ref-empty.)

13.14.3 Define and set label and ref keys for enumext environment

label Sets the default *⟨labels⟩* of the *four levels* of `enumext` environment, along with the default values for
 ref `labelwidth`, `widest` and `ref` keys.

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
731 \cs_set_protected:Nn \__enumext_tmp:nnn
732 {
733   \tl_const:cn { c__enumext_default_label_ #2 _tl } {#3}
734   \keys_define:nn { enumext / #1 }
735   {
736     label .code:n =
737     {
738       \__enumext_label_style:cvn { \l__enumext_label_ #2 _tl }
739       { \l__enumext_counter_ #2 _tl } {##1}
740       \dim_set_eq:cN { \l__enumext_labelwidth_ #2 _dim }
741       \l__enumext_current_widest_dim
742     },
743     label .initial:n = #3,
744     label .value_required:n = true,
745     ref .code:n = { \__enumext_standard_ref:nn {#2} {##1} },
746     ref .value_required:n = true,
747   }
748 }
749 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
750 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
751 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
752 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for `label` and others.)

```

\__enumext_standard_ref:nn
\__enumext_standard_ref:

```

The `\__enumext_standard_ref:nn` function will first pass the key *argument* `ref` to the variable `\l__enumext_ref_key_arg_tl` and analyze its state, if it is not *empty* it will set a copy of the *current counter style* saved in `\l__enumext_the_counter_X_tl` to `\l__enumext_ref_the_count_tl` and then set the variable `\l__enumext_renew_counter_X_tl` which will modify `\theenumX`.

```

753 \cs_new_protected:Nn \__enumext_standard_ref:nn
754 {
755   \tl_set:Nn \l__enumext_ref_key_arg_tl {#2}
756   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
757   {
758     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
759   }
760   {
761     \tl_set_eq:Nc \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ #1 _tl }
762     \tl_set:ce { \l__enumext_renew_counter_ #1 _tl }
763     {
764       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
765       { \exp_not:V \l__enumext_ref_key_arg_tl }
766     }
767   }
768 }

```

Finally the function `\__enumext_standard_ref:` will execute the modification for the reference system in the second argument of the environment definition `enumext`.

```

769 \cs_new_protected:Nn \__enumext_standard_ref:
770 {
771   \tl_if_empty:cF { \l__enumext_renew_counter_ \__enumext_level: _tl }
772   {
773     \tl_use:c { \l__enumext_renew_counter_ \__enumext_level: _tl }
774   }
775 }

```

(End of definition for `\__enumext_standard_ref:nn` and `\__enumext_standard_ref:`.)

13.14.4 Define and set label and ref keys for enumext* and keyans* environments

label Sets the default *⟨labels⟩* for `enumext*` and `keyans*` environments, along with the default value for
 ref `labelwidth` key and `ref` key.

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
776 \cs_set_protected:Nn \__enumext_tmp:nnn
777 {
778   \tl_const:cn { c__enumext_default_label_ #2 _tl } {#3}
779   \keys_define:nn { enumext / #1 }

```

```

780     {
781       label .code:n =
782       {
783         \__enumext_label_style:cnv { l__enumext_label_ #2 _tl }
784         { l__enumext_counter_ #2 _tl } {##1}
785         \dim_set_eq:cN { l__enumext_labelwidth_ #2 _dim }
786         \l__enumext_current_widest_dim
787       },
788       label .initial:n = #3,
789       label .value_required:n = true,
790       ref .code:n = { \__enumext_starred_ref:n {##1} },
791       ref .value_required:n = true,
792     }
793   }
794   \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.* }
795   \__enumext_tmp:nnn { keyans* } { viii } { \Alph* ) }

```

(End of definition for `label` and others.)

__enumext_starred_ref:n
 __enumext_starred_ref:

The implementation of `\__enumext_starred_ref:n` is the same as that used for the environment `enumext`.

```

796 \cs_new_protected:Nn \__enumext_starred_ref:n
797 {
798   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
799   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
800   {
801     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
802     {
803       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
804     }
805     {
806       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
807       \tl_set:Ne \l__enumext_renew_counter_vii_tl
808       {
809         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
810         { \exp_not:V \l__enumext_ref_key_arg_tl }
811       }
812     }
813   }
814   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
815   {
816     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
817     {
818       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
819     }
820     {
821       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
822       \tl_set:Ne \l__enumext_renew_counter_viii_tl
823       {
824         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
825         { \exp_not:V \l__enumext_ref_key_arg_tl }
826       }
827     }
828   }
829 }

```

Finally the function `\__enumext_starred_ref:` will execute the modification for the reference system in the second argument of the `enumext*` and `keyans*` environment definition.

```

830 \cs_new_protected:Nn \__enumext_starred_ref:
831 {
832   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
833   {
834     \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
835     {
836       \tl_use:N \l__enumext_renew_counter_vii_tl
837     }
838   }
839   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
840   {
841     \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
842     {
843       \tl_use:N \l__enumext_renew_counter_viii_tl

```

```

844     }
845   }
846 }

```

(End of definition for `\__enumext_starred_ref:n` and `\__enumext_starred_ref:.`)

13.14.5 Define and set `label` and `ref` keys for `keyans` and `keyanspic` environments

`label` Sets the default *⟨label⟩* for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` if it has not been established and `ref` key. The `keyanspic` environment use the same *⟨label⟩* as the `keyans` environment.

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
847 \keys_define:nn { enumext / keyans }
848 {
849   label .code:n =
850   {
851     \__enumext_label_style:cnv { \__enumext_label_v_tl }
852     { \__enumext_counter_v_tl } {#1}
853     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
854     { \__enumext_counter_vi_tl } {#1}
855     \dim_set_eq:NN
856     \__enumext_labelwidth_v_dim \__enumext_current_widest_dim
857   },
858   label .initial:n = \Alph*,
859   label .value_required:n = true,
860   ref .code:n = { \__enumext_keyans_ref:n {#1} },
861   ref .value_required:n = true,
862 }

```

(End of definition for `label` and others.)

`\__enumext_keyans_ref:n` The implementation of `\__enumext_keyans_ref:n` is the same as that used for the environment `enumext`.
`\__enumext_keyans_ref:`

```

863 \cs_new_protected:Nn \__enumext_keyans_ref:n
864 {
865   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
866   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
867   {
868     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
869   }
870   {
871     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
872     \tl_put_right:Ne \__enumext_renew_counter_v_tl
873     {
874       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
875       { \exp_not:V \__enumext_ref_key_arg_tl }
876     }
877   }
878 }

```

Finally the function `\__enumext_keyans_ref:` will execute the modification for the reference system in the second argument of the `keyans` environment definition.

```

879 \cs_new_protected:Nn \__enumext_keyans_ref:
880 {
881   \tl_if_empty:NF \__enumext_renew_counter_v_tl
882   {
883     \tl_use:N \__enumext_renew_counter_v_tl
884   }
885 }

```

(End of definition for `\__enumext_keyans_ref:n` and `\__enumext_keyans_ref:.`)

13.15 Setting `start`, `start*` and `widest` keys

`\__enumext_start_from:NNn` The function `\__enumext_start_from:NNn` used by `start` and `start*` keys takes three arguments:

```

\__enumext_start_from:ccn
\__enumext_start_from:cce
#1: \__enumext_label_X_tl
#2: \__enumext_start_X_int
#3: ⟨integer or string⟩

```

The argument ‘#1’ is the “*counter style*” set by `label` key, ‘#2’ is returned by the function, ‘#3’ can be an *⟨integer⟩* or *⟨string⟩* of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start*=A` or `start=1` to be used.

- In version 1.6 it is allowed to pass the `resume` key *without value* by means of the command `\setenumext`, for the correct operation of this, the boolean variable `\l__enumext_resume_count_bool` set by the `resume` key *without value* must be set to `false` (§13.27). This is necessary to be able to “reset” the *start value* by means of the `start` or `start*` keys.

```

886 \cs_new_protected:Nn \__enumext_start_from:NNn
887 {
888   \bool_set_false:N \l__enumext_resume_count_bool
889   \__enumext_if_is_int:nTF {#3}
890   {
891     \int_set:Nn #2 {#3}
892   }
893   {
894     \regex_if_match:nVT { \c{Alpha} | \c{alpha} } #1
895     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
896     \regex_if_match:nVT { \c{Roman} | \c{roman} } #1
897     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
898   }
899 }
900 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

(End of definition for `\__enumext_start_from:NNn`.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

The function `\__enumext_widest_from:nNNn` used by the `widest` key takes four arguments:

- #1: The counter associated with the environment level
- #2: `\l__enumext_label_X_tl`
- #3: `\l__enumext_labelwidth_X_dim`
- #4: *integer or string*

The arguments ‘#2’ and ‘#3’ are the values set by `label` and `labelwidth` keys, ‘#4’ can be an *integer* or *string* of the form `\Alpha`, `\alpha`, `\Roman` or `\roman`. The value of ‘#4’ is set temporarily for the identified counter in this point (level), then the value is expanded into a “box” and the “width” of the “box” is returned.

```

901 \cs_new_protected:Nn \__enumext_widest_from:nNNn
902 {
903   \__enumext_if_is_int:nTF {#4}
904   {
905     \setcounter{enumX #1} {#4}
906   }
907   {
908     \regex_if_match:nVT { \c{Alpha} | \c{alpha} } #2
909     { \setcounter{enumX #1} { \int_from_alph:n {#4} } }
910     \regex_if_match:nVT { \c{Roman} | \c{roman} } #2
911     { \setcounter{enumX #1} { \int_from_roman:n {#4} } }
912   }
913   \__enumext_label_width_by_box:cv
914   { \__enumext_labelwidth_ #1 _dim } { \__enumext_label_ #1 _tl }
915 }
916 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for `\__enumext_widest_from:nNNn`.)

Now define and set `start*`, `start` and `widest` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

`start`

`start*`

`widest`

```

917 \cs_set_protected:Nn \__enumext_tmp:nn
918 {
919   \keys_define:nn { enumext / #1 }
920   {
921     start* .code:n =
922     {
923       \__enumext_start_from:ccn
924       { \__enumext_label_ #2 _tl }
925       { \__enumext_start_ #2 _int } {##1}
926     },
927     start* .value_required:n = true,
928     start .code:n =
929     {
930       \__enumext_start_from:cce
931       { \__enumext_label_ #2 _tl }
932       { \__enumext_start_ #2 _int } { \int_eval:n {##1} }
933     },
934     start .initial:n = 1,

```

```

935     start .value_required:n = true,
936     widest .code:n =
937     {
938         \__enumext_widest_from:nccn {#2}
939         { \__enumext_label_ #2 _tl }
940         { \__enumext_labelwidth_ #2 _dim } {##1}
941     },
942     widest .value_required:n = true,
943 }
944 }
945 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `start`, `start*`, and `widest`.)

13.16 Setting keys for penalties

The three parameters `\@beginparpenalty`, `\@itempenalty` and `\@endparpenalty` work together to ensure that list environments look good, avoiding unsightly page breaks that can break the flow of the `list`, so it's a good idea to have some *keys* to access these.

```

946 \cs_set_protected:Nn \__enumext_tmp:nn
947 {
948     \keys_define:nn { enumext / #1 }
949     {
950         beginpenalty .int_set:c = { \__enumext_beginparpenalty_ #2 _int },
951         beginpenalty .initial:n = -51,
952         beginpenalty .value_required:n = true,
953         midpenalty .int_set:c = { \__enumext_itempenalty_ #2 _int },
954         midpenalty .initial:n = -51,
955         midpenalty .value_required:n = true,
956         endpenalty .int_set:c = { \__enumext_endparpenalty_ #2 _int },
957         endpenalty .initial:n = -51,
958         endpenalty .value_required:n = true,
959     }
960 }
961 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `beginpenalty`, `midpenalty`, and `endpenalty`.)

13.17 Setting keys for vertical spaces

Define and set `topsep`, `partopsep`, `parsep`, `itemsep`, `noitemsep` and `nosep` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

962 \cs_set_protected:Nn \__enumext_tmp:nnnnnn
963 {
964     \keys_define:nn { enumext / #1 }
965     {
966         topsep .skip_set:c = { \__enumext_topsep_ #2 _skip },
967         topsep .initial:n = {#3},
968         topsep .value_required:n = true,
969         partopsep .skip_set:c = { \__enumext_partopsep_ #2 _skip },
970         partopsep .initial:n = {#4},
971         partopsep .value_required:n = true,
972         parsep .skip_set:c = { \__enumext_parsep_ #2 _skip },
973         parsep .initial:n = {#5},
974         parsep .value_required:n = true,
975         itemsep .skip_set:c = { \__enumext_itemsep_ #2 _skip },
976         itemsep .initial:n = {#6},
977         itemsep .value_required:n = true,
978         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
979         noitemsep .value_forbidden:n = true,
980         nosep .meta:n =
981         { itemsep = 0pt, parsep = 0pt, topsep = 0pt, partopsep = 0pt },
982         nosep .value_forbidden:n = true,
983     }
984 }

```

Now the values are set based on the standard `article` class in `10pt`.

```

985 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
986 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
987 { 4.0pt plus 2.0pt minus 1.0pt }

```

```

988 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
989 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
990 { 2.0pt plus 1.0pt minus 1.0pt }
991 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
992 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
993 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
994 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
995 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
996 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
997 { 2.0pt plus 1.0pt minus 1.0pt }
998 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
999 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
1000 { 4.0pt plus 2.0pt minus 1.0pt }
1001 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
1002 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
1003 { 2.0pt plus 1.0pt minus 1.0pt }

```

(End of definition for topsep and others.)

13.18 Setting base-fix key

When nesting starting right after `\item` (without material between them) there is a problem with the alignment of the *baseline* between the two environments. One way to get around this problem is to start the paragraph with `\mode_leave_vertical:`, apply a negative `\vspace` and set the keys `topsep`, `above` and `above*` equal to `0pt` for the “first level” of the nested `enumext` environment.

`base-fix` Defines the key `base-fix` only for the “first level” of `enumext` environment.

```

1004 \keys_define:nn { enumext / level-1 }
1005 {
1006   base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
1007   base-fix .initial:n = false,
1008   base-fix .value_forbidden:n = true,
1009 }

```

(End of definition for base-fix.)

`\__enumext_nested_base_line_fix:` The function `\__enumext_nested_base_line_fix:` passed to the `\__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.43) will be responsible for applying the *baseline correction* and adjusting the `\keys` for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§13.51).

First, the function code is implemented from the user side of the `base-fix` key, that is, only the user knows when it is necessary to apply it within the document in which case the variable `\l__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `\l__enumext_base_line_fix_bool` is true.

The negative “vertical space” is a line of text plus the value of `parsep` of the `enumext*` environment, since this is the gap that $\text{\TeX}$ leaves between the label of the `\item` and the first line of the nested list. With `columns` greater than one this gap does not include the `parsep`, so only the line of text is removed. The value of `\parsep` is not used, because inside `enumext*` it is equal to `itemsep`.

Sets the values of the keys `topsep`, `above` and `above*` for the “first level” of `enumext` environment equal to `0pt` and finally sets the variable `\l__enumext_base_line_fix_bool` to false.

```

1010 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
1011 {
1012   \bool_lazy_all:nT
1013   {
1014     { \bool_if_p:N \l__enumext_starred_first_bool }
1015     { \bool_if_p:N \l__enumext_base_line_fix_bool }
1016     { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
1017   }
1018   {
1019     \mode_leave_vertical:
1020     \vspace
1021     {
1022       -\dim_eval:n
1023       {
1024         \baselineskip
1025         + \int_compare:nNnTF { \l__enumext_columns_i_int } > { 1 }
1026         { 0pt }
1027         { \l__enumext_parsep_vii_skip }

```

```

1028         }
1029     }
1030     \keys_set:nn { enumext / level-1 }
1031     {
1032         topsep = 0pt, above = 0pt, above* = 0pt,
1033     }
1034 }

```

When the `\printkeyans` command is run with the *starred argument* ‘*’ the variable `\__enumext_print_keyans_star` is true the negative space is added with `\skip_vertical:n`, the `enumext` interface of `\vskip`, instead of `\vspace`. After `\mode_leave_vertical:` TeX is in (*horizontal mode*), where `\vspace` would only store the skip in a `\vadjust` to be applied after the current line, whereas `\vskip` ends the (still empty) paragraph and applies the skip immediately. The skip is followed by `\skip_vertical:N \c_zero_skip` so that the negative space is not the last item of the vertical list.

```

1035     \bool_lazy_and:nnT
1036     { \bool_if_p:N \__enumext_starred_first_bool }
1037     { \bool_if_p:N \__enumext_print_keyans_star_bool }
1038     {
1039         \mode_leave_vertical:
1040         \skip_vertical:n { -\baselineskip }
1041         \skip_vertical:N \c_zero_skip
1042         \keys_set:nn { enumext / level-1 }
1043         {
1044             topsep = 0pt, above = 0pt, above* = 0pt,
1045         }
1046     }
1047     \bool_set_false:N \__enumext_base_line_fix_bool
1048 }

```

(End of definition for `\__enumext_nested_base_line_fix:`.)

13.19 Setting keys for horizontal spaces

Define and set `itemindent`, `rightmargin`, `listparindent`, `list-offset` and `list-indent` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1049 \cs_set_protected:Nn \__enumext_tmp:nn
1050 {
1051     \keys_define:nn { enumext / #1 }
1052     {
1053         itemindent .dim_set:c = { \__enumext_fake_item_indent_ #2 _dim },
1054         itemindent .value_required:n = true,
1055         rightmargin .dim_set:c = { \__enumext_rightmargin_ #2 _dim },
1056         rightmargin .value_required:n = true,
1057         listparindent .dim_set:c = { \__enumext_listparindent_ #2 _dim },
1058         listparindent .value_required:n = true,
1059         list-offset .dim_set:c = { \__enumext_listoffset_ #2 _dim },
1060         list-offset .value_required:n = true,
1061         list-indent .code:n =
1062         {
1063             \bool_set_true:c { \__enumext_leftmargin_tmp_ #2 _bool }
1064             \dim_set:cn { \__enumext_leftmargin_tmp_ #2 _dim } {##1}
1065         },
1066         list-indent .value_required:n = true,
1067     }
1068 }
1069 \clist_map_inline:nn
1070 {
1071     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1072 }
1073 { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

1074 \cs_set_protected:Nn \__enumext_tmp:nn
1075 {
1076     \keys_define:nn { enumext / #1 }
1077     {
1078         itemindent .dim_set:c = { \__enumext_fake_item_indent_ #2 _dim },
1079         itemindent .value_required:n = true,
1080         rightmargin .dim_set:c = { \__enumext_rightmargin_ #2 _dim },

```

```

1081     rightmargin .value_required:n = true,
1082     listparindent .dim_set:c = { l__enumext_listparindent_ #2 _dim },
1083     listparindent .value_required:n = true,
1084     list-offset .dim_set:c = { l__enumext_listoffset_ #2 _dim },
1085     list-offset .value_required:n = true,
1086     list-indent .meta:n = { list-offset = ##1 },
1087     list-indent .value_required:n = true,
1088   }
1089 }
1090 \clist_map_inline:nn
1091 {
1092   {enumext*}{vii}, {keyans*}{viii}
1093 }
1094 { \__enumext_tmp:nn #1 }

```

(End of definition for `itemindent` and others.)

13.19.1 Functions for setting the fake `itemindent`

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. This value is stored in the variable and only applied when it is greater than `\opt`. Here it is necessary to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

1095 \cs_set_protected:Nn \__enumext_fake_item_indent:
1096 {
1097   \dim_compare:nNnT
1098   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1099   >
1100   { \c_zero_dim }
1101   {
1102     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1103     {
1104       \exp_not:N \mode_leave_vertical:
1105       \exp_not:n { \skip_horizontal:n }
1106       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1107       \exp_not:N \ignorespaces
1108     }
1109   }
1110 }
1111 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1112 {
1113   \dim_compare:nNnT
1114   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1115   {
1116     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1117     {
1118       \exp_not:N \mode_leave_vertical:
1119       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1120       \exp_not:N \ignorespaces
1121     }
1122   }
1123 }
1124 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1125 {
1126   \dim_compare:nNnT
1127   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1128   {
1129     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1130     {
1131       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1132       \exp_not:N \ignorespaces
1133     }
1134   }
1135 }
1136 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1137 {
1138   \dim_compare:nNnT
1139   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1140   {
1141     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1142     {

```

```

1143         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1144         \exp_not:N \ignorespaces
1145     }
1146 }
1147 }

```

(End of definition for `\__enumext_fake_item_indent:` and others.)

13.20 Setting show-length key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to `true` and calls the function `\__enumext_show_length:nnn` which prints all the values of the “vertical” and “horizontal” parameters calculated and used.

```

1148 \cs_set_protected:Nn \__enumext_tmp:nn
1149 {
1150     \keys_define:nn { enumext / #1 }
1151     {
1152         show-length .bool_set:c = { \l__enumext_show_length_ #2 _bool },
1153         show-length .initial:n = false,
1154         show-length .value_required:n = true,
1155     }
1156 }
1157 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `show-length`.)

13.21 Setting item-break key

item-break

Define and set `item-break` key for `enumext*` and `keyans*` environments. The key sets the boolean variable `\l__enumext_item_break_X_bool`, which is `false` by default, and is used when each `\item` starts (§13.48).

```

1158 \cs_set_protected:Nn \__enumext_tmp:nn
1159 {
1160     \keys_define:nn { enumext / #1 }
1161     {
1162         item-break .bool_set:c = { \l__enumext_item_break_ #2 _bool },
1163         item-break .initial:n = false,
1164         item-break .value_required:n = true,
1165     }
1166 }
1167 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `item-break`.)

13.22 Setting before, after and first keys

before

before*

after

first

Define and set `before`, `before*`, `after` and `first` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1168 \cs_set_protected:Nn \__enumext_tmp:nn
1169 {
1170     \keys_define:nn { enumext / #1 }
1171     {
1172         before .tl_set:c = { \l__enumext_before_no_starred_key_ #2 _tl },
1173         before .value_required:n = true,
1174         before* .tl_set:c = { \l__enumext_before_starred_key_ #2 _tl },
1175         before* .value_required:n = true,
1176         after .tl_set:c = { \l__enumext_after_stop_list_ #2 _tl },
1177         after .value_required:n = true,
1178         first .tl_set:c = { \l__enumext_after_list_args_ #2 _tl },
1179         first .value_required:n = true,
1180     }
1181 }
1182 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `before` and others.)

13.22.1 Functions for before, after and first keys in enumext

The function `\__enumext_before_args_exec:` executes the `{\code}` set by the `before*` key “before” the `enumext` environment is started. The `{\code}` is executed “without” knowing any definition of the `{\arg two}` of the list: `{\code}\list{\arg one}{\arg two}`.

```

1183 \cs_new_protected:Nn \__enumext_before_args_exec:

```

```

1184 {
1185     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1186 }

```

The function `\__enumext_before_keys_exec:` executes the `{\code}` set by the `before` key “before” the `enumext` environment is started in *second argument* of the list. The `{\code}` is executed “knowing” all the definitions and values provided by the `(keys): \list{\arg one}{\arg two}{\code}`

```

1187 \cs_new_protected:Nn \__enumext_before_keys_exec:
1188 {
1189     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1190 }

```

The function `\__enumext_after_stop_list:` executes the `{\code}` set by the `after` key “after” the `enumext` environment has finished: `\endlist{\code}`.

```

1191 \cs_new_protected:Nn \__enumext_after_stop_list:
1192 {
1193     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1194 }

```

The function `\__enumext_after_args_exec:` executes the `{\code}` set by the `first` key after the end of the second argument of the list defining the `enumext` environment, just before the first occurrence of `\item: \list{\arg one}{\arg two}{\code}\item.`

```

1195 \cs_new_protected:Nn \__enumext_after_args_exec:
1196 {
1197     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1198 }

```

(End of definition for `\__enumext_before_args_exec:` and others.)

13.22.2 Functions for before, after and first keys in keyans

`\__enumext_before_args_exec_v:` Same implementation as the one used in the `enumext` environment.

```

\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
1199 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1200 {
1201     \tl_use:N \l__enumext_before_starred_key_v_tl
1202 }
1203 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1204 {
1205     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1206 }
1207 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1208 {
1209     \tl_use:N \l__enumext_after_stop_list_v_tl
1210 }
1211 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1212 {
1213     \tl_use:N \l__enumext_after_list_args_v_tl
1214 }

```

(End of definition for `\__enumext_before_args_exec_v:` and others.)

13.22.3 Functions for before, after and first keys in enumext* and keyans*

`\__enumext_before_args_exec_vii:` Same implementation as the one used in the `enumext` environment, except that there are no `\__enumext_after_args_exec_viii:` and `\__enumext_after_args_exec_viii:` functions: the `{\code}` set by the `first` key is used directly by `\__enumext_start_item_vii:w` and `\__enumext_start_item_viii:w`, inside the minipage of each `\item.`

```

\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_stop_list_viii:
1215 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1216 {
1217     \tl_use:N \l__enumext_before_starred_key_vii_tl
1218 }
1219 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1220 {
1221     \tl_use:N \l__enumext_before_starred_key_viii_tl
1222 }
1223 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1224 {
1225     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1226 }
1227 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:

```

```

1228 {
1229   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1230 }
1231 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1232 {
1233   \tl_use:N \l__enumext_after_stop_list_vii_tl
1234 }
1235 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1236 {
1237   \tl_use:N \l__enumext_after_stop_list_viii_tl
1238 }

```

(End of definition for `\__enumext_before_args_exec_vii:` and others.)

13.23 Setting keys for multicols and minipage

The default value of the `columns-sep` key is zero; when the value is zero it is calculated in the internal definition of the environments from the dimension variable `\l__enumext_columns_sep_X_dim`. Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1239 \cs_set_protected:Nn \__enumext_tmp:nn
1240 {
1241   \keys_define:nn { enumext / #1 }
1242   {
1243     mini-env   .dim_set:c = { \l__enumext_minipage_right_ #2 _dim },
1244     mini-env   .value_required:n = true,
1245     mini-sep   .dim_set:c = { \l__enumext_minipage_hsep_ #2 _dim },
1246     mini-sep   .initial:n = 0.3333em,
1247     mini-sep   .value_required:n = true,
1248     columns-sep .dim_set:c = { \l__enumext_columns_sep_ #2 _dim },
1249     columns-sep .value_required:n = true,
1250     columns    .int_set:c = { \l__enumext_columns_ #2 _int },
1251     columns    .initial:n = 1,
1252     columns    .value_required:n = true,
1253   }
1254 }
1255 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the command `\miniright` is not available, so the keys `mini-right` and `mini-right*` are added to implement support for `minipage` environment.

```

1256 \cs_set_protected:Nn \__enumext_tmp:nn
1257 {
1258   \keys_define:nn { enumext / #1 }
1259   {
1260     mini-right .tl_gset:c = { g__enumext_miniright_code_ #2 _tl },
1261     mini-right .value_required:n = true,
1262     mini-right* .code:n =
1263     {
1264       \bool_gset_true:c { g__enumext_minipage_center_ #2 _bool }
1265       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1266     },
1267     mini-right* .value_required:n = true,
1268   }
1269 }
1270 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

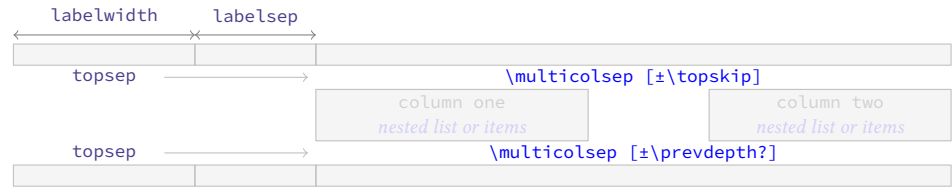
(End of definition for `mini-env` and others.)

13.24 Adjustment of vertical spaces for multicols

When nesting a “*list environment*” inside the `multicols` environment, the values of the “*vertical spaces*” are lost, basically the `multicols` environment takes control over them. Graphically, this can be seen in figure 7.

To keep the desired spaces *above* and *below* in the “*list environment*” (`\topsep` + `[\partopsep]`) it is necessary to “*adjust*” the spaces added by the `multicols` environment. The most appropriate option in this case is to use a “*context sensitive*” vertical space with `\addvspace`.

- It should be made clear that the implementation here is a “*bit questionable*”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values are “*stretched*”, possibly due to the use of `\raggedcolumns` and this affects the lower space when closing the environment, which is “*smaller*” than expected. My attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

Figure 7: Representation of the vertical space in `multicols` for a nested level.

13.24.1 Adjustment of vertical spaces for multicols in enumext

`\__enumext_multi_set_vskip:` The function `\__enumext_multi_set_vskip:` will take care of determining the “adjusted spaces” applied “above” and “below” the `multicols` environment in `enumext`.

The default values are set taking into account that $\text{T}_{\text{E}}\text{X}$ is in *horizontal mode*, then the settings for the *vertical mode*, in which `\partopsep` comes into play, are made.

Set the values of `\__enumext_multicols_above_X_skip` and `\__enumext_multicols_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

1271 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1272 {
1273   \skip_set:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1274   {
1275     \skip_use:c { \__enumext_topsep_ \__enumext_level: _skip }
1276   }
1277   \skip_set:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1278   {
1279     \skip_use:c { \__enumext_topsep_ \__enumext_level: _skip }
1280   }
1281   \__enumext_add_pre_parsep:
1282 }

```

(End of definition for `\__enumext_multi_set_vskip:.`)

`\__enumext_add_pre_parsep:` The function `\__enumext_add_pre_parsep:` “adjusts” the value of `\__enumext_multicols_above_X_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces*.

```

1283 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1284 {
1285   \int_case:nn { \__enumext_level_int }
1286   {
1287     { 2 }
1288     {
1289       \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
1290       {
1291         \skip_add:Nn \__enumext_multicols_above_ii_skip
1292         {
1293           \__enumext_parsep_i_skip
1294         }
1295       }
1296     }
1297     { 3 }
1298     {
1299       \skip_if_eq:nnF { \__enumext_parsep_ii_skip } { \c_zero_skip }
1300       {
1301         \skip_add:Nn \__enumext_multicols_above_iii_skip
1302         {
1303           \__enumext_parsep_ii_skip
1304         }
1305       }
1306     }
1307     { 4 }
1308     {
1309       \skip_if_eq:nnF { \__enumext_parsep_iii_skip } { \c_zero_skip }
1310       {
1311         \skip_add:Nn \__enumext_multicols_above_iv_skip
1312         {
1313           \__enumext_parsep_iii_skip
1314         }
1315       }
1316     }

```

```

1317     }
1318 }

```

(End of definition for `\__enumext_add_pre_parse:`)

`\__enumext_multi_addvspace:` The function `\__enumext_multi_addvspace:` will apply the spaces set using `\addvspace` “above” the `\multicols` environment in `enumext`, taking into account whether $\TeX$ is in *horizontal mode* or *vertical mode*.

```

1319 \cs_new_protected:Nn \__enumext_multi_addvspace:
1320 {
1321   \__enumext_multi_set_vskip:
1322   \mode_if_vertical:T
1323   {
1324     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1325     {
1326       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1327     }
1328     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1329     {
1330       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1331     }
1332   }
1333   \par\nopagebreak
1334   \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
1335 }

```

(End of definition for `\__enumext_multi_addvspace:`)

13.24.2 Adjustment of vertical spaces for multicols in keyans

`\__enumext_keyans_multi_set_vskip:` The function `\__enumext_keyans_multi_set_vskip:` will take care of determining the “adjusted spaces” applied “above” and “below” the `\multicols` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`, including the value of `\parsep` of the previous level, which is added to the space “above”.

`\__enumext_keyans_multi_addvspace:`

```

1336 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1337 {
1338   \skip_set:Nn \__enumext_multicols_above_v_skip
1339   {
1340     \l__enumext_topsep_v_skip
1341   }
1342   \skip_set:Nn \__enumext_multicols_below_v_skip
1343   {
1344     \l__enumext_topsep_v_skip
1345   }
1346   \skip_if_eq:nnF { \__enumext_parsep_i_skip } { \c_zero_skip }
1347   {
1348     \skip_add:Nn \__enumext_multicols_above_v_skip { \__enumext_parsep_i_skip }
1349   }
1350 }
1351 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1352 {
1353   \__enumext_keyans_multi_set_vskip:
1354   \mode_if_vertical:T
1355   {
1356     \skip_add:Nn \__enumext_multicols_above_v_skip
1357     {
1358       \skip_use:N \__enumext_partopsep_v_skip
1359     }
1360     \skip_add:Nn \__enumext_multicols_below_v_skip
1361     {
1362       \skip_use:N \__enumext_partopsep_v_skip
1363     }
1364   }
1365   \par\nopagebreak
1366   \addvspace{ \__enumext_multicols_above_v_skip }
1367 }

```

(End of definition for `\__enumext_keyans_multi_set_vskip:` and `\__enumext_keyans_multi_addvspace:`)

Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

13.25 Adjustment of vertical spaces for minipage

When nesting a “list environment” within the `minipage` environment, the values of the “vertical spaces” are lost. Graphically it can be seen like in the figure 8.

To keep the “left” and “right” environments “aligned on top”, preserving the `\baselineskip` and keeping the desired “spaces” (`\topsep` + `[\partopsep]`) it is necessary to “adjust” the “vertical spaces” for `minipage` environments.

Here there are several complications that must be circumvented: the `minipage` environment eliminates the “top” spaces, the `multicols` environment can be nested in the `minipage` environment, the “top” and “bottom” spaces are affected when `topsep=opt` and to this is added the `\partopsep` parameter that comes into action according to whether T_EX is in *(horizontal mode)* or *(vertical mode)*. Depending on these cases, the spaces are computed and added with `\addvspace` to obtain the “desired vertical spacing”.

- The “top” and “bottom” spaces cannot be read directly from the *glue* produced by the `minipage` environment, and when `multicols` is nested the depth of its last line is lost when it ends. For this reason the spaces on the “left side” are not obtained from fixed fractions of `\strutbox`: the space “above” is computed from `\prevdepth` and the current `\baselineskip`, which is the same *glue* that T_EX inserts between two lines, and the space “below” is stored together with the depth of the last line when the environment on the “left side” ends, to be applied once the environment on the “right side” is closed. In this way the spaces follow the size of the font used in the document.

13.25.1 Definition of variables

```
\l__enumext_minipage_right_skip
\l__enumext_mini_prev_dim
\l__enumext_mini_gap_dim
\l__enumext_mini_last_box
\g__enumext_mini_dp_dim
\g__enumext_mini_end_skip
\g__enumext_mini_end_bool
```

Internal variables used by the keys `mini-env` and `mini-sep` (§13.25). The variable `\l__enumext_minipage_right_skip` holds the space applied “above” the `\l__enumext_mini_page` environment on the “right side” and “below” the `multicols` environment. The variables `\l__enumext_mini_prev_dim`, `\l__enumext_mini_gap_dim` and `\l__enumext_mini_last_box` are used to compute the space “above” and to recover the last box of the `multicols` environment. The variables `\g__enumext_mini_dp_dim` and `\g__enumext_mini_end_skip` store the depth of the last line and the space “below”, while `\g__enumext_mini_end_bool` indicates that both are pending to be applied.

```
1368 \skip_new:N \l__enumext_minipage_right_skip
1369 \dim_new:N \l__enumext_mini_prev_dim
1370 \dim_new:N \l__enumext_mini_gap_dim
1371 \box_new:N \l__enumext_mini_last_box
1372 \dim_new:N \g__enumext_mini_dp_dim
1373 \skip_new:N \g__enumext_mini_end_skip
1374 \bool_new:N \g__enumext_mini_end_bool
```

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

13.25.2 Definition of error messages

```
wrong-miniright-place
wrong-miniright-use
wrong-miniright-starred
many-miniright-used
```

Messages used by `\miniright` command.

```
1375 \msg_new:nnnn { enumext } { wrong-miniright-place }
1376 {
1377   Wrong~place~for~'\token_to_str:N \miniright'\msg_line_context:.
1378 }
1379 {
1380   Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
1381 }
1382 \msg_new:nnnn { enumext } { wrong-miniright-use }
1383 {
1384   Wrong~use~for~'\token_to_str:N \miniright'\msg_line_context:.
1385 }
1386 {
1387   '\token_to_str:N \miniright'~needs~a~key~'mini-env'.
1388 }
1389 \msg_new:nnnn { enumext } { wrong-miniright-starred }
1390 {
1391   Can't~use~'\token_to_str:N \miniright'~in~starred~environments~
1392   \msg_line_context:.
```

```

1393 }
1394 {
1395   In~'enumext*'~and~'keyans*'~use~the~keys~'mini-right'~or~
1396   'mini-right*'~instead.
1397 }
1398 \msg_new:nnnn { enumext } { many-miniright-used }
1399 {
1400   Can't~use~'\token_to_str:N \miniright'~more~than~once~
1401   \msg_line_context:.
1402 }
1403 {
1404   '\token_to_str:N \miniright'~can~be~used~only~once~in~each~
1405   environment.
1406 }

```

(End of definition for wrong-miniright-place and others.)

13.25.3 Computation of the vertical spaces for minipage

`\__enumext_mini_baselineskip:`

The function `\__enumext_mini_baselineskip:` adds with `\addvspace` the space that $\text{\TeX}$ would insert between the previous line and a line of text whose height is that of the letter “l”, that is, `\baselineskip` minus `\prevdepth` minus that height, never less than `\lineskip`. If there is no previous line (`\prevdepth` is `-1000pt`) the space is zero. The `\nointerlineskip` command is executed before the space is added, to prevent the *glue* of the baseline from being added again between this space and the next box.

```

1407 \cs_new_protected:Nn \__enumext_mini_baselineskip:
1408 {
1409   \dim_set:Nn \l__enumext_mini_prev_dim { \prevdepth }
1410   \nointerlineskip
1411   \dim_compare:nNnTF { \l__enumext_mini_prev_dim } < { -999pt }
1412   { \dim_zero:N \l__enumext_mini_gap_dim }
1413   {
1414     \dim_set:Nn \l__enumext_mini_gap_dim
1415     { \baselineskip - \l__enumext_mini_prev_dim - \fontcharht\font`l }
1416     \dim_compare:nNnT { \l__enumext_mini_gap_dim } < { \lineskip }
1417     { \dim_set_eq:NN \l__enumext_mini_gap_dim \lineskip }
1418   }
1419   \addvspace{ \l__enumext_mini_gap_dim }
1420 }

```

(End of definition for `\__enumext_mini_baselineskip:`.)

`\__enumext_mini_capture_end:`

The function `\__enumext_mini_capture_end:` stores in `\g__enumext_mini_dp_dim` the value of `\prevdepth`, removes the last *glue* (if any) saving its value in `\g__enumext_mini_end_skip` and sets `\g__enumext_mini_end_bool` to `true`. The function `\__enumext_mini_capture_multicols:` does the same after a `multicols` environment, where `\prevdepth` is not reliable: the last box is recovered with `\lastbox`, its depth is stored and the box is put back, and the stored space is `\l__enumext_minipage_right_skip`. The function `\__enumext_mini_apply_end:` is called after closing the environment on the “right side”: if there is something stored, it restores `\prevdepth` and adds the stored space with `\addvspace`, so that the space “below” is measured from the last line of the environment on the “left side”.

```

1421 \cs_new_protected:Nn \__enumext_mini_capture_end:
1422 {
1423   \dim_gset:Nn \g__enumext_mini_dp_dim { \prevdepth }
1424   \skip_gzero:N \g__enumext_mini_end_skip
1425   \int_compare:nNnT { \lastnodetype } = { 11 }
1426   {
1427     \skip_gset:Nn \g__enumext_mini_end_skip { \lastskip }
1428     \unskip
1429   }
1430   \bool_gset_true:N \g__enumext_mini_end_bool
1431 }
1432 \cs_new_protected:Nn \__enumext_mini_capture_multicols:
1433 {
1434   \int_compare:nNnTF { \lastnodetype } = { 1 }
1435   {
1436     \box_set_to_last:N \l__enumext_mini_last_box
1437     \dim_gset:Nn \g__enumext_mini_dp_dim { \box_dp:N \l__enumext_mini_last_box }
1438     \nointerlineskip
1439     \box_use_drop:N \l__enumext_mini_last_box
1440   }

```

```

1441     { \dim_gset:Nn \g__enumext_mini_dp_dim { \prevdepth } }
1442     \skip_gset_eq:NN \g__enumext_mini_end_skip \l__enumext_minipage_right_skip
1443     \bool_gset_true:N \g__enumext_mini_end_bool
1444   }
1445   \cs_new_protected:Nn \__enumext_mini_apply_end:
1446   {
1447     \bool_if:NT \g__enumext_mini_end_bool
1448     {
1449       \par
1450       \dim_compare:nNnF { \g__enumext_mini_dp_dim } < { -999pt }
1451       { \dim_set_eq:NN \prevdepth \g__enumext_mini_dp_dim }
1452       \addpenalty { \@endparpenalty }
1453       \addvspace { \g__enumext_mini_end_skip }
1454       \bool_gset_false:N \g__enumext_mini_end_bool
1455     }
1456   }

```

(End of definition for `\__enumext_mini_capture_end:`, `\__enumext_mini_capture_multicols:`, and `\__enumext_mini_apply_end:`.)

13.25.4 Adjustment of vertical spaces for minipage in enumext

```

\__enumext_minipage_set_skip:
\__enumext_minipage_add_space:

```

The function `\__enumext_minipage_set_skip:` will take care of determining the value of `\l__enumext_minipage_right_skip`, the space applied “above” the `\__enumext_mini_page` environment on the “right side” and “below” the `multicols` environment in `enumext`.

First it sets the value of `\l__enumext_minipage_right_skip` equal to `\topsep`, then checks if $\TeX$ is in $\langle vertical mode \rangle$ and, if so, adds `\partopsep`.

```

1457 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1458 {
1459   \skip_set:Nn \l__enumext_minipage_right_skip
1460   {
1461     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1462   }
1463   \mode_if_vertical:T
1464   {
1465     \skip_add:Nn \l__enumext_minipage_right_skip
1466     {
1467       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1468     }
1469   }

```

Sets the value of `\l__enumext_multicols_above_X_skip` equal to `\l__enumext_minipage_right_skip`.

```

1470   \skip_set_eq:cN
1471   { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip

```

If the environment `multicols` is active, it sets `\topskip=0pt` and then makes `\multicolsep` have the same value as `\l__enumext_multicols_above_X_skip`.

```

1472   \int_compare:nNnT
1473   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1474   {
1475     \skip_zero:N \topskip
1476     \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1477   }
1478 }

```

The function `\__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `\__enumext_mini_page` environment, taking into account whether $\TeX$ is in $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$. First the last $\langle skip \rangle$ or $\langle kern \rangle$ is removed; in $\langle vertical mode \rangle$ only `\nopagebreak` is added, while in $\langle horizontal mode \rangle$ the paragraph is ended and `\partopsep` is set to zero, since this parameter comes into play according to the mode and this affects the *vertical spacing*. The space is then added with the function `\__enumext_mini_baselineskip:`.

```

1479 \cs_new_protected:Nn \__enumext_minipage_add_space:
1480 {
1481   \__enumext_minipage_set_skip:
1482   \__enumext_unskip_unkern:
1483   \mode_if_vertical:TF
1484   {
1485     \nopagebreak
1486   }
1487 }

```

```

1488         \par\nopagebreak
1489         \skip_zero:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1490     }
1491     \l__enumext_mini_baselineskip:
1492 }

```

(End of definition for \l__enumext_minipage_set_skip: and \l__enumext_minipage_add_space:.)

13.25.5 Adjustment of vertical spaces for minipage in keyans

\l__enumext_keyans_minipage_set_skip:
\l__enumext_keyans_minipage_add_space:

The function `\l__enumext_keyans_minipage_set_skip:` will take care of determining the value of `\l__enumext_minipage_right_skip` in *keyans*. The implementation of this function is the same as the one used in *enumext*.

```

1493 \cs_new_protected:Nn \l__enumext_keyans_minipage_set_skip:
1494 {
1495     \skip_zero:N \l__enumext_minipage_right_skip
1496     \skip_set:Nn \l__enumext_minipage_right_skip
1497     {
1498         \l__enumext_topsep_v_skip
1499     }
1500     \mode_if_vertical:T
1501     {
1502         \skip_add:Nn \l__enumext_minipage_right_skip
1503         {
1504             \l__enumext_partopsep_v_skip
1505         }
1506     }
1507     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1508     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1509     {
1510         \skip_zero:N \topskip
1511         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1512     }
1513 }
1514 \cs_new_protected:Nn \l__enumext_keyans_minipage_add_space:
1515 {
1516     \l__enumext_keyans_minipage_set_skip:
1517     \l__enumext_unskip_unkern:
1518     \mode_if_vertical:TF
1519     {
1520         \nopagebreak
1521     }
1522     {
1523         \par\nopagebreak
1524         \skip_zero:N \l__enumext_partopsep_v_skip
1525     }
1526     \l__enumext_mini_baselineskip:
1527 }

```

(End of definition for \l__enumext_keyans_minipage_set_skip: and \l__enumext_keyans_minipage_add_space:.)

13.25.6 Adjustment of vertical spaces for minipage in enumext* and keyans*

\l__enumext_mini_set_vskip_vii:
\l__enumext_mini_set_vskip_viii:

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the space applied “above” the `\l__enumext_mini_page` environment on the “right side” in *enumext** and *keyans**. It is stored in `\g__enumext_minipage_right_skip` and has the same value as the `\topsep` of the environment, which is the space that the list adds above the environment on the “left side”.

```

1528 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1529 {
1530     \skip_gset:Nn \g__enumext_minipage_right_skip { \l__enumext_topsep_vii_skip }
1531 }
1532 \cs_new_protected:Nn \l__enumext_mini_set_vskip_viii:
1533 {
1534     \skip_gset:Nn \g__enumext_minipage_right_skip { \l__enumext_topsep_viii_skip }
1535 }

```

(End of definition for \l__enumext_mini_set_vskip_vii: and \l__enumext_mini_set_vskip_viii:.)

\l__enumext_mini_addvspace_vii:
\l__enumext_mini_addvspace_viii:

The functions `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\l__enumext_mini_page` environment on the left side

when the `mini-right` key is active in the `enumext*` and `keyans*` environments, using the function `\__enumext_mini_baselineskip:`.

Here it does NOT take into account whether $\text{\TeX}$ is in *(horizontal mode)* or *(vertical mode)*, since `\partopsep` is equal to `\opt` in both environments.

```

1536 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1537 {
1538   \__enumext_mini_set_vskip_vii:
1539   \par\nopagebreak
1540   \__enumext_mini_baselineskip:
1541 }
1542 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1543 {
1544   \__enumext_mini_set_vskip_viii:
1545   \par\nopagebreak
1546   \__enumext_mini_baselineskip:
1547 }
```

(End of definition for `\__enumext_mini_addvspace_vii:` and `\__enumext_mini_addvspace_viii:`)

13.25.7 The command `\miniright`

The command `\miniright` will close the `\__enumext_mini_page` environment on the “left side”, open the `\__enumext_mini_page` environment on the “right side” adding the *adjusted vertical space*. By default `\centering` is added when starting the “right side” environment. The *starred argument* ‘*’ inhibits the use of `\centering` command i.e. the usual $\text{\TeX}$ justification is maintained in the `\__enumext_mini_page` on the “right side”.

`\miniright` The command accepts a *starred argument* ‘*’ in ‘#1’. First, some checks prevent the command from being executed outside the `enumext` environment or somewhere inappropriate. Only the first error found is reported and the command does nothing else. Then the internal functions are called to execute it in the `enumext` and `keyans` environments.

```

1548 \NewDocumentCommand \miniright { s }
1549 {
1550   % outside
1551   \bool_lazy_or:nnTF
1552   { \int_compare_p:nNn { \__enumext_keyans_pic_level_int } = { 1 } }
1553   {
1554     \bool_lazy_and_p:nn
1555     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1556     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1557   }
1558   {
1559     \msg_error:nn { enumext } { wrong-miniright-place }
1560   }
1561   {
1562     % starred env
1563     \bool_lazy_and:nnTF
1564     { \bool_if_p:N \g__enumext_starred_bool }
1565     { \bool_not_p:n { \__enumext_standard_bool } }
1566     {
1567       \msg_error:nn { enumext } { wrong-miniright-starred }
1568     }
1569     {
1570       % exec
1571       \int_compare:nNnTF { \__enumext_keyans_level_int } = { 1 }
1572       {
1573         \__enumext_keyans_mini_right_cmd:n {#1}
1574       }
1575       { \__enumext_mini_right_cmd:n {#1} }
1576     }
1577   }
1578 }
```

(End of definition for `\miniright`. This function is documented on page 13.)

`\__enumext_mini_right_cmd:n` The function `\__enumext_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `enumext` environment. It checks if the `mini-env` key is active via the variable `\__enumext_minipage_right_X_dim`; if so, it closes the `multicols` environment with the `\__enumext_mini_page` environment on the “left side”, then opens the `\__enumext_mini_page` environment on the “right side”, applies the adjusted “vertical space” above it, followed by adding the `\centering`

command when the *starred argument* ‘*’ is not present, and sets `\g__enumext_minipage_stat_int` to zero; otherwise it returns an error. Before closing the environment on the “*left side*” the space “*below*” is stored with `\__enumext_mini_capture_end:` (or with `\__enumext_mini_capture_multicols:` when `multicols` is active), to be applied after the environment on the “*right side*”.

```

1579 \cs_new_protected:Nn \__enumext_mini_right_cmd:n
1580 {
1581   \dim_compare:nNnTF
1582     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1583     {
1584       \__enumext_multicols_stop:
1585       \int_compare:nNnT
1586         { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } = { 1 }
1587         {
1588           \par
1589           \__enumext_mini_capture_end:
1590         }
1591       \end__enumext_mini_page
1592       \hfill
1593       \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1594         \par\nointerlineskip
1595         \addvspace { \__enumext_minipage_right_skip }
1596         \bool_if:nF {#1}
1597         {
1598           \centering
1599         }
1600       \int_gzero:N \g__enumext_minipage_stat_int
1601     }
1602     { \msg_error:nn { enumext } { wrong-miniright-use } }
1603   % paranoia
1604   \RenewDocumentCommand \miniright { s }
1605   {
1606     \msg_error:nn { enumext } { many-miniright-used }
1607   }
1608 }

```

(End of definition for `\__enumext_mini_right_cmd:n`.)

`\__enumext_keyans_mini_right_cmd:n`

The function `\__enumext_keyans_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `\__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1609 \cs_new_protected:Nn \__enumext_keyans_mini_right_cmd:n
1610 {
1611   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1612   {
1613     \__enumext_keyans_multicols_stop:
1614     \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1615     {
1616       \par
1617       \__enumext_mini_capture_end:
1618     }
1619     \end__enumext_mini_page
1620     \hfill
1621     \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1622       \par\nointerlineskip
1623       \addvspace { \__enumext_minipage_right_skip }
1624       \bool_if:nF {#1}
1625       {
1626         \centering
1627       }
1628     \int_gzero:N \g__enumext_minipage_stat_int
1629   }
1630   { \msg_error:nn { enumext } { wrong-miniright-use } }
1631   % paranoia
1632   \RenewDocumentCommand \miniright { s }
1633   {
1634     \msg_error:nn { enumext } { many-miniright-used }
1635   }
1636 }

```

(End of definition for `\__enumext_keyans_mini_right_cmd:n`.)

13.26 Setting above and below keys

While having controlled the *vertical spaces* within the `enumext` and `keyans` environments when using the `columns` or `mini-env` keys, sometimes the “vertical spaces above” or “vertical spaces below” the environments are not as expected and it is necessary to be able to apply a “fine correction” to these. Since these *glitches* cannot be corrected completely, the best option is to provide a couple of (*keys*) dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

```
\l__enumext_vspace_a_star_X_bool
\l__enumext_vspace_b_star_X_bool
```

The boolean vars `\l__enumext_vspace_a_star_X_bool` and `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys.

```
1637 \cs_set_protected:Nn \__enumext_tmp:n
1638 {
1639   \bool_new:c { \l__enumext_vspace_a_star_ #1 _bool }
1640   \bool_new:c { \l__enumext_vspace_b_star_ #1 _bool }
1641 }
1642 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_vspace_a_star_X_bool` and `\l__enumext_vspace_b_star_X_bool`.)

Define `above`, `above*`, `below` and `below*` keys for `enumext` and `keyans` environments.

```
above
above*
below
below*
1643 \cs_set_protected:Nn \__enumext_tmp:nn
1644 {
1645   \keys_define:nn { enumext / #1 }
1646   {
1647     above .skip_set:c = { \l__enumext_vspace_above_ #2 _skip },
1648     above .value_required:n = true,
1649     above* .code:n =
1650     {
1651       \bool_set_true:c { \l__enumext_vspace_a_star_ #2 _bool }
1652       \keys_set:nn { enumext / #1 } { above = {##1} }
1653     },
1654     above* .value_required:n = true,
1655     below .skip_set:c = { \l__enumext_vspace_below_ #2 _skip },
1656     below .value_required:n = true,
1657     below* .code:n =
1658     {
1659       \bool_set_true:c { \l__enumext_vspace_b_star_ #2 _bool }
1660       \keys_set:nn { enumext / #1 } { below = {##1} }
1661     },
1662     below* .value_required:n = true,
1663   }
1664 }
1665 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

(End of definition for `above` and others.)

13.26.1 Functions for above and below keys in enumext

`\__enumext_vspace_above:` The function `\__enumext_vspace_above:` applies the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```
1666 \cs_new_protected:Nn \__enumext_vspace_above:
1667 {
1668   \skip_if_eq:nnF
1669   { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1670   {
1671     \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1672     {
1673       \vspace*{ \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1674     }
1675     {
1676       \vspace { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1677     }
1678   }
1679 }
```

(End of definition for `\__enumext_vspace_above:`.)

`\__enumext_vspace_below:` The function `\__enumext_vspace_below:` applies the *vertical space below* the `enumext` environment set by the `below*` and `below` keys.

```
1680 \cs_new_protected:Nn \__enumext_vspace_below:
```

```

1681 {
1682   \skip_if_eq:nnF
1683   { \skip_use:c { \l__enumext_vspace_below_ \l__enumext_level: _skip } } { \c_zero_skip }
1684   {
1685     \bool_if:cTF { \l__enumext_vspace_b_star_ \l__enumext_level: _bool }
1686     {
1687       \vspace*{ \skip_use:c { \l__enumext_vspace_below_ \l__enumext_level: _skip } }
1688     }
1689     {
1690       \vspace { \skip_use:c { \l__enumext_vspace_below_ \l__enumext_level: _skip } }
1691     }
1692   }
1693 }

```

(End of definition for \l__enumext_vspace_below:.)

13.26.2 Functions for above and below keys in keyans

\l__enumext_vspace_above_v: The function \l__enumext_vspace_above_v: applies the *vertical space above* the **keyans** environment set by the **above** and **above*** keys.

```

1694 \cs_new_protected:Nn \l__enumext_vspace_above_v:
1695 {
1696   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1697   {
1698     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1699     {
1700       \vspace*{ \l__enumext_vspace_above_v_skip }
1701     }
1702     { \vspace { \l__enumext_vspace_above_v_skip } }
1703   }
1704 }

```

(End of definition for \l__enumext_vspace_above_v:.)

\l__enumext_vspace_below_v: The function \l__enumext_vspace_below_v: applies the *vertical space below* the **keyans** environment set by the **below*** and **below** keys.

```

1705 \cs_new_protected:Nn \l__enumext_vspace_below_v:
1706 {
1707   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1708   {
1709     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1710     {
1711       \vspace*{ \l__enumext_vspace_below_v_skip }
1712     }
1713     { \vspace { \l__enumext_vspace_below_v_skip } }
1714   }
1715 }

```

(End of definition for \l__enumext_vspace_below_v:.)

13.26.3 Functions for above and below keys in enumext* keyans*

\l__enumext_vspace_above_vii: The functions \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii: apply the *vertical space above* the **enumext*** and **keyans*** environments set by the **above** and **above*** keys.

\l__enumext_vspace_above_viii:

```

1716 \cs_new_protected:Nn \l__enumext_vspace_above_vii:
1717 {
1718   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1719   {
1720     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1721     {
1722       \vspace*{ \l__enumext_vspace_above_vii_skip }
1723     }
1724     { \vspace { \l__enumext_vspace_above_vii_skip } }
1725   }
1726 }
1727 \cs_new_protected:Nn \l__enumext_vspace_above_viii:
1728 {
1729   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1730   {
1731     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1732     {

```

```

1733         \vspace*{ \l__enumext_vspace_above_viii_skip }
1734     }
1735     { \vspace { \l__enumext_vspace_above_viii_skip } }
1736 }
1737 }

```

(End of definition for \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii:.)

```

\l__enumext_vspace_below_vii:
\l__enumext_vspace_below_viii:

```

The functions `\l__enumext_vspace_below_vii:` and `\l__enumext_vspace_below_viii:` apply the *vertical space below* the `enumext*` and `keyans*` environments set by the `below*` and `below` keys.

```

1738 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1739 {
1740     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1741     {
1742         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1743         {
1744             \vspace*{ \l__enumext_vspace_below_vii_skip }
1745         }
1746         { \vspace { \l__enumext_vspace_below_vii_skip } }
1747     }
1748 }
1749 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1750 {
1751     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1752     {
1753         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1754         {
1755             \vspace*{ \l__enumext_vspace_below_viii_skip }
1756         }
1757         { \vspace { \l__enumext_vspace_below_viii_skip } }
1758     }
1759 }

```

(End of definition for \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii:.)

13.27 Setting series, resume and resume* keys

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the *(keys)* passed to the *optional argument* of the environments `enumext` and `enumext*`, but, discarding some specific *(keys)*.

13.27.1 Definition of variables

```

\l__enumext_series_name_tl
\l__enumext_resume_series_X_bool
\l__enumext_resume_star_key_X_bool
\g__enumext_resume_last_keys_X_tl

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.27).

```

1760 \tl_new:N \l__enumext_series_name_tl
1761 \cs_set_protected:Nn \l__enumext_tmp:n
1762 {
1763     \bool_new:c { \l__enumext_resume_series_ #1 _bool }
1764     \bool_new:c { \l__enumext_resume_star_key_ #1 _bool }
1765     \tl_new:c { \g__enumext_resume_last_keys_ #1 _tl }
1766 }
1767 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }

```

(End of definition for \l__enumext_series_name_tl and others.)

13.27.2 Definition of error messages

```

unknown-series-starred
unknown-series-standard

```

Messages used by `resume` key.

```

1768 \msg_new:nnnn { enumext } { unknown-series-starred }
1769 {
1770     The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
1771     enumext*~environment~\msg_line_context:.
1772 }
1773 {
1774     Define~the~series~with~the~key~'series'~in~a~previous~
1775     'enumext*'~environment~and~check~the~name.
1776 }
1777 \msg_new:nnnn { enumext } { unknown-series-standard }
1778 {
1779     The~series~'#1'~for~the~resume~key~does~not~exist~at~level~
1780     \int_use:N \l__enumext_level_int

```

```

1781     \c_space_tl of~enumext~environment~\msg_line_context:.
1782   }
1783   {
1784     Define~the~series~with~the~key~'series'~at~that~level~in~
1785     a~previous~'enumext'~environment~and~check~the~name.
1786   }

```

(End of definition for *unknown-series-starred* and *unknown-series-standard*.)

13.27.3 Definition of keys for series, resume and resume*

series Defines the keys **series**, **resume** and **resume*** for the “all levels” of **enumext** and **enumext***. Both keys do nothing while **\printkeyans** is running, otherwise the *start value* of the environments would be increased when using **resume** or **resume*** keys.

In version 1.6 it is allowed to pass the key **resume** *without value* by means of the command **\setenumext**, for the correct operation of this, the boolean variable **__enumext_resume_count_bool** set by the key **resume** *without value* must be set to **true** to be later processed by the function **__enumext_parse_series:n** in the definition of the environments **enumext** and **enumext***.

```

1787 \cs_set_protected:Nn \__enumext_tmp:nn
1788 {
1789   \keys_define:nn { enumext / #1 }
1790   {
1791     series .str_set:N = \__enumext_series_name_str,
1792     series .value_required:n = true,
1793     resume .code:n =
1794     {
1795       \bool_if:NF \__enumext_print_keyans_cmd_bool
1796       {
1797         \tl_set:Nn \__enumext_series_name_tl {##1}
1798         \tl_if_empty:NTF \__enumext_series_name_tl
1799         {
1800           \bool_set_true:c { \__enumext_resume_count_ #2 _bool }
1801           \bool_set_eq:Nc
1802             \__enumext_resume_count_bool
1803             { \__enumext_resume_count_ #2 _bool }
1804         }
1805         {
1806           \__enumext_resume:n {##1}
1807         }
1808       }
1809     },
1810     resume* .code:n =
1811     {
1812       \bool_if:NF \__enumext_print_keyans_cmd_bool
1813       {
1814         \bool_set_true:c { \__enumext_resume_count_ #2 _bool }
1815         \bool_set_eq:Nc
1816           \__enumext_resume_count_bool { \__enumext_resume_count_ #2 _bool }
1817         \bool_set_true:c { \__enumext_resume_star_key_ #2 _bool }
1818         \__enumext_resume_star:
1819       }
1820     },
1821     resume* .value_forbidden:n = true,
1822   }
1823 }
1824 \clist_map_inline:nn
1825 {
1826   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},{enumext*}{vii},
1827 }
1828 { \__enumext_tmp:nn #1 }

```

(End of definition for *series*, *resume*, and *resume**.)

13.27.4 Internal function to save counter and integer values

__enumext_standard_save_counter: The **__enumext_standard_save_counter:** and **__enumext_starred_save_counter:** functions will save the last counter value to **\g__enumext_series_#1_X_int** with the name given to the key **series={⟨series name⟩}** if it has been passed, to **\c@__enumext_resume_X_int** if none of the keys **series**, **resume={⟨series name⟩}** and **save-ans** has been passed (the case of the key **resume** *without value*), to **\g__enumext_series_#1_X_int** with the name given to the key **resume={⟨series name⟩}** if it has

been passed and, in the “*first level*”, to `\g__enumext_resume_#1_int` with the `{\store name}` if the key `save-ans={\store name}` has been passed.

- The variables `\l__enumext_series_name_str` and `\l__enumext_series_name_tl` contain the same `{\series name}` but are executed at different moments, the integer variable with `\l__enumext_series_name_str` sets the value when executing `series={\series name}` and the integer variable with `\l__enumext_series_name_tl` sets the subsequent values when using `resume={\series name}`. This function is passed to the `enumext` environment definition (§13.43) and the `enumext*` environment definition (§13.48).

```

1829 \cs_new_protected:Nn \__enumext_standard_save_counter:
1830 {
1831     \bool_if:NTF \g__enumext_standard_bool
1832     {
1833         \__enumext_standard_save_counter_aux:
1834         \int_compare:nNnT { \l__enumext_level_int } = { 1 }
1835         {
1836             \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1837             {
1838                 \int_gset_eq:cN
1839                 { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1840             }
1841         }
1842     }
1843     {
1844         \__enumext_standard_save_counter_aux:
1845     }
1846 }
1847 \cs_new_protected:Nn \__enumext_standard_save_counter_aux:
1848 {
1849     \str_if_empty:NF \l__enumext_series_name_str
1850     {
1851         \int_gset_eq:cc
1852         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
1853         { c@enumX \__enumext_level: }
1854     }
1855     \tl_if_empty:NTF \l__enumext_series_name_tl
1856     {
1857         \str_if_empty:NT \l__enumext_series_name_str
1858         {
1859             \tl_if_empty:NT \l__enumext_store_name_tl
1860             {
1861                 \int_gset_eq:cc
1862                 { c@__enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
1863             }
1864         }
1865     }
1866     {
1867         \int_if_exist:cT
1868         { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1869         {
1870             \int_gset_eq:cc
1871             { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1872             { c@enumX \__enumext_level: }
1873         }
1874     }
1875 }
1876 \cs_new_protected:Nn \__enumext_starred_save_counter:
1877 {
1878     \bool_if:NTF \g__enumext_starred_bool
1879     {
1880         \__enumext_starred_save_counter_aux:
1881         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1882         {
1883             \int_gset_eq:cN
1884             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1885         }
1886     }
1887     {
1888         \__enumext_starred_save_counter_aux:
1889     }
1890 }
1891 \cs_new_protected:Nn \__enumext_starred_save_counter_aux:

```

```

1892 {
1893   \str_if_empty:NF \l__enumext_series_name_str
1894   {
1895     \int_gset_eq:cN
1896     { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
1897   }
1898   \tl_if_empty:NTF \l__enumext_series_name_tl
1899   {
1900     \str_if_empty:NT \l__enumext_series_name_str
1901     {
1902       \tl_if_empty:NT \l__enumext_store_name_tl
1903       {
1904         \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
1905       }
1906     }
1907   }
1908   {
1909     \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
1910     {
1911       \int_gset_eq:cN
1912       { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
1913     }
1914   }
1915 }

```

(End of definition for `\__enumext_standard_save_counter`: and others.)

13.27.5 Internal function for resume counters

`\__enumext_resume_counter`: The `\__enumext_resume_counter`: function is executed by the `resume*` key and `resume` key *without value*, only the “*counters*” for the “*levels*” of the environments in which it is executed will be set. If the `save-ans` key is active it will set the “*counter*” according to the value of the integer variable created by that key.

```

1916 \cs_new_protected:Nn \__enumext_resume_counter:
1917 {
1918   \cs_set:Nn \__enumext_tmp:n
1919   {
1920     \bool_if:cT { l__enumext_resume_count_ \int_to_roman:n {##1} _bool }
1921     {
1922       \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
1923       {
1924         \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
1925       }
1926     }
1927   }
1928   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1929   {
1930     \bool_lazy_and:nnTF
1931     { \bool_if_p:N \l__enumext_standard_first_bool }
1932     { \bool_if_p:N \l__enumext_store_active_bool }
1933     {
1934       \int_set:Nn \l__enumext_start_i_int
1935       {
1936         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1937       }
1938     }
1939     {
1940       \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
1941     }
1942   }
1943   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
1944   {
1945     \bool_lazy_and:nnTF
1946     { \bool_if_p:N \l__enumext_starred_first_bool }
1947     { \bool_if_p:N \l__enumext_store_active_bool }
1948     {
1949       \int_set:Nn \l__enumext_start_vii_int
1950       {
1951         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1952       }
1953     }
1954   }

```

```

1955         \bool_if:NT \l__enumext_resume_count_vii_bool
1956         {
1957             \int_set:Nn \l__enumext_start_vii_int
1958             {
1959                 \int_use:c { c@ __enumext_resume_vii_int } + 1
1960             }
1961         }
1962     }
1963 }
1964 }

```

(End of definition for `\__enumext_resume_counter:`.)

13.27.6 Internal functions for series key

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn

```

The function `\__enumext_filter_series:n` will be in charge of filtering the *(keys)* to be stored, where ‘#1’ represents the *optional argument* passed to the environment. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in [chat-Tex-SX](#)

```

1965 \cs_new:Nn \__enumext_filter_series:n
1966 {
1967     \use:e
1968     {
1969         \keyval_parse:NNn
1970         \__enumext_filter_series_key:n
1971         \__enumext_filter_series_pair:nn {#1}
1972     }
1973 }

```

The function `\__enumext_filter_series_key:n` will be responsible for filtering the *(keys)* that are passed “without value” by excluding the `resume`, `resume*`, `reset`, `reset*` and `base-fix` keys.

```

1974 \cs_new:Nn \__enumext_filter_series_key:n
1975 {
1976     \str_case:nnF {#1}
1977     {
1978         { resume } { } { resume* } { } { reset } { } { reset* } { }
1979         { base-fix } { }
1980     }
1981     { , { \exp_not:n {#1} } }
1982 }

```

The function `\__enumext_filter_series_pair:nn` will be responsible for filtering the *(keys)* that are passed “with value” by excluding the `series`, `resume`, `start`, `start*`, `save-ans` and `save-key` keys.

```

1983 \cs_new:Nn \__enumext_filter_series_pair:nn
1984 {
1985     \str_case:nnF {#1}
1986     {
1987         { series } { } { resume } { } { start } { }
1988         { start* } { } { save-ans } { } { save-key } { }
1989     }
1990     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1991 }

```

(End of definition for `\__enumext_filter_series:n`, `\__enumext_filter_series_key:n`, and `\__enumext_filter_series_pair:nn`.)

```

\__enumext_save_last_keys:n
  \__enumext_resume_last_counter:

```

The function `\__enumext_save_last_keys:n` will be in charge of saving the filtering *(keys)* when the keys `series={ (series name) }` or `resume={ (series name) }` or `resume*` are NOT active and will save them in the variable `\g__enumext_resume_last_keys_X_tl` for the `enumext` environment and in the variable `\g__enumext_resume_last_keys_vii_tl` for the `enumext*` environment.

- 🌱 The boolean variable `\l__enumext_resume_series_X_bool` is set to `true` by the key `resume={ (series name) }`, the boolean variable `\l__enumext_resume_star_key_X_bool` is set to `true` by the key `resume*`, in this case both variables must be set to `false` so that they don’t override the default filtered *(keys)*.

```

1992 \cs_new_protected:Nn \__enumext_save_last_keys:n
1993 {
1994     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1995     {
1996         \bool_if:cF { \l__enumext_resume_series_ \__enumext_level: _bool }
1997         {
1998             \bool_if:cF { \l__enumext_resume_star_key_ \__enumext_level: _bool }

```

```

1999         {
2000             \tl_gclear:c { g__enumext_resume_last_keys_ \__enumext_level: _tl }
2001             \tl_gset:ce
2002                 { g__enumext_resume_last_keys_ \__enumext_level: _tl }
2003                 { \__enumext_filter_series:n {#1} }
2004         }
2005     }
2006 }
2007 \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2008 {
2009     \bool_if:NF \l__enumext_resume_series_vii_bool
2010     {
2011         \tl_gclear:N \g__enumext_resume_last_keys_vii_tl
2012         \tl_gset:Ne \g__enumext_resume_last_keys_vii_tl
2013             { \__enumext_filter_series:n {#1} }
2014     }
2015 }
2016 }

```

The `\__enumext_resume_last_counter:` function will be in charge of setting the “*counters*” when the keys `series={⟨series name⟩}` or `resume={⟨series name⟩}` are NOT active and the `resume` key is being used *without value* either in the *optional argument* of the environments or through the `\setenumext` command.

- The boolean variable `\l__enumext_resume_count_bool` is set to `true` by the keys `resume without value` and `resume*`; and set to `false` by the keys `start` and `start*` (§13.15).

```

2017 \cs_new_protected:Nn \__enumext_resume_last_counter:
2018 {
2019     \bool_lazy_and:nnT
2020         { \bool_if_p:N \l__enumext_resume_count_bool }
2021         { \tl_if_empty_p:N \l__enumext_series_name_tl }
2022     {
2023         \__enumext_resume_counter:
2024     }
2025 }

```

(End of definition for `\__enumext_save_last_keys:n` and `\__enumext_resume_last_counter:`)

`\__enumext_parse_series:n` The `\__enumext_parse_series:n` function handled by the `series` key will be responsible for storing the *filtered ⟨keys⟩* from the *optional arguments* of the `enumext` and `enumext*` environments for the `resume` and `resume*` keys. If the `series` key is NOT active it will call the `\__enumext_save_last_keys:n` function to store the *filtered ⟨keys⟩* that will be used by the `resume*` key and then the `\__enumext_resume_last_counter:` function used by the `resume` key *without value* if it is active, otherwise store the *filtered ⟨keys⟩* in the global variable `\g__enumext_series_#1_X_tl` along with the creation of the integer variable `\g__enumext_series_#1_X_int` used by the `resume` key *with value*.

- This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§13.43) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.48).

```

2026 \cs_new_protected:Nn \__enumext_parse_series:n
2027 {
2028     \str_if_empty:NTF \l__enumext_series_name_str
2029     {
2030         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2031         {
2032             \__enumext_save_last_keys:n {#1}
2033             \__enumext_resume_last_counter:
2034         }
2035         \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2036         {
2037             \__enumext_save_last_keys:n {#1}
2038             \__enumext_resume_last_counter:
2039         }
2040     }
2041     {
2042         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2043         {
2044             \tl_gclear_new:c
2045                 { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2046             \tl_gset:ce
2047                 { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2048                 { \__enumext_filter_series:n {#1} }
2049             \int_if_exist:cF

```

```

2050         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2051     {
2052         \int_new:c
2053         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2054     }
2055 }
2056 \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2057 {
2058     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2059     \tl_gset:ce
2060     { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2061     { \__enumext_filter_series:n {#1} }
2062     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2063     {
2064         \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2065     }
2066 }
2067 }
2068 }

```

(End of definition for __enumext_parse_series:n.)

13.27.7 Internal functions for resume key with value

__enumext_resume:n

The function __enumext_resume:n will handle the *argument* {<series name>} passed to the resume key in enumext and enumext* environments. First it checks if the global variable \g__enumext_series_#1_X_tl exists; if so, it calls the function __enumext_resume_series:n and passes the <keys> stored in \g__enumext_series_#1_X_tl to the environments, otherwise it returns an error.

```

2069 \cs_new_protected:Nn \__enumext_resume:n
2070 {
2071     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2072     {
2073         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2074         {
2075             \__enumext_resume_series:n {#1}
2076             \exp_args:Ne \keys_set:nv { enumext /
2077                 level-\int_use:N \l__enumext_level_int }
2078             { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2079         }
2080         {
2081             \msg_error:nnn { enumext } { unknown-series-standard } {#1}
2082         }
2083     }
2084     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2085     {
2086         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2087         {
2088             \__enumext_resume_series:n {#1}
2089             \keys_set:nv { enumext / enumext* }
2090             { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2091         }
2092         {
2093             \msg_error:nnn { enumext } { unknown-series-starred } {#1}
2094         }
2095     }
2096 }

```

(End of definition for __enumext_resume:n.)

__enumext_resume_series:n
 __enumext_resume_integer_series:

The function __enumext_resume_series:n will set the variable \l__enumext_resume_series_X_bool to true and pass the {<argument>} to the variable \l__enumext_series_name_tl then call the function __enumext_resume_integer_series:.

```

2097 \cs_new_protected:Nn \__enumext_resume_series:n
2098 {
2099     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2100     {
2101         \bool_set_true:c { l__enumext_resume_series_ \__enumext_level: _bool }
2102         \tl_clear:N \l__enumext_series_name_tl
2103         \tl_set:Nn \l__enumext_series_name_tl {#1}
2104         \__enumext_resume_integer_series:

```

```

2105     }
2106     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2107     {
2108         \bool_set_true:N \l__enumext_resume_series_vii_bool
2109         \tl_clear:N \l__enumext_series_name_tl
2110         \tl_set:Nn \l__enumext_series_name_tl {#1}
2111         \__enumext_resume_integer_series:
2112     }
2113 }

```

The function `\__enumext_resume_integer_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the *start value* for the “counter” of the “current level” of the environments in which it is run according to the value of the “integer variables” created by the `series` key. If the `save-ans` key is active it will set the *start value* for the “counter” according to the value of the integer variable created by that key.

```

2114 \cs_new_protected:Nn \__enumext_resume_integer_series:
2115 {
2116     \cs_set:Nn \__enumext_tmp:n
2117     {
2118         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _
2119             \int_to_roman:n {##1} _int }
2120         {
2121             \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2122             {
2123                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _
2124                     \int_to_roman:n {##1} _int } + 1
2125             }
2126         }
2127     }
2128     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2129     {
2130         \bool_lazy_and:nnTF
2131         { \bool_if_p:N \l__enumext_standard_first_bool }
2132         { \bool_if_p:N \l__enumext_store_active_bool }
2133         {
2134             \int_set:Nn \l__enumext_start_i_int
2135             {
2136                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2137             }
2138         }
2139         {
2140             \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
2141         }
2142     }
2143     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2144     {
2145         \bool_lazy_and:nnTF
2146         { \bool_if_p:N \l__enumext_starred_first_bool }
2147         { \bool_if_p:N \l__enumext_store_active_bool }
2148         {
2149             \int_set:Nn \l__enumext_start_vii_int
2150             {
2151                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2152             }
2153         }
2154         {
2155             \int_set:Nn \l__enumext_start_vii_int
2156             {
2157                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
2158             }
2159         }
2160     }
2161 }

```

(End of definition for `\__enumext_resume_series:n` and `\__enumext_resume_integer_series:.`)

13.27.8 Internal function for `resume*` key

`\__enumext_resume_star:` The function `\__enumext_resume_star:` will handle the `resume*` key in the `enumext` and `enumext*` environments. This function will execute the filtered `⟨keys⟩` in the last one and will continue with the numbering and `⟨keys⟩` according to the last execution of the environment `enumext` or `enumext*` in which the keys `resume={⟨series name⟩}` or `series={⟨series name⟩}` were NOT active.

```

2162 \cs_new_protected:Nn \__enumext_resume_star:

```

```

2163 {
2164   \cs_set:Nn \__enumext_tmp:n
2165   {
2166     \tl_if_empty:cTF { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
2167     {
2168       \__enumext_resume_counter:
2169     }
2170     {
2171       \__enumext_resume_counter:
2172       \exp_args:Ne \keys_set:nv
2173       { enumext / level-\int_use:N \l__enumext_level_int }
2174       { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
2175     }
2176   }
2177   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2178   {
2179     \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
2180   }
2181   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2182   {
2183     \tl_if_empty:NTF g__enumext_resume_last_keys_vii_tl
2184     {
2185       \__enumext_resume_counter:
2186     }
2187     {
2188       \__enumext_resume_counter:
2189       \keys_set:nV { enumext / enumext* } \g__enumext_resume_last_keys_vii_tl
2190     }
2191   }
2192 }

```

(End of definition for `\__enumext_resume_star:.`)

13.28 The `\resetenumext` command

Sometimes it is necessary to be able to reset the “*counters*” of the environments according to some value, for example `\chapter`. Since “*internal counters*” are used for the `resume` and `resume*` keys which set the *start value*, but are not accessible by the user, it is necessary to provide a public command for this. This implementation is an adaptation of the answers given by Clea F. Rees (@cfr) and Jonathan P. Spratte (@Skillmon) in [Correct implementation of optional argument \(comma-separated\) in expl3](#).

out-of-range Message used by `\resetenumext` command.

```

2193 \msg_new:nnnn { enumext } { out-of-range }
2194 {
2195   The~number~must~be~exactly~1,~2,~3~or~4.
2196 }
2197 {
2198   Received:~'!#1'.
2199 }

```

(End of definition for `out-of-range`.)

Internal functions used by the `\resetenumext` command.

```

2200 \cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
2201 {
2202   \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } {%#2}
2203 }
2204 \cs_new_protected:Nn \__enumext_starred_reset:n
2205 {
2206   \__enumext_reset_count_resume:nn { vii } {#1}
2207 }
2208 \cs_new_protected:Nn \__enumext_reset_count_resume:nn
2209 {
2210   \counterwithin*{enumX #1}{#2}
2211   \counterwithin*{__enumext_resume_ #1 _int}{#2}
2212 }
2213 \cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
2214 \cs_new_protected:Nn \__enumext_reset_count_resume_all:n
2215 {
2216   \clist_map_inline:nn { i,ii,iii,iv,vii }

```

```

2217     {
2218         \__enumext_reset_count_resume:nn {##1} {#1}
2219     }
2220 }
2221 \cs_new_protected:Nn \__enumext_reset_count_resume_levels:n
2222 {
2223     \clist_map_inline:nn { i,ii,iii,iv }
2224     {
2225         \__enumext_reset_count_resume:nn {##1} {#1}
2226     }
2227 }

```

(End of definition for `\__enumext_standard_reset:nn` and others.)

`\resetenumext` The `\resetenumext` command “resets” the *start value* of the “counters” for the `enumext` and `enumext*` environments along with the “internal counters” used by the keys `resume` *without value* and `resume*` according to the value of $\langle \textit{some counter} \rangle$. It accepts a *starred argument* ‘`*`’ in ‘`#1`’, an optional argument in ‘`#2`’ with the level from 1 to 4 or the character ‘`*`’, and the mandatory $\langle \textit{some counter} \rangle$ in ‘`#3`’.

```

2228 \NewDocumentCommand \resetenumext { s o m }
2229 {
2230     \bool_if:nTF {#1}
2231     {
2232         \__enumext_reset_count_resume_all:n {#3}
2233     }
2234     {
2235         \tl_if_novalue:nTF {#2}
2236         {
2237             \__enumext_reset_count_resume_levels:n {#3}
2238         }
2239         {
2240             \str_if_eq:nnTF {#2} { * }
2241             { \__enumext_starred_reset:n {#3} }
2242             {
2243                 \bool_lazy_and:nnTF
2244                 { \int_compare_p:nNn {#2} > { 0 } }
2245                 { \int_compare_p:nNn {#2} < { 5 } }
2246                 { \__enumext_standard_reset:nn {#2} {#3} }
2247                 {
2248                     \msg_error:nne { enumext } { out-of-range }
2249                     { \int_eval:n {#2} }
2250                 }
2251             }
2252         }
2253     }
2254 }

```

(End of definition for `\resetenumext`. This function is documented on page 12.)

13.29 The reset and reset* keys

The `\resetenumext` command does not work, for example, after an unnumbered chapter, so it is preferable to provide a pair of $\langle \textit{keys} \rangle$ that adjust the internal variables if necessary.

`reset` Defines the keys `reset` and `reset*` for the “all levels” of `enumext` and `enumext*`.
`reset*`

```

2255 \cs_set_protected:Nn \__enumext_tmp:n
2256 {
2257     \keys_define:nn { enumext / #1 }
2258     {
2259         reset .code:n = { \__enumext_standard_reset_key: },
2260         reset .value_forbidden:n = true,
2261         reset* .code:n = { \__enumext_standard_reset_key_star: },
2262         reset* .value_forbidden:n = true,
2263     }
2264 }
2265 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
2266 \keys_define:nn { enumext / enumext* }
2267 {
2268     reset .code:n = { \__enumext_starred_reset_key: },
2269     reset .value_forbidden:n = true,
2270     reset* .code:n = { \__enumext_starred_reset_key: },

```

```

2271     reset* .value_forbidden:n = true,
2272 }

```

(End of definition for `reset` and `reset*`.)

13.29.1 Internal functions for `reset` and `reset*` keys

```

__enumext_standard_reset_key:
__enumext_standard_reset_key_star:
__enumext_starred_reset_key:

```

The function `__enumext_standard_reset_key:` will be handled by the `reset` key and will “reset” the counter `\c@__enumext_resume_X_int` to “zero” according to the *level* at which it is executed within the `enumext` environment.

```

2273 \cs_new_protected:Nn __enumext_standard_reset_key:
2274 {
2275     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
2276     {
2277         \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n { \l__enumext_level_int }
2278             _int }
2279         {
2280             \int_gzero:c { c@__enumext_resume_ \int_to_roman:n { \l__enumext_level_int }
2281                 _int }
2282         }
2283     }
2284 }

```

The function `__enumext_standard_reset_key_star:` will be handled by the `reset*` key and will “reset” the counters `\c@__enumext_resume_X_int` to “zero” from the *level* at which it is executed within the `enumext` environment to the *lower levels*.

```

2285 \cs_new_protected:Nn __enumext_standard_reset_key_star:
2286 {
2287     \cs_set:Nn __enumext_tmp:n
2288     {
2289         \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n {##1} _int }
2290         {
2291             \int_gzero:c { c@__enumext_resume_ \int_to_roman:n {##1} _int }
2292         }
2293     }
2294     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
2295     {
2296         \int_step_function:nnN { \l__enumext_level_int } { 4 } __enumext_tmp:n
2297     }
2298 }

```

The function `__enumext_starred_reset_key:` will be handled by `reset` keys and `reset*` will “reset” the counter `\c@__enumext_resume_vii_int` to “zero” when executed in the `enumext*` environment.

```

2299 \cs_new_protected:Nn __enumext_starred_reset_key:
2300 {
2301     \int_gzero:c { c@__enumext_resume_vii_int }
2302 }

```

(End of definition for `__enumext_standard_reset_key:`, `__enumext_standard_reset_key_star:`, and `__enumext_starred_reset_key:`.)

13.30 Setting `save-ans`, `check-ans` and `no-store` keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

13.30.1 Definition of error messages

`save-ans-empty`

Message used by `save-ans` key.

```

2303 \msg_new:nnnn { enumext } { save-ans-empty }
2304 {
2305     Key~'save-ans'~needs~a~value~in~'#1'~\msg_line_context:.
2306 }
2307 {
2308     Give~a~name~to~the~storage,~for~example~'save-ans=name'.
2309 }

```

(End of definition for `save-ans-empty`.)

`missing-starred`
`many-starred`

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

2310 \msg_new:nnn { enumext } { missing-starred }

```

```

2311 {
2312     Missing~'\c_backslash_str #1'~#2.
2313 }
2314 \msg_new:nnn { enumext } { many-starred }
2315 {
2316     Many~'\c_backslash_str #1'~#2.
2317 }

```

(End of definition for *missing-starred* and *many-starred*.)

13.30.2 Definition of messages for the storage system

These messages are not error messages, they report to the `.log` file what the “*storage system*” is doing.

save-ans-log
save-ans-log-hook

Messages written to the `.log` file when the `save-ans` key starts and stops.

```

2318 \msg_new:nnn { enumext } { save-ans-log }
2319 {
2320     Start~'#1'~with~'save-ans=#2'~\msg_line_context:.
2321 }
2322 \msg_new:nnn { enumext } { save-ans-log-hook }
2323 {
2324     Stop~'#1'~with~'save-ans=#2'~\msg_line_context:.
2325 }

```

(End of definition for *save-ans-log* and *save-ans-log-hook*.)

store-prop
store-seq
store-int

Messages written to the `.log` file when the *prop list*, *sequence* and integer variables used by the `save-ans` key are created.

```

2326 \msg_new:nnn { enumext } { store-prop }
2327 {
2328     Creating~'\c_backslash_str g__enumext_ #1 _prop~\msg_line_context:.
2329 }
2330 \msg_new:nnn { enumext } { store-seq }
2331 {
2332     Creating~'\c_backslash_str g__enumext_ #1 _seq~\msg_line_context:.
2333 }
2334 \msg_new:nnn { enumext } { store-int }
2335 {
2336     Creating~'\c_backslash_str g__enumext_resume_ #1 _int~\msg_line_context:.
2337 }

```

(End of definition for *store-prop*, *store-seq*, and *store-int*.)

items-same-answer
item-greater-answer
item-less-answer

Messages used by the internal system to check answer used by `check-ans` key.

```

2338 \msg_new:nnn { enumext } { items-same-answer }
2339 {
2340     Answers~in~'#1'~(#2):~OK,~items~==~answers.\\
2341     Started~#3~and~closed~\msg_line_context:.
2342 }
2343 \msg_new:nnn { enumext } { item-greater-answer }
2344 {
2345     Answers~in~'#1'~(#2):~NOT~OK,~items~>~answers.\\
2346     Started~#3~and~closed~\msg_line_context:.
2347 }
2348 \msg_new:nnn { enumext } { item-less-answer }
2349 {
2350     Answers~in~'#1'~(#2):~NOT~OK,~items~<~answers.\\
2351     Started~#3~and~closed~\msg_line_context:.
2352 }

```

(End of definition for *items-same-answer*, *item-greater-answer*, and *item-less-answer*.)

13.30.3 Setting save-ans key

save-ans

Defines the key `save-ans` only for the “*first level*” of `enumext` and `enumext*`.

```

2353 \cs_set_protected:Nn \__enumext_tmp:n
2354 {
2355     \keys_define:nn { enumext / #1 }
2356     {
2357         save-ans .code:n = { \__enumext_storing_set:n {##1} },

```

```

2358         save-ans .value_required:n = true,
2359     }
2360 }
2361 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for save-ans.)

13.30.4 Internal functions for save-ans key

```

\__enumext_start_save_ans_msg:
\__enumext_stop_save_ans_msg:

```

The functions `\__enumext_start_save_ans_msg:` and `\__enumext_stop_save_ans_msg:` will write in the `.log` file the environment in which the `save-ans` key was executed along with the line at the beginning and end of it. The function `\__enumext_start_save_ans_msg:` will be passed to `\__enumext_storing_set:n` and the function `\__enumext_stop_save_ans_msg:` will be passed to the function `\__enumext_execute_after_env:`.

```

2362 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2363 {
2364     \msg_info:nnVV { enumext } { save-ans-log }
2365     \g__enumext_envir_name_tl \l__enumext_store_name_tl
2366 }
2367 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2368 {
2369     \msg_info:nnVV { enumext } { save-ans-log-hook }
2370     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2371 }

```

(End of definition for __enumext_start_save_ans_msg: and __enumext_stop_save_ans_msg:.)

```

\__enumext_storing_set:n
\__enumext_storing_exec:

```

The function `\__enumext_storing_set:n` first passes the value of the `save-ans` key to the variable `\l__enumext_store_name_tl` which contains the `{\<store name>}` of the *sequence* and *prop list* to be used. If `\l__enumext_store_name_tl` is *empty* it returns an error message, otherwise it returns the appropriate message `\__enumext_start_save_ans_msg:` and proceeds to execute the function `\__enumext_storing_exec:` for `enumext` and `enumext*` environments.

```

2372 \cs_new_protected:Nn \__enumext_storing_set:n
2373 {
2374     \tl_set:Ne \l__enumext_store_name_tl {#1}
2375     \tl_if_empty:NTF \l__enumext_store_name_tl
2376     {
2377         \bool_lazy_or:nnT
2378         { \l__enumext_standard_first_bool } { \l__enumext_starred_first_bool }
2379         {
2380             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2381         }
2382     }
2383     {
2384         \bool_lazy_or:nnT
2385         { \l__enumext_standard_first_bool } { \l__enumext_starred_first_bool }
2386         {
2387             \__enumext_start_save_ans_msg:
2388             \__enumext_storing_exec:
2389         }
2390     }
2391 }

```

The function `\__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, and copies the `{\<store name>}` into the variable `\g__enumext_store_name_tl`.

```

2392 \cs_new_protected:Nn \__enumext_storing_exec:
2393 {
2394     \bool_set_true:N \l__enumext_store_active_bool
2395     \bool_set_true:N \l__enumext_check_answers_bool
2396     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl

```

The *prop list* `\g__enumext_series_#1_prop` and the *sequence* `\g__enumext_series_#1_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_#1_int` used by the keys `resume` and `resume*`.

```

2397     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2398     {

```

```

2399     \msg_info:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2400     \prop_new:c { g__enumext_ \l__enumext_store_name_tl_prop }
2401   }
2402   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl_seq }
2403   {
2404     \msg_info:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2405     \seq_new:c { g__enumext_ \l__enumext_store_name_tl_seq }
2406   }
2407   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl_int }
2408   {
2409     \msg_info:nnV { enumext } { store-int } \l__enumext_store_name_tl
2410     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl_int }
2411   }
2412 }

```

(End of definition for `\__enumext_storing_set:n` and `\__enumext_storing_exec:.`)

13.30.5 The check answer mechanism

The internal mechanism for “checking answers” follows this logic:

If the line begins with `\item` or `\item*` and does NOT *open a nested environment*, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

If the line begins with `\item` or `\item*` and *opens a nested environment* each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must be decremented by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) it is necessary to:

1. Keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. Keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keep track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. The cases are analyzed below:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting the count is decremented by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test a “hook” function is used after closing the *first level* of the environment.

13.30.6 Setting check-ans and no-store keys

check-ans no-store Defines the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```

2413 \cs_set_protected:Nn \__enumext_tmp:n
2414 {
2415   \keys_define:nn { enumext / #1 }
2416   {
2417     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2418     check-ans .initial:n = false,
2419     check-ans .value_required:n = true,
2420     no-store .code:n =
2421     {
2422       \bool_set_false:N \l__enumext_check_answers_bool
2423       \bool_set_false:N \l__enumext_check_ans_key_bool
2424     },
2425     no-store .value_forbidden:n = true,
2426   }
2427 }
2428 \clist_map_inline:nn
2429 {

```

```

2430     level-1, level-2, level-3, level-4, enumext*
2431 }
2432 { \__enumext_tmp:n {#1} }

```

(End of definition for *check-ans* and *no-store*.)

13.30.7 Set-up check answer mechanism

`\__enumext_check_ans_active:` The function `\__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl` that is, the *save-ans* key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key *no-store* and will execute the function `\__enumext_check_ans_level:` only if true, i.e. the key *no-store* is not active.

```

2433 \cs_new_protected:Nn \__enumext_check_ans_active:
2434 {
2435     \tl_if_empty:NF \l__enumext_store_name_tl
2436     {
2437         \bool_if:NT \l__enumext_check_answers_bool
2438         {
2439             \__enumext_check_ans_level:
2440         }
2441     }
2442 }

```

The function `\__enumext_check_ans_level:` will decrement by “one” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment *enumext*, taking into account whether it is nested within *enumext** or the opposite and set `\l__enumext_item_number_bool` to *false*.

```

2443 \cs_new_protected:Nn \__enumext_check_ans_level:
2444 {
2445     \int_case:nn { \l__enumext_level_int }
2446     {
2447         { 1 }
2448         {
2449             \bool_lazy_all:nT
2450             {
2451                 { \bool_if_p:N \g__enumext_starred_bool }
2452                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2453             }
2454             {
2455                 \int_gdecr:N \g__enumext_item_number_int
2456                 \bool_set_false:N \l__enumext_item_number_bool
2457             }
2458         }
2459         { 2 }
2460         {
2461             \int_gdecr:N \g__enumext_item_number_int
2462             \bool_set_false:N \l__enumext_item_number_bool
2463         }
2464         { 3 }
2465         {
2466             \int_gdecr:N \g__enumext_item_number_int
2467             \bool_set_false:N \l__enumext_item_number_bool
2468         }
2469         { 4 }
2470         {
2471             \int_gdecr:N \g__enumext_item_number_int
2472             \bool_set_false:N \l__enumext_item_number_bool
2473         }
2474     }

```

This is executed only if *enumext** is nested in the “*first level*” of *enumext*, for the rest of the cases the value of `\g__enumext_item_number_int` is already decreased.

```

2475     \int_case:nn { \l__enumext_level_h_int }
2476     {
2477         { 1 }
2478         {
2479             \bool_lazy_all:nT
2480             {
2481                 { \bool_if_p:N \g__enumext_standard_bool }
2482                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2483             }

```

```

2484         {
2485             \int_gdecr:N \g__enumext_item_number_int
2486             \bool_set_false:N \l__enumext_item_number_bool
2487         }
2488     }
2489 }
2490 }

```

(End of definition for `\__enumext_check_ans_active:` and `\__enumext_check_ans_level:`.)

`\__enumext_check_ans_key_hook:` The function `\__enumext_check_ans_key_hook:` will *export* the status of the local variable `\l__enumext_check_ans_key_bool` to the global variable `\g__enumext_check_ans_key_bool` only if the key `check-ans` is active.

```

2491 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2492 {
2493     \bool_lazy_and:nnT
2494     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2495     { \bool_if_p:N \g__enumext_standard_bool }
2496     {
2497         \bool_gset_true:N \g__enumext_check_ans_key_bool
2498     }
2499     \bool_lazy_and:nnT
2500     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2501     { \bool_if_p:N \g__enumext_starred_bool }
2502     {
2503         \bool_gset_true:N \g__enumext_check_ans_key_bool
2504     }
2505 }

```

(End of definition for `\__enumext_check_ans_key_hook:`.)

`\__enumext_item_answer_diff:` The function `\__enumext_item_answer_diff:` will set the value of the variable `\g__enumext_item_answer_diff_int` which is used by the functions `\__enumext_check_ans_show:` for the key `save-ans` and by the function `\__enumext_check_ans_log:` by the internal “*check answer*” mechanism. This function will be passed to the function `\__enumext_execute_after_env:`.

```

2506 \cs_new_protected:Nn \__enumext_item_answer_diff:
2507 {
2508     \int_gset:Nn \g__enumext_item_answer_diff_int
2509     {
2510         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2511     }
2512 }

```

(End of definition for `\__enumext_item_answer_diff:`.)

`\__enumext_check_ans_show:` The function `\__enumext_check_ans_show:` will be executed within the function `\__enumext_execute_after_env:` when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is true and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff:`.

```

2513 \cs_new_protected:Nn \__enumext_check_ans_show:
2514 {
2515     \int_case:nn { \g__enumext_item_answer_diff_int }
2516     {
2517         { -1 } { \__enumext_check_ans_msg_less: }
2518         { 0 } { \__enumext_check_ans_msg_same_ok: }
2519         { 1 } { \__enumext_check_ans_msg_greater: }
2520     }
2521 }
2522 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2523 {
2524     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2525     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2526 }
2527 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2528 {
2529     \msg_info:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2530     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2531 }
2532 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:

```

```

2533 {
2534   \msg_warning:nneee { enumext } { item-greater-answer }
2535   { \g__enumext_store_name_tl }
2536   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2537 }

```

(End of definition for __enumext_check_ans_show: and others.)

```

\__enumext_check_ans_log:
\__enumext_check_ans_log_msg_less:
\__enumext_check_ans_log_msg_same_ok:
\__enumext_check_ans_log_msg_greater:

```

The function `\__enumext_check_ans_log:` will be executed within the function `\__enumext_execute_after_env` when the key `check-ans` is not active, that is, when `\g__enumext_check_ans_key_bool` is false and write in the log the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `\__enumext_item_answer_diff:`.

```

2538 \cs_new_protected:Nn \__enumext_check_ans_log:
2539 {
2540   \int_case:nn { \g__enumext_item_answer_diff_int }
2541   {
2542     { -1 } { \__enumext_check_ans_log_msg_less: }
2543     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2544     { 1 } { \__enumext_check_ans_log_msg_greater: }
2545   }
2546 }
2547 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2548 {
2549   \msg_info:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2550   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2551 }
2552 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2553 {
2554   \msg_info:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2555   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2556 }
2557 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2558 {
2559   \msg_info:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2560   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2561 }

```

(End of definition for __enumext_check_ans_log: and others.)

13.30.8 Check for \item* and \anspic* commands

```
\__enumext_check_starred_cmd:n
```

The function `\__enumext_check_starred_cmd:n` performs an *extra check* for the `keyans`, `keyans*` and `keyanspic` environments. Unlike the *check* executed by `check-ans` key this one is not controlled by any key, it is intended to prevent the forgetting of `\item*` or `\anspic*` in these environments.

```

2562 \cs_new_protected:Nn \__enumext_check_starred_cmd:n
2563 {
2564   \int_compare:nNnT
2565   { \g__enumext_check_starred_cmd_int } = { 0 }
2566   {
2567     \msg_warning:nnnV
2568     { enumext } { missing-starred } {#1} \l__enumext_check_start_line_env_tl
2569   }
2570   \int_compare:nNnT
2571   { \g__enumext_check_starred_cmd_int } > { 1 }
2572   {
2573     \msg_warning:nnnV
2574     { enumext } { many-starred } {#1} \l__enumext_check_start_line_env_tl
2575   }
2576   \int_gzero:N \g__enumext_check_starred_cmd_int
2577   \tl_clear:N \l__enumext_check_start_line_env_tl
2578 }

```

(End of definition for __enumext_check_starred_cmd:n.)

13.31 Keys and functions associated with storage

13.31.1 Definition of variables

```

\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool

```

Internal variables used by `save-key` key (§13.31.4) and the “*storing structure*”.

```
2579 \cs_set_protected:Nn \__enumext_tmp:n
```

```

2580 {
2581   \tl_new:c { \__enumext_store_save_key_ #1 _tl }
2582   \bool_new:c { \__enumext_store_save_key_ #1 _bool }
2583 }
2584 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\__enumext_store_save_key_X_tl` and `\__enumext_store_save_key_X_bool`.)

13.31.2 Keys for marks, wrap and show

The `enumext` package provides a set of *keys* for manipulating “symbol marks” associated with “answers” and how they are displayed and stored in the *sequence* and *prop list* as well as an internal “label and ref” system.

For the `keyans` and `keyans*` environments only the keys `mark-ans*`, `mark-pos*`, `mark-sep*`, `wrap-ans*`, `wrap-opt`, `save-sep`, `show-ans` and `show-pos` are added.

```

2585 \cs_set_protected:Nn \__enumext_tmp:nn
2586 {
2587   \keys_define:nn { enumext / #1 }
2588   {
2589     mark-ans* .tl_set:c = { \__enumext_mark_answer_sym_ #2 _tl },
2590     mark-ans* .initial:n = \textasteriskcentered,
2591     mark-ans* .value_required:n = true,
2592     mark-pos* .choices:nn =
2593       { left , right , center }
2594       {
2595         \int_case:nn { \l_keys_choice_int }
2596         {
2597           {1} { \str_set:cn { \__enumext_mark_position_ #2 _str } { l } }
2598           {2} { \str_set:cn { \__enumext_mark_position_ #2 _str } { r } }
2599           {3} { \str_set:cn { \__enumext_mark_position_ #2 _str } { c } }
2600         }
2601       },
2602     mark-pos* .initial:n = right,
2603     mark-pos* .value_required:n = true,
2604     mark-pos* / unknown .code:n =
2605       \msg_error:nneee { enumext } { unknown-choice }
2606       { mark-pos* } { left,~right,~center } { \exp_not:n {##1} },
2607     mark-sep* .dim_set:c = { \__enumext_mark_sym_sep_ #2 _dim },
2608     mark-sep* .value_required:n = true,
2609     wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_ #2 :n } ##1,
2610     wrap-ans* .value_required:n = true,
2611     wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_ #2 :n } ##1,
2612     wrap-opt .initial:n = [{##1}],
2613     wrap-opt .value_required:n = true,
2614     save-sep .tl_set:c = { \__enumext_store_keyans_item_opt_sep_ #2 _tl },
2615     save-sep .initial:n = {,~},
2616     save-sep .value_required:n = true,
2617     show-ans .bool_set:N = \__enumext_show_answer_bool,
2618     show-ans .initial:n = false,
2619     show-ans .value_required:n = true,
2620     show-pos .bool_set:N = \__enumext_show_position_bool,
2621     show-pos .initial:n = false,
2622     show-pos .value_required:n = true,
2623   }
2624 }
2625 \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `mark-ans*` and others.)

Adds the *keys* `mark-ref` and `save-ref` related to the “storage system” and internal mechanism of “label and ref” along with the *keys* `show-ans`, `show-pos` and the *keys* `mark-ans`, `mark-pos`, `mark-sep` and `wrap-ans` for the command `\anskey`, the environment `anskey*` and the *keys* for environments `keyans` and `keyans*` only at the *first level* of `enumext` and `enumext*`.

```

2626 \cs_set_protected:Nn \__enumext_tmp:n
2627 {
2628   \keys_define:nn { enumext / #1 }
2629   {
2630     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2631     mark-pos* .initial:n = \textreferencemark,
2632     mark-sep* .value_required:n = true,
2633   }

```

```

2633     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2634     save-ref .initial:n = false,
2635     save-ref .value_required:n = true,
2636     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2637     show-ans .initial:n = false,
2638     show-ans .value_required:n = true,
2639     show-pos .bool_set:N = \l__enumext_show_position_bool,
2640     show-pos .initial:n = false,
2641     show-pos .value_required:n = true,
2642     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2643     mark-ans .initial:n = \textasteriskcentered,
2644     mark-ans .value_required:n = true,
2645     mark-sep .dim_set:N = \l__enumext_mark_sym_sep_dim,
2646     mark-sep .value_required:n = true,
2647     mark-pos .choices:nn =
2648       { left , right , center }
2649     {
2650       \int_case:nn { \l_keys_choice_int }
2651       {
2652         {1} { \str_set:Nn \l__enumext_mark_position_str { l } }
2653         {2} { \str_set:Nn \l__enumext_mark_position_str { r } }
2654         {3} { \str_set:Nn \l__enumext_mark_position_str { c } }
2655       }
2656     },
2657     mark-pos .initial:n = right,
2658     mark-pos .value_required:n = true,
2659     mark-pos / unknown .code:n =
2660       \msg_error:nneee { enumext } { unknown-choice }
2661       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2662
2663     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2664     wrap-ans .initial:n =
2665       {
2666         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep
2667           -2\fboxrule}}{##1}}
2668       },
2669     wrap-ans .value_required:n = true,
2670     mark-ans* .code:n =
2671       {
2672         \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2673         \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2674       },
2675     mark-ans* .value_required:n = true,
2676     mark-pos* .code:n =
2677       {
2678         \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2679         \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2680       },
2681     mark-pos* .value_required:n = true,
2682     mark-sep* .code:n =
2683       {
2684         \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2685         \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2686       },
2687     mark-sep* .value_required:n = true,
2688     wrap-ans* .code:n =
2689       {
2690         \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2691         \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2692       },
2693     wrap-ans* .value_required:n = true,
2694     wrap-opt .code:n =
2695       {
2696         \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2697         \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2698       },
2699     wrap-opt .value_required:n = true,
2700     save-sep .code:n =
2701       {
2702         \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2703         \keys_set:nn { enumext / keyans* } { save-sep = {##1} }

```

```

2704         },
2705         save-sep .value_required:n = true,
2706     }
2707 }
2708 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for mark-ref and others.)

13.31.3 Storing structure of the environments

The idea behind “*storing structure*” in the *sequence* is to have a copy of the *structure of the environment* in which the key `save-ans` is being executed, so the *optional argument* passed to the levels of the environment in which it is executed must be captured and “*stored*” in the *sequence*.

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for the “*storing keys*” filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. This function is executed only if the variable `\__enumext_store_save_key_X_bool` is false, that is, the key `save-key` is not active, establishing the variable `\__enumext_store_save_key_X_tl` with the filtered *keys*.

```

2709 \cs_new_protected:Nn \__enumext_store_active_keys:n
2710 {
2711     \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2712     {
2713         \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }
2714         \tl_set:ce
2715             { \__enumext_store_save_key_ \__enumext_level: _tl }
2716             { \__enumext_filter_save_key:n {#1} }
2717     }
2718 }
2719 \cs_new_protected:Nn \__enumext_store_active_keys_vii:n
2720 {
2721     \bool_if:NF \__enumext_store_save_key_vii_bool
2722     {
2723         \tl_clear:N \__enumext_store_save_key_vii_tl
2724         \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2725     }
2726 }

```

(End of definition for `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`.)

13.31.4 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the `save-ans` key when executing `\anskey` or `anskey*`, it cannot be modified. The best option here is to have a key that allows modifying the *optional argument* of the “*storing structure*” in the *sequence*.

`save-key` The values set by this key passed in the *optional argument* of the `enumext` and `enumext*` environments will override the values of the `\__enumext_store_save_key_X_tl` variable set by the functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`. Now define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2727 \cs_set_protected:Nn \__enumext_tmp:n
2728 {
2729     \keys_define:nn { enumext / enumext* }
2730     {
2731         save-key .code:n = { \__enumext_parse_save_key_vii:n {##1} },
2732         save-key .value_required:n = true,
2733     }
2734     \keys_define:nn { enumext / #1 }
2735     {
2736         save-key .code:n = { \__enumext_parse_save_key:n {##1} },
2737         save-key .value_required:n = true,
2738     }
2739 }
2740 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

The functions `\__enumext_parse_save_key:n` and `\__enumext_parse_save_key_vii:n` will be responsible for “*storing keys*” in the variable `\__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2741 \cs_new_protected:Nn \__enumext_parse_save_key:n

```

```

2742 {
2743   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2744   \tl_set:ce
2745     { l__enumext_store_save_key_ \__enumext_level: _tl }
2746     { \__enumext_filter_save_key:n {#1} }
2747 }
2748 \cs_new_protected:Nn \__enumext_parse_save_key_vii:n
2749 {
2750   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2751   \tl_clear:N \l__enumext_store_save_key_vii_tl
2752   \tl_set:Nc \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2753 }

```

(End of definition for __enumext_parse_save_key:n and __enumext_parse_save_key_vii:n.)

13.31.5 Internal functions to store optional arguments

The function __enumext_filter_save_key:n will be in charge of “filtering keys” to be stored in sequence where {#1} represents the *optional argument* passed to the environment.

```

\__enumext_filter_save_key:n
  \__enumext_filter_save_key_key:n
  \__enumext_filter_save_key_pair:nn
2754 \cs_new:Nn \__enumext_filter_save_key:n
2755 {
2756   \use:e
2757   {
2758     \keyval_parse:NNn
2759       \__enumext_filter_save_key_key:n
2760       \__enumext_filter_save_key_pair:nn {#1}
2761   }
2762 }

```

The function __enumext_filter_save_key_key:n will be responsible for “filtering keys” that are passed “without value” by excluding the `resume`, `resume*`, `reset`, `reset*`, `no-store` and `base-fix` keys.

```

2763 \cs_new:Nn \__enumext_filter_save_key_key:n
2764 {
2765   \str_case:nnF {#1}
2766   {
2767     { resume } { } { resume* } { } { reset } { } { reset* } { }
2768     { no-store } { } { base-fix } { }
2769   }
2770   { , { \exp_not:n {#1} } }
2771 }

```

The function __enumext_filter_save_key_pair:nn will be responsible for “filtering keys” that are passed “with value” by excluding the `series`, `resume`, `save-ans`, `save-ref`, `save-key`, `check-ans`, `show-ans`, `show-pos`, `mark-ans`, `mark-pos`, `mark-sep`, `wrap-ans`, `mark-ans*`, `mark-pos*`, `mark-sep*`, `wrap-ans*`, `wrap-opt`, `save-sep`, `mark-ref`, `mini-env`, `mini-sep`, `mini-right` and `mini-right*` keys.

```

2772 \cs_new:Nn \__enumext_filter_save_key_pair:nn
2773 {
2774   \str_case:nnF {#1}
2775   {
2776     { series } { } { resume } { } { save-ans } { }
2777     { save-ref } { } { save-key } { } { check-ans } { }
2778     { show-ans } { } { show-pos } { } { mark-ans } { }
2779     { mark-pos } { } { mark-sep } { } { wrap-ans } { }
2780     { mark-ans* } { } { mark-pos* } { } { mark-sep* } { }
2781     { wrap-ans* } { } { wrap-opt } { } { save-sep } { }
2782     { mark-ref } { } { mini-env } { } { mini-sep } { }
2783     { mini-right } { } { mini-right* } { }
2784   }
2785   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2786 }

```

(End of definition for __enumext_filter_save_key:n, __enumext_filter_save_key_key:n, and __enumext_filter_save_key_pair:nn.)

13.31.6 Function for storing content in prop list

__enumext_store_addto_prop:n The function __enumext_store_addto_prop:n stores the {<content>} in *prop list* defined by `save-ans` key. The “stored content” is retrieved by means of the `\getkeyans` command.

The form in which the {<content>} is “stored” in the *prop list* is {<position>}{<content>}. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and

`\anspic*` in `keyanspic` environment. If there is no *store name*, because it was used outside the `save-ans` key and the error has already been reported, it does nothing.

```

2787 \cs_new_protected:Nn \__enumext_store_addto_prop:n
2788 {
2789     \tl_if_empty:NF \l__enumext_store_name_tl
2790     {
2791         \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2792         {
2793             \int_eval:n
2794             { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2795         }
2796         {#1}
2797     }
2798 }
2799 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

(End of definition for `\__enumext_store_addto_prop:n`)

13.31.7 Function for storing content in sequence

`\__enumext_store_addto_seq:n` The function `\__enumext_store_addto_seq:n` stores the $\{\langle content \rangle\}$ in *sequence* defined by `save-ans` key. This function is used by `\anskey` and `anskey*` in `enumext` and `enumext*`, `\item*` in `keyans` and `keyans*` and `\anspic*` in `keyanspic`.

The form in which the $\{\langle content \rangle\}$ is stored in *sequence* is in an internal `enumext` or `enumext*` environments with the “*same structure*” in which the command was executed.

The “*stored content*” is retrieved by means of the `\printkeyans` command. As in the previous function, it does nothing when there is no *store name*.

```

2800 \cs_new_protected:Nn \__enumext_store_addto_seq:n
2801 {
2802     \tl_if_empty:NF \l__enumext_store_name_tl
2803     {
2804         \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } {#1}
2805     }
2806 }
2807 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

(End of definition for `\__enumext_store_addto_seq:n`)

13.31.8 Functions for storing structure in the sequence

`\__enumext_store_level_open:` The “*storing structure*” is handled by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```

2808 \cs_new_protected:Nn \__enumext_store_level_open:
2809 {
2810     \bool_if:NT \l__enumext_check_answers_bool
2811     {
2812         \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2813         {
2814             \__enumext_store_addto_seq:n
2815             {
2816                 \item \begin{enumext}
2817             }
2818         }
2819     }
2820     \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2821     {
2822         \item \begin{enumext} [
2823     }
2824     \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2825     {
2826     ]
2827     }
2828     \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2829 }
2830 }
2831 }
2832 \cs_new_protected:Nn \__enumext_store_level_close:
2833 {
2834     \bool_if:NT \l__enumext_check_answers_bool
2835     {

```

```

2836         \__enumext_store_addto_seq:n { \end{enumext} }
2837     }
2838 }

```

(End of definition for __enumext_store_level_open: and __enumext_store_level_close:.)

```

\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:

```

The “*storing structure*” is handled by the functions `\__enumext_store_level_open_vii:` and `\__enumext_store_level_close_vii:` which are executed in the `enumext*` environment.

```

2839 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2840 {
2841     \bool_if:NT \l__enumext_check_answers_bool
2842     {
2843         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2844         {
2845             \__enumext_store_addto_seq:n
2846             {
2847                 \item \begin{enumext*}
2848             }
2849         }
2850         {
2851             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2852             {
2853                 \item \begin{enumext*}[
2854             }
2855             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2856             {
2857                 ]
2858             }
2859             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2860         }
2861     }
2862 }
2863 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2864 {
2865     \bool_if:NT \l__enumext_check_answers_bool
2866     {
2867         \__enumext_store_addto_seq:n { \end{enumext*} }
2868     }
2869 }

```

(End of definition for __enumext_store_level_open_vii: and __enumext_store_level_close_vii:.)

13.31.9 Function for show marks and position

```

\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc

```

The function `\__enumext_print_keyans_box:NN` prints a box in the left margin with `\l__enumext_mark_answer_` used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

#1: `\l__enumext_labelwidth_X_dim`
 #2: `\l__enumext_labelsep_X_dim`

```

2870 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2871 {
2872     \mode_leave_vertical:
2873     \skip_horizontal:n { -\dim_use:N #2 }
2874     \hbox_overlap_left:n
2875     {
2876         \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2877         {
2878             \tl_use:N \l__enumext_mark_answer_sym_tl
2879         }
2880     }
2881     \skip_horizontal:n { \dim_use:N #2 }
2882 }
2883 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

13.32 The internal label and ref

The function `\__enumext_store_internal_ref:` handles the “*internal label and ref*” system used by the `save-ref` and `mark-ref` keys for `\anskey` allow executing `\ref{⟨store name⟩ : position}` and will return `1.(a).i.A.`

_enumext_store_internal_ref:

First the dots “.” are removed from the current $\langle labels \rangle$, since double dots in references are not wanted, then the result is placed in the variable _enumext_newlabel_arg_two_tl.

```

2884 \cs_new_protected:Nn \\_enumext_store_internal_ref:
2885 {
2886   \cs_set_protected:Nn \\_enumext_tmp:n
2887   {
2888     \tl_set_eq:cc { l__enumext_label_copy_ ##1 _tl } { l__enumext_label_ ##1 _tl }
2889     \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
2890     \tl_remove_once:cn { l__enumext_label_copy_ ##1 _tl } { . }
2891     \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
2892   }
2893   \clist_map_inline:nn { i, ii, iii, iv, vii } { \\_enumext_tmp:n {##1} }
2894   \cs_set:Nn \\_enumext_tmp:n
2895   { . \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

Here it is necessary to analyze the cases where the environment is started with `enumext*` and if `\anskey` or `anskey*` is running alone in it or if it is running in a nested `enumext` environment within the starting environment.

```

2896   \bool_lazy_all:nT
2897   {
2898     { \bool_if_p:N \g__enumext_starred_bool }
2899     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2900   }
2901   {
2902     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2903     { \tl_use:N \\_enumext_label_copy_vii_tl }
2904   }
2905   \bool_lazy_all:nT
2906   {
2907     { \bool_not_p:n { \g__enumext_standard_bool } }
2908     { \bool_if_p:N \\_enumext_standard_bool }
2909     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2910   }
2911   {
2912     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2913     {
2914       \tl_use:N \\_enumext_label_copy_vii_tl
2915       \int_step_function:nnN { 1 } { \l__enumext_level_int } \\_enumext_tmp:n
2916     }
2917   }

```

If started with `enumext` and if `\anskey` or `anskey*` is running alone in it or if it is running in a nested `enumext*` environment within the starting environment.

```

2918   \bool_lazy_all:nT
2919   {
2920     { \bool_if_p:N \g__enumext_standard_bool }
2921     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2922     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2923   }
2924   {
2925     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2926     {
2927       \tl_use:N \\_enumext_label_copy_i_tl
2928       \int_step_function:nnN { 2 } { \l__enumext_level_int } \\_enumext_tmp:n
2929     }
2930   }
2931   \cs_set:Nn \\_enumext_tmp:n
2932   { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2933   \bool_lazy_all:nT
2934   {
2935     { \bool_if_p:N \g__enumext_standard_bool }
2936     { \bool_if_p:N \\_enumext_starred_bool }
2937     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2938   }
2939   {
2940     \tl_put_right:Ne \\_enumext_newlabel_arg_two_tl
2941     {
2942       \int_step_function:nnN { 1 } { \l__enumext_level_int } \\_enumext_tmp:n
2943       \tl_use:N \\_enumext_label_copy_vii_tl
2944     }
2945   }

```

Now the variable `\l__enumext_newlabel_arg_one_tl` is set, which contains $\langle \textit{store name} : \textit{position} \rangle$.

```

2946 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2947 {
2948   \l__enumext_store_name_tl \c_colon_str
2949   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2950 }

```

Now execute the function `\__enumext_newlabel:nn` and save the result in the variable `\l__enumext_write_aux_file_tl` and finally write it in the `.aux` file.

```

2951 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2952 {
2953   \__enumext_newlabel:nn
2954   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2955   { \l__enumext_newlabel_arg_two_tl }
2956 }
2957 \l__enumext_write_aux_file_tl
2958 }

```

(End of definition for `\__enumext_store_internal_ref:.`)

13.33 Common functions for `\anskey` and `anskey*` environment

13.33.1 Definition of variables

`\l__enumext_store_anskey_arg_tl` The variable `\l__enumext_store_anskey_arg_tl` saves the *argument* of `\anskey` (§13.34).

```

2959 \tl_new:N \l__enumext_store_anskey_arg_tl

```

(End of definition for `\l__enumext_store_anskey_arg_tl`.)

13.33.2 Definition of error messages

`anskey-unnumber-item` Messages used by `\anskey` and `anskey*`.

`anskey-empty-arg`
`anskey-math-mode`

```

2960 \msg_new:nnnn { enumext } { anskey-unnumber-item }
2961 {
2962   Can't store with an unnumbered '\token_to_str:N \item' \msg_line_context:.
2963 }
2964 {
2965   Store the content with a numbered '\token_to_str:N \item'.
2966 }
2967 \msg_new:nnnn { enumext } { anskey-empty-arg }
2968 {
2969   Can't store empty content \msg_line_context:.
2970 }
2971 {
2972   Give some content to '\token_to_str:N \anskey' or remove the command.
2973 }
2974 \msg_new:nnnn { enumext } { anskey-math-mode }
2975 {
2976   '#1' can't work in math mode \msg_line_context:.
2977 }
2978 {
2979   Leave math mode before using '#1'.
2980 }

```

(End of definition for `anskey-unnumber-item`, `anskey-empty-arg`, and `anskey-math-mode`.)

13.33.3 Internal functions common to `\anskey` and `anskey*`

`\__enumext_store_anskey_arg:n` The internal function `\__enumext_store_anskey_arg:n` first passes the $\langle \textit{argument} \rangle$ to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the `save-ref` key and calls the function `\__enumext_store_internal_ref:` for the “*internal label and ref*” system. After this, if the `show-ans` or `show-pos` keys are active, it shows the “*wrapped*” $\langle \textit{argument} \rangle$.

```

2981 \cs_new_protected:Nn \__enumext_store_anskey_arg:n
2982 {
2983   \int_gincr:N \g__enumext_item_anskey_int
2984   \__enumext_store_addto_prop:n {#1}
2985   \bool_if:NT \l__enumext_store_ref_key_bool
2986   {
2987     \__enumext_store_internal_ref:
2988   }
2989   \__enumext_anskey_show_wrap_left:n {#1}

```

Now the processing of the `[(key = val)]` passed to the command starts, building the `\item` in the variable `\l__enumext_store_anskey_arg_tl` which is “stored” in the *sequence*. First the variable `\l__enumext_store_anskey_arg_tl` is cleared and the `(keys)` are processed; if the `break-col` key is present and the command is running under `enumext` (not in `enumext*`), `\columnbreak` and then `\item` are added.

```

2990 \tl_clear:N \l__enumext_store_anskey_arg_tl
2991 \bool_lazy_and:nnT
2992 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2993 { \bool_not_p:n { \l__enumext_starred_bool } }
2994 {
2995   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2996 }
2997 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the `item-join` key is present and the command is not running under `enumext*`, `(\number)` is added to `\l__enumext_store_anskey_arg_tl`.

```

2998 \bool_lazy_and:nnT
2999 { \bool_not_p:n { \l__enumext_starred_bool } }
3000 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
3001 {
3002   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3003   {
3004     ( \exp_not:V \l__enumext_store_item_join_int )
3005   }
3006 }

```

Then the keys `item-star`, `item-sym*` and `item-pos*` are reviewed and passed to `\l__enumext_store_anskey_arg` along with the `{(argument)}` for `\anskey` or `(body)` for `anskey*`.

```

3007 \bool_if:NTF \l__enumext_store_item_star_bool
3008 {
3009   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
3010   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
3011   {
3012     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3013     {
3014       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
3015     }
3016   }
3017   \dim_compare:nT
3018   {
3019     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
3020   }
3021   {
3022     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3023     {
3024       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
3025     }
3026   }
3027   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
3028 }
3029 {
3030   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
3031 }

```

Finally it checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with “symbol” set by `mark-ref` key and then store in *sequence*.

```

3032 \bool_lazy_and:nnT
3033 { \bool_if_p:N \l__enumext_store_ref_key_bool }
3034 { \bool_if_p:N \l__enumext_hyperref_bool }
3035 {
3036   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3037   {
3038     \hfill \exp_not:N \hyperlink
3039     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3040     { \exp_not:V \l__enumext_mark_ref_sym_tl }
3041   }
3042 }
3043 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
3044 }

```

(End of definition for `\l__enumext_store_anskey_arg:n`)

`\__enumext_anskey_show_wrap_arg:n`

The function `\__enumext_anskey_show_wrap_arg:n` “wraps” the $\langle argument \rangle$ passed to `\anskey` and the $\langle body \rangle$ for `anskey*` when using the `wrap-ans` and `mark-sep` keys.

```

3045 \cs_new_protected:Nn \__enumext_anskey_show_wrap_arg:n
3046 {
3047   \par
3048   \bool_if:NTF \l__enumext_starred_bool
3049   {
3050     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3051     {
3052       \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
3053     }
3054     \__enumext_print_keyans_box:NN
3055     \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
3056   }
3057   {
3058     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3059     {
3060       \dim_set:Nn \l__enumext_mark_sym_sep_dim
3061       {
3062         \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
3063       }
3064     }
3065     \__enumext_print_keyans_box:cc
3066     { \l__enumext_labelwidth_ \__enumext_level: _dim } { \l__enumext_mark_sym_sep_dim }
3067   }
3068   \__enumext_anskey_wrapper:n {#1}
3069 }

```

(End of definition for `\__enumext_anskey_show_wrap_arg:n`.)

`\__enumext_anskey_show_wrap_left:n`

The function `\__enumext_anskey_show_wrap_left:n` will show the “mark” defined by the `mark-ans` key or the “position” of the $\langle content \rangle$ stored in the *prop list* when using the `show-pos` key on the left margin next to the “wraps” $\langle argument \rangle$ passed to `\anskey` and the $\langle body \rangle$ in `anskey*` on the right side when using the `show-ans` key.

```

3070 \cs_new_protected:Nn \__enumext_anskey_show_wrap_left:n
3071 {
3072   \bool_if:NT \l__enumext_show_answer_bool
3073   {
3074     \__enumext_anskey_show_wrap_arg:n {#1}
3075   }
3076   \bool_if:NT \l__enumext_show_position_bool
3077   {
3078     \tl_set:Nc \l__enumext_mark_answer_sym_tl
3079     {
3080       \group_begin:
3081       \exp_not:N \normalfont
3082       \exp_not:N \footnotesize [ \int_eval:n
3083       {
3084         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3085       }
3086       ]
3087       \group_end:
3088     }
3089     \__enumext_anskey_show_wrap_arg:n {#1}
3090   }
3091 }

```

(End of definition for `\__enumext_anskey_show_wrap_left:n`.)

13.34 The command `\anskey`

Since “content” is stored in a *list* environment within *sequences* and the options passed to each level can be managed (more or less), it is necessary to have a little more control over `\item` when storing.

The `\anskey` command will cover this point and give it similar behavior to that of `\item` in the `enumext` and `enumext*` environments executed as follows `\anskey[⟨key = val⟩]{⟨content⟩}`.

13.34.1 Definition of variables for `\anskey`

`\l__enumext_anskey_level_int`

The integer variable `\l__enumext_anskey_level_int` will control the nesting level of the `\anskey` command.

```

3092 \int_new:N \l__enumext_anskey_level_int

```

(End of definition for `\__enumext_anskey_level_int`.)

13.34.2 Definition of error messages for `\anskey`

Messages used by `\anskey` command.

```

3093 \msg_new:nnnn { enumext } { anskey-wrong-place }
3094 {
3095   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.
3096 }
3097 {
3098   '\c_backslash_str #1'~works~in~the~environment~'#2'.
3099 }
3100 \msg_new:nnnn { enumext } { anskey-nested }
3101 {
3102   The~command~'\token_to_str:N \anskey'~can't~be~nested~
3103   \msg_line_context:.
3104 }
3105 {
3106   Close~the~first~'\token_to_str:N \anskey'~before~using~another~one.
3107 }
3108 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
3109 {
3110   The~key~'#1'~is~unknown~by~'\token_to_str:N \anskey'~and~is~being~
3111   ignored~\msg_line_context:.
3112 }
3113 {
3114   The~command~'\token_to_str:N \anskey'~does~not~have~a~key~called~'#1'.~
3115   Check~that~you~have~spelled~the~key~name~correctly.
3116 }
3117 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
3118 {
3119   The~key~'#1=#2'~is~unknown~by~'\token_to_str:N \anskey'~and~is~being~
3120   ignored~\msg_line_context:.
3121 }
3122 {
3123   The~command~'\token_to_str:N \anskey'~does~not~have~a~key~called~'#1'.~
3124   Check~that~you~have~spelled~the~key~name~correctly.
3125 }

```

(End of definition for `anskey-wrong-place` and others.)

13.34.3 Definition of keys for `\anskey`

First, the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*` are added.

```

3126 \keys_define:nn { enumext / anskey }
3127 {
3128   break-col .bool_set:N = \__enumext_store_columns_break_bool,
3129   break-col .default:n = true,
3130   break-col .value_forbidden:n = true,
3131   item-join .int_set:N = \__enumext_store_item_join_int,
3132   item-join .value_required:n = true,
3133   item-star .bool_set:N = \__enumext_store_item_star_bool,
3134   item-star .default:n = true,
3135   item-star .value_forbidden:n = true,
3136   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
3137   item-sym* .value_required:n = true,
3138   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
3139   item-pos* .value_required:n = true,
3140   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
3141 }

```

(End of definition for `break-col` and others.)

13.34.4 Internal functions for the command

The `<keys>` are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\__enumext_anskey_unknown:n`.

```

3142 \cs_new_protected:Nn \__enumext_anskey_unknown:n
3143 {
3144   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
3145 }

```

```

3146 \cs_new_protected:Nn \__enumext_anskey_unknown:nn
3147 {
3148   \tl_if_blank:nTF {#2}
3149   {
3150     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
3151   }
3152   {
3153     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
3154   }
3155 }

```

(End of definition for `\__enumext_anskey_unknown:n` and `\__enumext_anskey_unknown:nn`.)

`\__enumext_in_keyans:` Inside `keyans`, `keyans*` and `keyanspic` the flag `\l__enumext_store_active_bool` is set to false on purpose, so the `save-ans` checks cannot use it there. The predicate `\__enumext_in_keyans:` is true when any of these environments is open, and it avoids a second error message for the same mistake.

```

3156 \prg_new_conditional:Npnn \__enumext_in_keyans: { p , T , F , TF }
3157 {
3158   \int_compare:nNnTF
3159   { \l__enumext_keyans_level_int + \l__enumext_keyans_level_h_int
3160     + \l__enumext_keyans_pic_level_int } > { 0 }
3161   { \prg_return_true: } { \prg_return_false: }
3162 }

```

(End of definition for `\__enumext_in_keyans:`.)

`\__enumext_anskey_safe_outer:` The `\__enumext_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the `save-ans` key was activated or inside the `keyans`, `keyans*` or `keyanspic` environments.

`\__enumext_anskey_safe_inner:`

```

3163 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
3164 {
3165   \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
3166   {
3167     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
3168   }
3169   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3170   {
3171     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
3172   }
3173   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3174   {
3175     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
3176   }
3177   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3178   {
3179     \msg_error:nnnn { enumext }
3180     { command-wrong-place } { anskey } { keyanspic }
3181   }
3182 }

```

The `\__enumext_anskey_safe_inner:` function will first check if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

3183 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3184 {
3185   \int_incr:N \l__enumext_anskey_level_int
3186   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
3187   {
3188     \msg_error:nn { enumext } { anskey-nested }
3189   }
3190   \bool_if:NF \l__enumext_item_number_bool
3191   {
3192     \msg_error:nn { enumext } { anskey-unnumber-item }
3193   }
3194   \mode_if_math:T
3195   {
3196     \msg_error:nne { enumext } { anskey-math-mode }
3197     { \token_to_str:N \anskey }
3198   }
3199 }

```

(End of definition for `\__enumext_anskey_safe_outer:` and `\__enumext_anskey_safe_inner:`.)

- The `\anskey` command will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error.

`\anskey` The command accepts an optional [`<key = val>`] argument in ‘#1’ and a mandatory {`<content>`} in ‘#2’. The function `\__enumext_anskey_safe_outer:` is called first to be sure where the command is executed, then the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store` is checked; if it is true, `\g__enumext_item_anskey_int` is incremented for the internal “*check answer*” system and the function `\__enumext_anskey_safe_inner:n` is executed to ensure that the command is not nested and that the argument is not empty; finally the [`<key = val>`] is searched and the function `\__enumext_store_anskey_arg:n` is called.

```

3200 \NewDocumentCommand \anskey { o +m }
3201 {
3202   \__enumext_anskey_safe_outer:
3203   \group_begin:
3204     \bool_if:NT \l__enumext_check_answers_bool
3205     {
3206       \tl_if_novalue:nF {#1}
3207       {
3208         \keys_set:nn { enumext / anskey } {#1}
3209       }
3210       \tl_if_blank:nTF {#2}
3211       {
3212         \msg_error:nn { enumext } { anskey-empty-arg }
3213       }
3214       {
3215         \__enumext_anskey_safe_inner:
3216         \__enumext_store_anskey_arg:n {#2}
3217       }
3218     }
3219   \group_end:
3220 }

```

(End of definition for `\anskey`. This function is documented on page 16.)

13.35 The environment `anskey*`

The original implementation of the `anskey*` environment used non-public functions from the `scontents`[17] package, which was not the best approach. Fortunately L^AT_EX release 2025-06-01 implemented the new `c`-type argument in the `ltccmd`[11], with which the `<body>` of the environment can be recorded in *verbatim mode*, and `\tl_retokenize:n` (wrapper around the `\scantokens`) provided by L^AT_EX₃ release 2025-07-08 does the work as the original implementation.

13.35.1 Definition of variables for `anskey*`

`\l__enumext_store_anskey_env_tl` The variable `\l__enumext_store_anskey_env_tl` saves the `<body>` of the environment `anskey*` (§13.35).
`\l__enumext_write_anskey_env_bool` The variables `\l__enumext_write_anskey_env_bool`, `\l__enumext_write_anskey_env_file_name_tl`
`\l__enumext_write_anskey_env_file_name_tl` and `\l__enumext_write_anskey_env_file_iow` are used by the `write-env` and `overwrite` keys in the
`\l__enumext_write_anskey_env_file_iow` `anskey*` environment implementation.

```

3221 \tl_new:N \l__enumext_store_anskey_env_tl
3222 \bool_new:N \l__enumext_write_anskey_env_bool
3223 \tl_new:N \l__enumext_write_anskey_env_file_name_tl
3224 \iow_new:N \l__enumext_write_anskey_env_file_iow

```

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

`\c__enumext_anskey_env_hidden_space_str` The `\c__enumext_anskey_env_hidden_space_str` is a constant *string* used to hide the `<forced space>` added by T_EX when recording content in a macro. This *string* contains the *reserved phrase* ‘`%^^Aenumextheol%`’ which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```

3225 \str_const:Ne \c__enumext_anskey_env_hidden_space_str
3226 { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol
3227   \c_percent_str }

```

(End of definition for `\c__enumext_anskey_env_hidden_space_str`.)

13.35.2 Definition of error messages for anskey*

anskey-env-error Message used by reserved **anskey*** environment by **enumext** package.

```

3228 \msg_new:nnnn { enumext } { anskey-env-error }
3229 {
3230   The~environment~'#1'~is~reserved~by~'enumext'~package,~
3231   it~is~already~defined~\msg_line_context:.
3232 }
3233 {
3234   The~environment~'#1'~is~defined~internally~
3235   for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.
3236 }

```

(End of definition for anskey-env-error.)

anskey-env-wrong Messages used by **anskey*** environment.

anskey-env-key-unknown

anskey-env-key-value-unknown

```

3237 \msg_new:nnnn { enumext } { anskey-env-wrong }
3238 {
3239   The~environment~anskey*~cannot~be~used~in~'#1'~\msg_line_context:.
3240 }
3241 {
3242   'anskey*'~can~only~be~used~with~the~key~'save-ans'~in~
3243   'enumext'~or~'enumext*'.
3244 }
3245 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
3246 {
3247   The~key~'#1'~is~unknown~by~environment~
3248   'anskey*'~and~is~being~ignored~\msg_line_context:.
3249 }
3250 {
3251   The~environment~'anskey*'~does~not~have~a~key~called~'#1'.~
3252   Check~that~you~have~spelled~the~key~name~correctly.
3253 }
3254 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
3255 {
3256   The~key~'#1=#2'~is~unknown~by~environment~
3257   'anskey*'~and~is~being~ignored~\msg_line_context:.
3258 }
3259 {
3260   The~environment~'anskey*'~does~not~have~a~key~called~'#1'.~
3261   Check~that~you~have~spelled~the~key~name~correctly.
3262 }

```

(End of definition for anskey-env-wrong, anskey-env-key-unknown, and anskey-env-key-value-unknown.)

13.35.3 Definition of messages for writing files in anskey*

These messages are not error messages, they report to the terminal what happens with the file written by the environment.

overwrite-file Messages used when **anskey*** writes the external file.

writing-file

not-writing

```

3263 \msg_new:nnn { enumext } { overwrite-file }
3264 {
3265   Overwriting~file~'#1'.
3266 }
3267 \msg_new:nnn { enumext } { writing-file }
3268 {
3269   Writing~file~'#1'.
3270 }
3271 \msg_new:nnn { enumext } { not-writing }
3272 {
3273   File~'#1'~already~exists.~Not~writing.
3274 }

```

(End of definition for overwrite-file, writing-file, and not-writing.)

13.35.4 Definition of keys for anskey*

break-col First, the same keys from the **\anskey** command are added along with the **force-eol**, **write-env** and **overwrite** keys that were in the original implementation that used the **scontents** support package for these.

item-join

item-star

item-sym*

item-pos*

```

3275 \keys_define:nn { enumext / anskey* }

```

©2024–2026 by Pablo González L

```

3276 {
3277     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3278     break-col .default:n = true,
3279     break-col .value_forbidden:n = true,
3280     item-join .int_set:N = \l__enumext_store_item_join_int,
3281     item-join .value_required:n = true,
3282     item-star .bool_set:N = \l__enumext_store_item_star_bool,
3283     item-star .default:n = true,
3284     item-star .value_forbidden:n = true,
3285     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
3286     item-sym* .value_required:n = true,
3287     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
3288     item-pos* .value_required:n = true,
3289     force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
3290     force-eol .initial:n = false,
3291     force-eol .default:n = true,
3292     write-env .code:n =
3293     {
3294         \bool_set_true:N \l__enumext_write_anskey_env_bool
3295         \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
3296     },
3297     write-env .value_required:n = true,
3298     overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
3299     overwrite .initial:n = false,
3300     overwrite .default:n = true,
3301     unknown .code:n = { \l__enumext_anskey_env_unknown:n {#1} },
3302 }

```

(End of definition for `break-col` and others.)

13.35.5 Internal functions for the environment

```

\__enumext_anskey_env_unknown:n
\__enumext_anskey_env_unknown:nn

```

The *⟨keys⟩* are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\__enumext_anskey_env_unknown:n`.

```

3303 \cs_new_protected:Nn \__enumext_anskey_env_unknown:n
3304 {
3305     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
3306 }
3307 \cs_new_protected:Nn \__enumext_anskey_env_unknown:nn
3308 {
3309     \tl_if_blank:nTF {#2}
3310     {
3311         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
3312     }
3313     {
3314         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3315     }
3316 }

```

(End of definition for `\__enumext_anskey_env_unknown:n` and `\__enumext_anskey_env_unknown:nn`.)

```

\__enumext_anskey_env_file_if_writable:n
\__enumext_anskey_env_file_if_writable:nT
\__enumext_anskey_env_file_if_writable:nF
\__enumext_anskey_env_file_if_writable:nTF
\__enumext_anskey_env_file_write:nn
\__enumext_anskey_env_file_write:VV

```

The conditional function `\__enumext_anskey_env_file_if_writable:n` is used by the `write-env` and `overwrite` keys in the `anskey*` environment to determine whether the output file is written or overwritten.

```

3317 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1
3318 { T, F, TF }
3319 {
3320     \bool_if:NTF \l__enumext_write_anskey_env_bool
3321     {
3322         \file_if_exist:nTF {#1}
3323         {
3324             \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
3325             {
3326                 \msg_warning:nne { enumext } { overwrite-file } {#1}
3327                 \prg_return_true:
3328             }
3329             {
3330                 \msg_warning:nne { enumext } { not-writing } {#1}
3331                 \prg_return_false:
3332             }
3333         }
3334     }

```

```

3335         \msg_warning:nne { enumext } { writing-file } {#1}
3336     \prg_return_true:
3337 }
3338 }
3339 { \prg_return_false: }
3340 }

```

The `\__enumext_anskey_env_file_write:n` function is used by the `write-env` key in the `anskey*` environment to write the output file with the `<body>` of the environment.

```

3341 \cs_new_protected:Nn \__enumext_anskey_env_file_write:n
3342 {
3343     \__enumext_anskey_env_file_if_writable:nT {#1}
3344     {
3345         \iow_open:Nn \__enumext_write_anskey_env_file_iow {#1}
3346         \iow_now:Nn \__enumext_write_anskey_env_file_iow {#2}
3347         \iow_close:N \__enumext_write_anskey_env_file_iow
3348     }
3349 }
3350 \cs_generate_variant:Nn \__enumext_anskey_env_file_write:n { VV }

```

(End of definition for `\__enumext_anskey_env_file_if_writable:n` and others.)

`\__enumext_anskey_env_safe_outer:` The function `\__enumext_anskey_env_safe_outer:` will return the appropriate messages when `anskey*` is executed outside the environment in which the `save-ans` key was activated or within the `keyans`, `keyans*` or `keyanspic` environments.

```

3351 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3352 {
3353     \bool_lazy_or:nnF { \__enumext_store_active_bool } { \__enumext_in_keyans_p: }
3354     {
3355         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3356     }
3357     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3358     {
3359         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
3360     }
3361     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
3362     {
3363         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3364     }
3365     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3366     {
3367         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3368     }
3369 }

```

The function `\__enumext_anskey_env_safe_inner:` will first check if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

3370 \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
3371 {
3372     \bool_if:NF \__enumext_item_number_bool
3373     {
3374         \msg_error:nn { enumext } { anskey-unnumber-item }
3375     }
3376     \mode_if_math:T
3377     {
3378         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
3379     }
3380 }

```

The `\__enumext_store_anskey_env:n` function will first pass the c-type argument `<body>` to the variable `\__enumext_store_anskey_env_tl` and replace the macro `\obeyedline` with `^^J` and then execute the `write-env` and `overwrite` keys, check the state of the variable `\__enumext_anskey_env_force_eol_bool` managed by the `force-eol` key, and adds `\c__enumext_anskey_env_hidden_space_str` if necessary. Finally it uses `\exp_args:Ne` on the `\__enumext_store_anskey_arg:n` to expand the `\tl_retokenize:n` function which rescans the `\__enumext_store_anskey_env_tl` variable before processing it.

```

3381 \cs_new_protected:Nn \__enumext_store_anskey_env:n
3382 {
3383     \tl_set:Nn \__enumext_store_anskey_env_tl {#1}
3384     \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^J }

```

```

3385 \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline }
3386 { \iow_char:N ^^J }
3387 \__enumext_anskey_env_file_write:VV
3388 \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
3389 \bool_if:NF \l__enumext_anskey_env_force_eol_bool
3390 {
3391   \tl_put_right:Ne \l__enumext_store_anskey_env_tl
3392   {
3393     \c__enumext_anskey_env_hidden_space_str
3394   }
3395 }
3396 \exp_args:Nc
3397 \__enumext_store_anskey_arg:n
3398 {
3399   \tl_retokenize:n { \l__enumext_store_anskey_env_tl }
3400 }
3401 }

```

Since `\obeyedline` can be redefined by the user, for example to `\mbox{}\par`, it is necessary to redefine it to `^^J` in order to use `\tl_replace_all:Nee` otherwise it returns an error.

(End of definition for `\__enumext_anskey_env_safe_outer:`, `\__enumext_anskey_env_safe_inner:`, and `\__enumext_store_anskey_env:n`.)

anskey* The environment accepts an optional `[⟨key = val⟩]` argument in ‘#1’ and a c-type argument in ‘#2’. First, the function `\__enumext_anskey_env_safe_outer:` is called to make sure where the environment is running, then the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store` is checked. If it is true, it looks for `[⟨key = val⟩]` and verifies that the *argument* c-type (*body*) is not empty. Finally, it runs the internal check function `\__enumext_anskey_env_safe_inner:n` and calls the function `\__enumext_store_anskey_arg:n`.

```

3402 \NewDocumentEnvironment{anskey*}{ o c }
3403 {
3404   \__enumext_anskey_env_safe_outer:
3405   \bool_if:NT \l__enumext_check_answers_bool
3406   {
3407     \tl_if_novalue:nF {#1}
3408     {
3409       \keys_set:nn { enumext / anskey* } {#1}
3410     }
3411     \tl_if_blank:nTF {#2}
3412     {
3413       \msg_error:nn { enumext } { anskey-empty-arg }
3414     }
3415     {
3416       \__enumext_anskey_env_safe_inner:
3417       \__enumext_store_anskey_env:n {#2}
3418     }
3419   }
3420 } { }

```

(End of definition for `anskey*`. This function is documented on page 16.)

13.36 Executing check-ans system and write .log

`\__enumext_execute_after_env:` The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then calls the `\__enumext_item_answer_diff:` function and then writes the values of the global variables used to the `.log` file. If the key `check-ans` is active, it executes the function `\__enumext_check_ans_show:`, which warns on the terminal if the numbers differ, otherwise it executes the function `\__enumext_check_ans_log:` and writes the results in the `.log` file, and finally executes the function `\__enumext_reset_global_vars:` returning the used variables to their original state.

```

3421 \cs_new_protected:Nn \__enumext_execute_after_env:
3422 {
3423   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3424   {
3425     \tl_if_empty:NF \g__enumext_store_name_tl
3426     {
3427       \__enumext_stop_save_ans_msg:
3428       \__enumext_item_answer_diff:
3429       \__enumext_log_global_vars:

```

```

3430         \__enumext_log_answer_vars:
3431         \bool_if:NTF \g__enumext_check_ans_key_bool
3432         {
3433             \__enumext_check_ans_show:
3434         }
3435         { \__enumext_check_ans_log: }
3436     }
3437     \__enumext_reset_global_vars:
3438 }
3439 }

```

🔗 This function is passed to the function `\__enumext_after_env:n` for the environments `enumext` (§13.43) and `enumext*` (§13.48) and it is executed only when the environments are not nested or at some level of these.

(End of definition for `\__enumext_execute_after_env:.`)

13.37 Common functions for `keyans`, `keyans*` and `keyanspic`

13.37.1 Definition of variables

Internal variables used by `mark-sep`, `show-ans` and `show-pos` keys (§13.31).

```

\l__enumext_mark_sep_tmpa_dim
\l__enumext_mark_sep_tmpb_dim
\l__enumext_show_pos_tmp_int
3440 \dim_new:N \l__enumext_mark_sep_tmpa_dim
3441 \dim_new:N \l__enumext_mark_sep_tmpb_dim
3442 \int_new:N \l__enumext_show_pos_tmp_int

```

(End of definition for `\l__enumext_mark_sep_tmpa_dim`, `\l__enumext_mark_sep_tmpb_dim`, and `\l__enumext_show_pos_tmp_int`.)

13.37.2 Definition of error messages

`keyans-wrong-level`
`wrong-place` Messages used by `keyans`, `keyans*` and `keyanspic` environments.

```

3443 \msg_new:nnnn { enumext } { keyans-wrong-level }
3444 {
3445     Wrong~level~position~for~'keyans'~\msg_line_context:.
3446 }
3447 {
3448     The~environment~'keyans'~can~only~be~in~the~first~level.
3449 }
3450 \msg_new:nnnn { enumext } { wrong-place }
3451 {
3452     Wrong~place~for~'#1'~environment~\msg_line_context:.
3453 }
3454 {
3455     '#1'~is~only~found~with~'#2'~in~'enumext'.
3456 }

```

(End of definition for `keyans-wrong-level` and `wrong-place`.)

13.37.3 Storing content in prop list

`\__enumext_keyans_addto_prop:n` The function `\__enumext_keyans_addto_prop:n` will pass the current $\langle label \rangle$ for `\item*` in `keyans` environment and the current $\langle label \rangle$ for `\anspic*` in `keyanspic` environment followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `\__enumext_store_addto_prop:V`.

```

3457 \cs_new_protected:Nn \__enumext_keyans_addto_prop:n
3458 {
3459     \tl_clear:N \l__enumext_store_current_label_tl
3460     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3461     {
3462         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3463     }
3464     {
3465         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3466     }

```

If the *optional argument* is present and the `save-sep` key is not empty, it is saved.

```

3467     \tl_if_novalue:nF {#1}
3468     {
3469         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3470         {
3471             \tl_put_right:NV \l__enumext_store_current_label_tl

```

```

3472         \l__enumext_store_keyans_item_opt_sep_v_tl
3473     }
3474     \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
3475 }
3476 \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3477 }

```

(End of definition for `\__enumext_keyans_addto_prop:n`.)

13.37.4 The save-ref key for keyans, keyans* and keyanspic

The “*internal label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has *slight differences* with the one implemented for `\anskey` basically because in this environments the interest is in the current `<label>` for `\item*` and `\anspic*` with the `<contents>` of the *optional argument*. The mechanism defined here allows executing `\ref{<store name : position>}` and will return `1. (A)`.

```

\__enumext_keyans_store_ref:
  \__enumext_keyans_store_ref_aux_i:
  \__enumext_keyans_store_ref_aux_ii:

```

The function `\__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the *save-ref* key for `\item*` and `\anspic*` commands. First, copies of the current `<labels>` are created and the dots “.” are removed from them, since double dots in references are not wanted.

```

3478 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3479 {
3480   \bool_if:NT \l__enumext_store_ref_key_bool
3481   {
3482     \cs_set_protected:Nn \__enumext_tmp:n
3483     {
3484       \tl_set_eq:cc { l__enumext_label_copy_ ##1 _tl } { l__enumext_label_ ##1 _tl }
3485       \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
3486       \tl_remove_once:cn { l__enumext_label_copy_ ##1 _tl } { . }
3487       \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
3488     }
3489     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3490     \__enumext_keyans_store_ref_aux_i:
3491   }
3492 }

```

The auxiliary function `\__enumext_keyans_store_ref_aux_i:` sets the variable `\l__enumext_newlabel_arg_one` which will contain `{<store name : position>}` analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

3493 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3494 {
3495   \bool_if:NT \g__enumext_starred_bool
3496   {
3497     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3498   }
3499   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3500   {
3501     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3502     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3503   }
3504   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3505   {
3506     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3507     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3508   }
3509   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3510   {
3511     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3512     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3513   }
3514   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3515   {
3516     \l__enumext_store_name_tl \c_colon_str
3517     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3518   }
3519   \__enumext_keyans_store_ref_aux_ii:
3520 }

```

Now the auxiliary function `\__enumext_keyans_store_ref_aux_ii:` saves the result in the variable `\l__enumext_write_aux_file_tl` and finally writes it in the `.aux` file.

```

3521 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:

```

```

3522 {
3523   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3524   {
3525     \__enumext_newlabel:nn
3526     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3527     { \l__enumext_newlabel_arg_two_tl }
3528   }
3529   \l__enumext_write_aux_file_tl
3530 }

```

(End of definition for `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, and `\__enumext_keyans_store_ref_aux_ii:`.)

13.37.5 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current *label* `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3531 \cs_new_protected:Nn \__enumext_keyans_addto_seq:n
3532 {
3533   \tl_clear:N \l__enumext_store_current_label_tl
3534   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3535   {
3536     \tl_put_right:Ne \l__enumext_store_current_label_tl
3537     { \item \l__enumext_label_vi_tl }
3538   }
3539   {
3540     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3541   }
3542   \tl_if_novalue:nF {#1}
3543   {
3544     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3545     {
3546       \tl_put_right:NV \l__enumext_store_current_label_tl
3547       \l__enumext_store_keyans_item_opt_sep_v_tl
3548     }
3549     \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
3550   }
3551   \__enumext_keyans_addto_seq_link:
3552 }

```

Checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `\__enumext_store_addto_seq:V` function. Finally, when the `check-ans` key is active, it increments the value of the integer variable `\g__enumext_item_anskey_int`.

```

3553 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3554 {
3555   \bool_lazy_and:nnT
3556   { \bool_if_p:N \l__enumext_store_ref_key_bool }
3557   { \bool_if_p:N \l__enumext_hyperref_bool }
3558   {
3559     \tl_put_right:Ne \l__enumext_store_current_label_tl
3560     {
3561       \hfill \exp_not:N \hyperlink
3562       {
3563         \exp_not:V \l__enumext_newlabel_arg_one_tl
3564       }
3565       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3566     }
3567   }
3568   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3569   \bool_if:NT \l__enumext_check_answers_bool
3570   {
3571     \int_gincr:N \g__enumext_item_anskey_int
3572   }
3573 }

```

(End of definition for `\__enumext_keyans_addto_seq:n` and `\__enumext_keyans_addto_seq_link:`.)

13.37.6 The show-ans and show-pos keys for keyans and keyanspic

The function `\__enumext_keyans_save_item_opt:n` will save the optional argument of `\item*` and `\anspic*` in the variable `\l__enumext_store_current_opt_arg_tl`.

```
\__enumext_keyans_save_item_opt:n
\__enumext_keyans_show_item_opt:
\__enumext_keyans_show_item_opt_viii:
```

```
3574 \cs_new_protected:Nn \__enumext_keyans_save_item_opt:n
3575 {
3576   \tl_if_novalue:nF {#1}
3577   {
3578     \tl_set:Nn \l__enumext_store_current_opt_arg_tl {#1}
3579   }
3580 }
```

The function `\__enumext_keyans_show_item_opt:` will print the optional arguments of `\item*` and `\anspic*` when the `show-ans` or `show-pos` keys are set next to the key `wrap-opt` in `keyans` and `keyanspic` environments.

```
3581 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3582 {
3583   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3584   {
3585     \bool_lazy_or:nnT
3586     { \bool_if_p:N \l__enumext_show_answer_bool }
3587     { \bool_if_p:N \l__enumext_show_position_bool }
3588     {
3589       \__enumext_keyans_wrapper_opt_v:n
3590       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3591     }
3592   }
3593 }
```

The function `\__enumext_keyans_show_item_opt_viii:` will print the optional argument of `\item*` when the `show-ans` or `show-pos` keys are set next to the key `wrap-opt` in `keyans*` environment.

```
3594 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3595 {
3596   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3597   {
3598     \bool_lazy_or:nnT
3599     { \bool_if_p:N \l__enumext_show_answer_bool }
3600     { \bool_if_p:N \l__enumext_show_position_bool }
3601     {
3602       \__enumext_keyans_wrapper_opt_viii:n
3603       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3604     }
3605   }
3606 }
```

(End of definition for `\__enumext_keyans_save_item_opt:n`, `\__enumext_keyans_show_item_opt:`, and `\__enumext_keyans_show_item_opt_viii:`.)

```
\__enumext_keyans_pos_mark_set:
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
```

The function `\__enumext_keyans_pos_mark_set:` adjusts the horizontal spaces for the `mark-sep*` key taking into account the value of the `align` key and the width of `\label`.

```
3607 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3608 {
3609   \__enumext_label_width_by_box:Nn
3610   \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3611   \str_case:Vn \l__enumext_align_label_pos_v_str
3612   {
3613     { l }
3614     {
3615       \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3616     }
3617     { r }
3618     {
3619       \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3620       { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3621     }
3622     { c }
3623     {
3624       \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3625       { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3626     }
3627   }
```

Sets the default values for the keys `mark-ans*`, `mark-sep*` and `mark-pos*`.

```

3628   \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3629   {
3630     \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3631   }
3632   \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3633   \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3634   \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3635 }

```

The function `\__enumext_keyans_show_ans:` will print the *(symbol)* set by the `mark-ans*` key when the `show-ans` key is active.

```

3636 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3637 {
3638   \bool_lazy_all:nT
3639   {
3640     { \bool_if_p:N \l__enumext_show_answer_bool }
3641     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3642   }
3643   {
3644     \__enumext_keyans_pos_mark_set:
3645     \__enumext_print_keyans_box:NN
3646     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3647   }
3648 }

```

The function `\__enumext_keyans_show_pos:` will print the *(position)* of the stored content in *prop list*. It is the number of elements of the prop list plus the value of `\l__enumext_show_pos_tmp_int`, which is incremented in each execution inside `keyans` and is zero in the other environments.

```

3649 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3650 {
3651   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3652   {
3653     \int_incr:N \l__enumext_show_pos_tmp_int
3654   }
3655   {
3656     \int_zero:N \l__enumext_show_pos_tmp_int
3657   }
3658   \bool_lazy_all:nT
3659   {
3660     { \bool_if_p:N \l__enumext_show_position_bool }
3661     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3662   }
3663   {
3664     \tl_set:Ne \l__enumext_mark_answer_sym_v_tl
3665     {
3666       \group_begin:
3667       \exp_not:N \normalfont
3668       \exp_not:N \footnotesize [ \int_eval:n
3669       {
3670         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3671         + \l__enumext_show_pos_tmp_int
3672       }
3673       ]
3674       \group_end:
3675     }
3676     \__enumext_keyans_pos_mark_set:
3677     \__enumext_print_keyans_box:NN
3678     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3679   }
3680 }

```

(End of definition for `\__enumext_keyans_pos_mark_set:`, `\__enumext_keyans_show_ans:`, and `\__enumext_keyans_show_pos:`.)

13.38 Redefining `\item` and `\makeLabel` in `enumext`

Redefining the `\item` command is not as simple as it seems. This command works in conjunction with the `\makeLabel` command, so both of them have to be redefined; in addition, a couple of *global* variables are needed to pass the values from one command to the other.

When *tagged* PDF is active `\makelabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is to redefine `\makelabel` using `\makebox`. The best way to implement this is to use the conditional command `\IfDocumentMetadataTF` to force this redefinition and the dedicated `mode-box` key to manually activate it by the user.

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext`, and `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]` are added.

`\g__enumext_item_symbol_aux_tl` Internal variable used by `item-sym*` key (§13.39).

```
3681 \tl_new:N \g__enumext_item_symbol_aux_tl
```

(End of definition for `\g__enumext_item_symbol_aux_tl`.)

`\__enumext_default_item:n`

First it checks if the *optional argument* is present; if it is NOT present, it checks the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, sets the boolean variable `\l__enumext_wrap_label_X_bool` to `true` for the key `wrap-label` and executes `\__enumext_item_std:w` and the key `itemindent`; otherwise it checks the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and executes `\__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```
3682 \cs_new_protected:Nn \__enumext_default_item:n
3683 {
3684   \tl_if_novalue:nTF {#1}
3685   {
3686     \bool_if:NT \l__enumext_check_answers_bool
3687     {
3688       \int_gincr:N \g__enumext_item_number_int
3689       \bool_set_true:N \l__enumext_item_number_bool
3690     }
3691     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3692     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3693   }
3694   {
3695     \bool_set_eq:cc
3696     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3697     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3698     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3699   }
3700 }
```

(End of definition for `\__enumext_default_item:n`.)

`\__enumext_item_starred_exec:nn`

`\__enumext_item_starred_exec:`

The `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]` work like the *numbered* `\item`, but placing a *⟨symbol⟩* to the “left” of the *⟨label⟩* separated from it by the value of the second *optional argument* *⟨offset⟩*.

```
#1: \l__enumext_item_symbol_X_tl
```

```
#2: \l__enumext_item_symbol_sep_X_dim
```

First a copy of `\l__enumext_item_symbol_X_tl` is made, which is set by the key `item-sym*` or passed as “first” *optional argument* in the global variable `\g__enumext_item_symbol_aux_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item-pos*` or by the “second” *optional argument*, then the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store` is checked, the boolean variable `\l__enumext_wrap_label_X_bool` is set to `true` for the key `wrap-label` and `\__enumext_item_std:w` and the key `itemindent` are executed.

```
3701 \cs_new_protected:Nn \__enumext_item_starred_exec:nn
3702 {
3703   \tl_if_novalue:nTF {#1}
3704   {
3705     \tl_gset_eq:Nc
3706     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3707   }
3708   {
3709     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3710   }
3711   \tl_if_novalue:nTF {#2}
3712   {
3713     \dim_set_eq:cc
3714     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3715     { \l__enumext_labelsep_ \__enumext_level: _dim }
3716   }
3717 }
```

```

3718         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3719     }
3720     \bool_if:NT \l__enumext_check_answers_bool
3721     {
3722         \int_gincr:N \g__enumext_item_number_int
3723         \bool_set_true:N \l__enumext_item_number_bool
3724     }
3725     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3726     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3727 }

```

The function `\__enumext_item_starred_exec:` will be responsible for executing `\item*` for the `enumext` environment.

```

3728 \cs_new_protected:Nn \__enumext_item_starred_exec:
3729 {
3730     \tl_if_empty:cf { l__enumext_item_symbol_ \__enumext_level: _tl }
3731     {
3732         \mode_leave_vertical:
3733         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_
3734             \__enumext_level: _dim } }
3735         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3736         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_
3737             \__enumext_level: _dim } }
3738     }
3739 }

```

(End of definition for `\__enumext_item_starred_exec:nn` and `\__enumext_item_starred_exec:.`)

`\__enumext_redefine_item:`

The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function is passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.43).

```

3740 \cs_new_protected:Nn \__enumext_redefine_item:
3741 {
3742     \RenewDocumentCommand \item { s o o }
3743     {
3744         \bool_if:nTF {##1}
3745         {
3746             \__enumext_item_starred_exec:nn {##2} {##3}
3747         }
3748         { \__enumext_default_item:n {##2} }
3749     }
3750 }

```

(End of definition for `\__enumext_redefine_item:.`)

`\__enumext_make_label:`
`\__enumext_make_label_std:`
`\__enumext_make_label_box:`

The function `\__enumext_make_label:` redefines `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function is passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.43).

```

3751 \cs_new_protected:Nn \__enumext_make_label:
3752 {
3753     \IfDocumentMetadataTF
3754     {
3755         \__enumext_make_label_box:
3756     }
3757     {
3758         \bool_if:NTF \l__enumext_mode_box_bool
3759         {
3760             \__enumext_make_label_box:
3761         }
3762         {
3763             \__enumext_make_label_std:
3764         }
3765     }
3766 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3767 \cs_new_protected:Nn \__enumext_make_label_std:
3768 {
3769     \RenewDocumentCommand \makelabel { m }
3770     {

```

```

3771 \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3772 \__enumext_item_starred_exec:
3773 \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3774 \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3775 {
3776   \use:c { __enumext_wrapper_label_ \__enumext_level: :n } {##1}
3777 }
3778 {##1}
3779 \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3780 \tl_gclear:N \g__enumext_item_symbol_aux_tl
3781 }
3782 }

```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

- Here it is necessary to use `\strut\smash` to maintain text *alignment* in case the user wants to use `\labelbx` for example. In experiments with *mimicking* the `description` environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic `list` environment package than `enumext`.

```

3783 \cs_new_protected:Nn \__enumext_make_label_box:
3784 {
3785   \RenewDocumentCommand \makelabel { m }
3786   {
3787     \strut\smash
3788     {
3789       \makebox
3790       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3791       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3792       {
3793         \__enumext_item_starred_exec:
3794         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3795         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3796         {
3797           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } {##1}
3798         }
3799         {##1}
3800         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3801       }
3802     } % close smash
3803   }
3804 }

```

(End of definition for `\__enumext_make_label:`, `\__enumext_make_label_std:`, and `\__enumext_make_label_box:`)

13.39 Setting item-sym* and item-pos* keys

In order to have a cleaner implementation of `\item*` for the `enumext` and `enumext*` environments it is best to define a couple of keys that allow controlling and setting by default the `<symbol>` and its `<offset>`.

item-sym* Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

item-pos* 3805 \cs_set_protected:Nn \__enumext_tmp:nn
3806 {
3807   \keys_define:nn { enumext / #1 }
3808   {
3809     item-sym* .tl_set:c = { l__enumext_item_symbol_ #2 _tl },
3810     item-sym* .value_required:n = true,
3811     item-sym* .initial:n = {\textborn},
3812     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_ #2 _dim },
3813     item-pos* .value_required:n = true,
3814   }
3815 }
3816 \clist_map_inline:nn
3817 {
3818   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3819 }
3820 { \__enumext_tmp:nn #1 }

```

(End of definition for `item-sym*` and `item-pos*`.)

13.40 Handling unknown keys

At this point in the code no more `<keys>` will be added. Since the definitions of environments and commands have already been quite *restrictive*, the only thing left to do is to be equally strict with the `<keys>` (consistency matters).

- Well, the paragraph above did not hold entirely: more *keys* had to be added than planned, and not everything turns out as expected.

13.40.1 Definition of error messages

starred-unknown-key
starred-unknown-key-value

Message used by unknown *keys* in *enumext** environment.

```
3821 \msg_new:nnnn { enumext } { starred-unknown-key }
3822 {
3823   The~key~'#1'~is~unknown~by~environment~
3824   '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
3825 }
3826 {
3827   The~environment~'\l__enumext_envir_name_tl'~does~not~
3828   have~a~key~called~'#1'.~
3829   Check~that~you~have~spelled~the~key~name~correctly.
3830 }
3831 \msg_new:nnnn { enumext } { starred-unknown-key-value }
3832 {
3833   The~key~'#1=#2'~is~unknown~by~environment~
3834   '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
3835 }
3836 {
3837   The~environment~'\l__enumext_envir_name_tl'~does~not~
3838   have~a~key~called~'#1'.~
3839   Check~that~you~have~spelled~the~key~name~correctly.
3840 }
```

(End of definition for starred-unknown-key and starred-unknown-key-value.)

standard-unknown-key
standard-unknown-key-value

Message used by unknown *keys* in *enumext* environment.

```
3841 \msg_new:nnnn { enumext } { standard-unknown-key }
3842 {
3843   The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl'~on~level~
3844   \int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored~
3845   \msg_line_context:.
3846 }
3847 {
3848   The~environment~'\l__enumext_envir_name_tl'~does~not~
3849   have~a~key~called~'#1'~on~level~\int_use:N \l__enumext_level_int.~
3850   Check~that~you~have~spelled~the~key~name~correctly.
3851 }
3852 \msg_new:nnnn { enumext } { standard-unknown-key-value }
3853 {
3854   The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl'~on~level~
3855   \int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored~
3856   \msg_line_context:.
3857 }
3858 {
3859   The~environment~'\l__enumext_envir_name_tl'~does~not~
3860   have~a~key~called~'#1'~on~level~\int_use:N \l__enumext_level_int.~
3861   Check~that~you~have~spelled~the~key~name~correctly.
3862 }
```

(End of definition for standard-unknown-key and standard-unknown-key-value.)

keyans-unknown-key
keyans-unknown-key-value

Message used by unknown *keys* in *keyans*, *keyans** and *keyanspic* environments.

```
3863 \msg_new:nnnn { enumext } { keyans-unknown-key }
3864 {
3865   The~key~'#1'~is~unknown~by~environment~
3866   '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
3867 }
3868 {
3869   The~environment~'\l__enumext_envir_name_tl'~does~not~
3870   have~a~key~called~'#1'.~
3871   Check~that~you~have~spelled~the~key~name~correctly.
3872 }
3873 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
3874 {
3875   The~key~'#1=#2'~is~unknown~by~environment~
3876   '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
```

```

3877     }
3878     {
3879         The~environment~'\__enumext_envir_name_tl'~does~not~
3880         have~a~key~called~'#1'.~
3881         Check~that~you~have~spelled~the~key~name~correctly.
3882     }

```

(End of definition for `keyans-unknown-key` and `keyans-unknown-key-value`.)

13.40.2 Handling unknown keys for `keyans`, `keyans*` and `keyanspic`

Define and set `unknown` key for `keyans`, `keyans*` and `keyanspic` environments. Here it is necessary to set `\__enumext_envir_name_tl` in case an `unknown` key is passed using `\setenumext`.

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn

```

```

3883 \cs_set_protected:Nn \__enumext_tmp:n
3884 {
3885     \keys_define:nn { enumext / #1 }
3886     {
3887         unknown .code:n =
3888         {
3889             \tl_set:Nn \__enumext_envir_name_tl {#1}
3890             \__enumext_keyans_unknown_keys:n {##1}
3891         },
3892     }
3893 }
3894 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3895 \cs_new_protected:Nn \__enumext_keyans_unknown_keys:n
3896 {
3897     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3898 }
3899 \cs_new_protected:Nn \__enumext_keyans_unknown_keys:nn
3900 {
3901     \tl_if_blank:nTF {#2}
3902     {
3903         \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3904     }
3905     {
3906         \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3907     }
3908 }

```

(End of definition for `unknown`, `\__enumext_keyans_unknown_keys:n`, and `\__enumext_keyans_unknown_keys:nn`.)

13.40.3 Handling unknown keys for `enumext*`

Define and set `unknown` key for `enumext*` environment.

```

3909 \keys_define:nn { enumext / enumext* }
3910 {
3911     unknown .code:n =
3912     {
3913         \tl_set:Nn \__enumext_envir_name_tl { enumext* }
3914         \__enumext_starred_unknown_keys:n {#1}
3915     },
3916 }

```

(End of definition for `unknown`.)

```

\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn

```

Internal functions for handling `unknown` key.

```

3917 \cs_new_protected:Nn \__enumext_starred_unknown_keys:n
3918 {
3919     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3920 }
3921 \cs_new_protected:Nn \__enumext_starred_unknown_keys:nn
3922 {
3923     \tl_if_blank:nTF {#2}
3924     {
3925         \msg_error:nne { enumext } { starred-unknown-key } {#1}
3926     }
3927     {
3928         \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}

```

```

3929     }
3930 }

```

(End of definition for `\__enumext_starred_unknown_keys:n` and `\__enumext_starred_unknown_keys:nn`.)

13.40.4 Handling unknown keys for enumext

`unknown` Defines and sets the key `unknown` for `enumext` environment.

```

\__enumext_standard_unknown_keys:n
\__enumext_standard_unknown_keys:nn
3931 \cs_set_protected:Nn \__enumext_tmp:n
3932 {
3933   \keys_define:nn { enumext / level-#1 }
3934   {
3935     unknown .code:n =
3936     {
3937       \int_set:Nn \__enumext_level_int {#1}
3938       \tl_set:Nc \__enumext_envir_name_tl { enumext }
3939       \__enumext_standard_unknown_keys:n {#1}
3940     },
3941   }
3942 }
3943 \clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3944 \cs_new_protected:Nn \__enumext_standard_unknown_keys:n
3945 {
3946   \exp_args:NV \__enumext_standard_unknown_keys:nn \l_keys_key_str {#1}
3947 }
3948 \cs_new_protected:Nn \__enumext_standard_unknown_keys:nn
3949 {
3950   \tl_if_blank:nTF {#2}
3951   {
3952     \msg_error:nne { enumext } { standard-unknown-key } {#1}
3953   }
3954   {
3955     \msg_error:nnee { enumext } { standard-unknown-key-value } {#1} {#2}
3956   }
3957 }

```

(End of definition for `unknown`, `\__enumext_standard_unknown_keys:n`, and `\__enumext_standard_unknown_keys:nn`.)

13.41 Redefining `\item` and `\makeLabel` in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current *⟨label⟩* next to the *⟨content⟩* if it is present in the *sequence* and *prop list* defined by `save-ans` key.

`\__enumext_keyans_default_item:n` The function `\__enumext_keyans_default_item:n` executes the original behavior of the `\item` along with the keys `wrap-label`, `wrap-label*` and `itemindent`.

```

3958 \cs_new_protected:Nn \__enumext_keyans_default_item:n
3959 {
3960   \tl_if_novalue:nTF {#1}
3961   {
3962     \bool_set_true:N \__enumext_wrap_label_v_bool
3963     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3964   }
3965   {
3966     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3967     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3968   }
3969 }

```

(End of definition for `\__enumext_keyans_default_item:n`.)

`\__enumext_keyans_starred_item:n` The function `\__enumext_keyans_starred_item:n` will take as argument ‘`#1`’ the *optional argument* *[⟨*content*⟩]* passed to `\item*` and save it via the `\__enumext_keyans_save_item_opt:n` function, then activate the `wrap-label` key, execute `\item` using `\__enumext_item_std:w`, the `itemindent` key and print the *optional argument* using the `\__enumext_keyans_show_item_opt:` function handled by the `wrap-opt` key.

```

3970 \cs_new_protected:Nn \__enumext_keyans_starred_item:n
3971 {

```

```

3972     \__enumext_keyans_save_item_opt:n {#1}
3973     \bool_set_true:N \l__enumext_wrap_label_v_bool
3974     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3975     \__enumext_keyans_show_item_opt:

```

Now store the current *label* first in the *prop* list (including the *optional argument*), run the internal “*label and ref*” system if the *save-ref* key is active, then store in the *sequence* and finally it increments `\g__enumext_check_starred_cmd_int` for internal check system.

```

3976     \__enumext_keyans_addto_prop:n {#1}
3977     \__enumext_keyans_store_ref:
3978     \__enumext_keyans_addto_seq:n {#1}
3979     \int_gincr:N \g__enumext_check_starred_cmd_int
3980 }

```

(End of definition for `\__enumext_keyans_starred_item:n`.)

```

\item*
\__enumext_keyans_redefine_item:

```

The function `\__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `\__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here the variable `\l__enumext_item_wrap_key_bool` used by the `wrap-ans*` key is set to true only when `\item*` is executed and, additionally, it is necessary to use `\peek_remove_spaces:n` to avoid an unwanted space when using `\item*` together with the `itemindent` key. This function is passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.42).

```

3981 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3982 {
3983     \RenewDocumentCommand \item { s o }
3984     {
3985         \bool_if:nTF {##1}
3986         {
3987             \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3988             \peek_remove_spaces:n
3989             {
3990                 \__enumext_keyans_starred_item:n {##2}
3991             }
3992         }
3993         {
3994             \bool_set_false:N \l__enumext_item_wrap_key_bool
3995             \__enumext_keyans_default_item:n {##2}
3996         }
3997     }
3998 }

```

(End of definition for `\item*` and `\__enumext_keyans_redefine_item:`. This function is documented on page 6.)

```

\__enumext_keyans_make_label:
\__enumext_keyans_wrapper_label:n
\__enumext_keyans_make_label_std:
\__enumext_keyans_make_label_box:
\__enumext_wrapper_label_v:n

```

The function `\__enumext_keyans_make_label:` redefines `\makeLabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*`, `wrap-ans*` and `\item*` for `keyans` environment. This function is passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.42).

```

3999 \cs_new_protected:Nn \__enumext_keyans_make_label:
4000 {
4001     \IfDocumentMetadataTF
4002     {
4003         \__enumext_keyans_make_label_box:
4004     }
4005     {
4006         \bool_if:NTF \l__enumext_mode_box_bool
4007         {
4008             \__enumext_keyans_make_label_box:
4009         }
4010         {
4011             \__enumext_keyans_make_label_std:
4012         }
4013     }
4014 }

```

Conditionals are added to the `\__enumext_keyans_wrapper_label:n` function to handle the keys `wrap-ans*`, `wrap-label` and `wrap-label*`.

```

4015 \cs_new_protected:Nn \__enumext_keyans_wrapper_label:n
4016 {
4017     \bool_lazy_all:nT
4018     {

```

```

4019         { \bool_if_p:N \l__enumext_wrap_label_v_bool          }
4020         { \bool_if_p:N \l__enumext_show_answer_bool          }
4021         { \bool_if_p:N \l__enumext_item_wrap_key_bool        }
4022         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
4023     }
4024     {
4025         \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
4026     }
4027     \bool_if:NTF \l__enumext_wrap_label_v_bool
4028     {
4029         \__enumext_wrapper_label_v:n {#1}
4030     }
4031     {#1}
4032 }

```

Standard definition when `\DocumentMetadata` is not active.

```

4033 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
4034 {
4035     \RenewDocumentCommand \makeLabel { m }
4036     {
4037         \tl_use:N \l__enumext_label_fill_left_v_tl
4038         \__enumext_keyans_show_ans:
4039         \__enumext_keyans_show_pos:
4040         \tl_use:N \l__enumext_label_font_style_v_tl
4041         \__enumext_keyans_wrapper_label:n {##1}
4042         \tl_use:N \l__enumext_label_fill_right_v_tl
4043     }
4044 }

```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

```

4045 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
4046 {
4047     \RenewDocumentCommand \makeLabel { m }
4048     {
4049         \strut\smash
4050         {
4051             \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
4052             {
4053                 \__enumext_keyans_show_ans:
4054                 \__enumext_keyans_show_pos:
4055                 \tl_use:N \l__enumext_label_font_style_v_tl
4056                 \__enumext_keyans_wrapper_label:n {##1}
4057             }
4058         }
4059     }
4060 }

```

(End of definition for `\__enumext_keyans_make_label:` and others.)

13.42 Second argument of the lists

At this point in the code most of the tools needed to create a *custom list* environment have already been programmed. Remember that the `\__enumext_start_list:nn` function takes two arguments: the “first” one is ready, and the “second” one is defined for all levels of the `enumext` environment, the `keyans` environment and the `enumext*` and `keyans*` environments.

Here the `\__enumext_list_arg_two_X:` function is implemented, which is responsible for setting all the list parameters, the counter, the redefinition of `\item`, `\makeLabel` along with the keys `ref`, `itemindent` and `show-length`.

13.42.1 Definition of variables

The dimension `\l__enumext_itemindent_X_dim` is set by the function `\__enumext_calc_hspace:NNNNNNN` (§13.42.4).

```

4061 \cs_set_protected:Nn \__enumext_tmp:n
4062 {
4063     \dim_new:c { l__enumext_itemindent_ #1 _dim }
4064 }
4065 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_itemindent_X_dim`.)

13.42.2 Definition of error messages

width-negative Messages used in length calculation.

```
4966 \msg_new:nnn { enumext } { width-negative }
4967 {
4968   Ignoring~negative~value~'#1=#2'~\msg_line_context:.\
4969   The~key~'#1'~accepts~values~>=~opt.
4970 }
```

(End of definition for width-negative.)

13.42.3 Definition of messages for the show-length key

The messages that print the lengths of the lists to the terminal; they are not error messages.

list-lengths Messages used by show-length key in enumext.

```
4971 \msg_new:nnn { enumext } { list-lengths }
4972 {
4973   ****~Lengths~used~by~'#1'enumext'~level~'#2'~
4974   \msg_line_context:~\c_space_tl ****\
4975   \__enumext_show_length:nnn { dim } { labelsep } { #1 }
4976   \__enumext_show_length:nnn { dim } { labelwidth } { #1 }
4977   \__enumext_show_length:nnn { dim } { itemindent } { #1 }
4978   \__enumext_show_length:nnn { dim } { leftmargin } { #1 }
4979   \__enumext_show_length:nnn { dim } { rightmargin } { #1 }
4980   \__enumext_show_length:nnn { dim } { listparindent } { #1 }
4981   \__enumext_show_length:nnn { skip } { topsep } { #1 }
4982   \__enumext_show_length:nnn { skip } { parsep } { #1 }
4983   \__enumext_show_length:nnn { skip } { partopsep } { #1 }
4984   \__enumext_show_length:nnn { skip } { itemsep } { #1 }
4985   ****~
4986 }
```

(End of definition for list-lengths.)

list-lengths-not-nested Messages used by show-length key in enumext*, keyans* and keyans.

```
4987 \msg_new:nnn { enumext } { list-lengths-not-nested }
4988 {
4989   ****~Lengths~used~by~'#2'~environment~
4990   \msg_line_context:~\c_space_tl ****\
4991   \__enumext_show_length:nnn { dim } { labelsep } { #1 }
4992   \__enumext_show_length:nnn { dim } { labelwidth } { #1 }
4993   \__enumext_show_length:nnn { dim } { itemindent } { #1 }
4994   \__enumext_show_length:nnn { dim } { leftmargin } { #1 }
4995   \__enumext_show_length:nnn { dim } { rightmargin } { #1 }
4996   \__enumext_show_length:nnn { dim } { listparindent } { #1 }
4997   \__enumext_show_length:nnn { skip } { topsep } { #1 }
4998   \__enumext_show_length:nnn { skip } { parsep } { #1 }
4999   \__enumext_show_length:nnn { skip } { partopsep } { #1 }
5000   \__enumext_show_length:nnn { skip } { itemsep } { #1 }
5001   ****~
5002 }
```

(End of definition for list-lengths-not-nested.)

13.42.4 Calculation of \leftmargin and \itemindent

Consider the figure 9 where the default margins (on the left) of a list are represented.

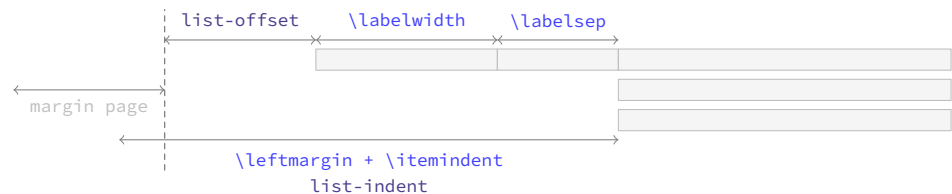
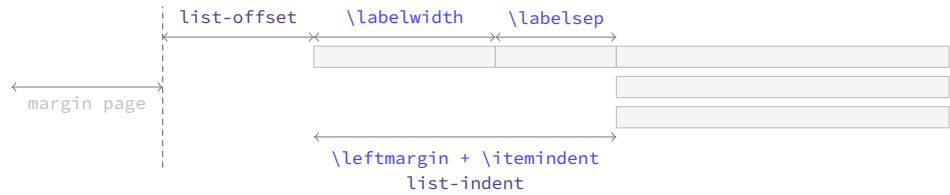
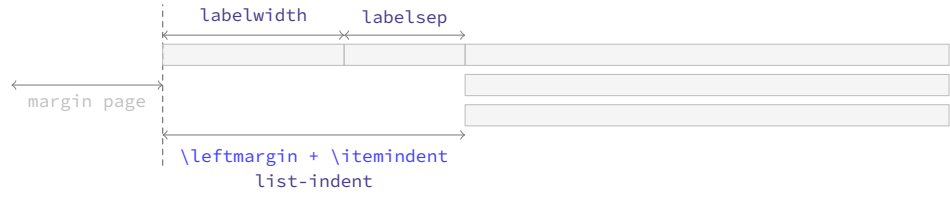


Figure 9: Representation of standard horizontal lengths in list environment.

The idea is to have control over these margins so that the list does not overlap the left margin of the page. The key relationship is that the “right edge” of the `\labelsep` equals the “right edge” of the `\itemindent`, so that the left edge of the “label box” is at `\leftmargin+\itemindent` minus `\labelwidth+\labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

Where the default values will look like in the figure 11.

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

The function `\__enumext_calc_hspace:NNNNNNN` takes seven arguments to be able to determine horizontal spaces for all list environment:

```
#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim
#3: \l__enumext_listoffset_X_dim
#4: \l__enumext_leftmargin_tmp_X_dim
#5: \l__enumext_leftmargin_X_dim
#6: \l__enumext_itemindent_X_dim
#7: \l__enumext_leftmargin_tmp_X_bool
```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

```
4103 \cs_new_protected:Nn \__enumext_calc_hspace:NNNNNNN
4104 {
4105   \dim_compare:nNnT {#1} < { \c_zero_dim }
4106   {
4107     \msg_warning:nnnV { enumext } { width-negative } { labelwidth } #1
4108     \dim_set:Nn #1 { \dim_abs:n {#1} }
4109   }
4110   \dim_compare:nNnT {#2} < { \c_zero_dim }
4111   {
4112     \msg_warning:nnnV { enumext } { width-negative } { labelsep } #2
4113     \dim_set:Nn #2 { \dim_abs:n {#2} }
4114   }
4115 }
```

If no value has been passed to the `labelwidth` and `labelsep` keys, the default values are set for `\l__enumext_leftmargin_tmp_X_dim`.

```
4115 \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

The cases are now analyzed and the values for `\leftmargin` and `\itemindent` are set.

```
4116 \dim_compare:nNnTF {#4} < { \c_zero_dim }
4117 {
4118   \dim_set:Nn #6 { #1 + #2 - #4 }
4119   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
4120 }
4121 {
4122   \dim_compare:nNnT {#4} = { #1 + #2 }
4123   { \dim_set:Nn #6 { \c_zero_dim } }
4124   \dim_compare:nNnT {#4} < { #1 + #2 }
4125   { \dim_set:Nn #6 { #1 + #2 - #4 } }
4126   \dim_compare:nNnT {#4} > { #1 + #2 }
4127   {
4128     \dim_set:Nn #6 { -#1 - #2 + #4 }
4129     \dim_set:Nn #6 { #6*-1 }
4130   }
4131   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
4132 }
4133 }
4134 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

(End of definition for `\__enumext_calc_hspace:NNNNNNN`.)

13.42.5 Setting second argument of the lists

`\leftmargini`, `\leftmarginii`, `\leftmarginiii` and `\leftmarginiv` are “not set”; in this case, the parameters for vertical and horizontal list spacing per level are set directly.

```

4135 \cs_set_protected:Nn \__enumext_tmp:n
4136 {
4137   \cs_new_protected:cpn { __enumext_list_arg_two_ #1 : }
4138   {
4139     \__enumext_calc_hspace:ccccc
4140     { \__enumext_labelwidth_ #1 _dim } { \__enumext_labelsep_ #1 _dim }
4141     { \__enumext_listoffset_ #1 _dim } { \__enumext_leftmargin_tmp_ #1 _dim }
4142     { \__enumext_leftmargin_ #1 _dim } { \__enumext_itemindent_ #1 _dim }
4143     { \__enumext_leftmargin_tmp_ #1 _bool }
4144     \clist_map_inline:nn
4145       { labelsep, labelwidth, itemindent, leftmargin, rightmargin,
4146         listparindent }
4147       { \dim_set_eq:cc {####1} { \__enumext_####1 _ #1 _dim } }
4148     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
4149       { \skip_set_eq:cc {####1} { \__enumext_####1 _ #1 _skip } }
4150     \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
4151       { \int_set_eq:cc {@####1} { \__enumext_####1 _ #1 _int } }
4152     \usecounter { enumX #1 }
4153     \setcounter { enumX #1 }
4154     { \int_eval:n { \int_use:c { \__enumext_start_ #1 _int } - 1 } }
4155     \str_if_eq:nnTF {#1} { v }
4156     {
4157       \__enumext_keyans_redefine_item:
4158       \__enumext_keyans_make_label:
4159       \__enumext_keyans_ref:
4160       \__enumext_keyans_fake_item_indent:
4161       \bool_if:cT { \__enumext_show_length_ #1 _bool }
4162       {
4163         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v }
4164         { keyans }
4165       }
4166     }
4167     {
4168       \__enumext_redefine_item:
4169       \__enumext_make_label:
4170       \__enumext_standard_ref:
4171       \__enumext_fake_item_indent:
4172       \bool_if:cT { \__enumext_show_length_ #1 _bool }
4173       {
4174         \msg_term:nnne { enumext } { list-lengths } {#1}
4175         { \int_use:N \__enumext_level_int }
4176       }
4177     }
4178   }
4179 }
4180 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for `\__enumext_list_arg_two_i`: and others.)

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. At this point the `parsep` key is modified to make it take the value of the `itemsep` key and later, in the environment definition, `\parindent` is modified to make it set the value of `listparindent` and `parsep` to set the value of `\parskip` locally.

🌱 For compatibility with *tagged* PDF and L^AT_EX release 2026-06-01, `\setcounter` is moved to `\begin{list}{\langle arg one \rangle}{\langle arg two \rangle}` in the definition of the environments `enumext*` and `keyans*` (§13.48, §13.49). This resolves the nesting problem reported by Tim Prescott (@teepeeemm) in `enumext*` nested in `enumext` starts at 2 with DocumentMetadata this was caused by the `latex-lab-block`[15] update in this release.

```

4181 \cs_set_protected:Nn \__enumext_tmp:n
4182 {
4183   \cs_new_protected:cpn { __enumext_list_arg_two_ #1 : }
4184   {
4185     \bool_set_true:c { \__enumext_leftmargin_tmp_ #1 _bool }
4186     \dim_zero:c { \__enumext_leftmargin_tmp_ #1 _dim }
4187     \__enumext_calc_hspace:ccccc
4188     { \__enumext_labelwidth_ #1 _dim } { \__enumext_labelsep_ #1 _dim }

```

```

4189     { \__enumext_listoffset_ #1 _dim } { \__enumext_leftmargin_tmp_ #1 _dim }
4190     { \__enumext_leftmargin_ #1 _dim } { \__enumext_itemindent_ #1 _dim }
4191     { \__enumext_leftmargin_tmp_ #1 _bool }
4192 \clist_map_inline:nn
4193   { labelsep, labelwidth, itemindent, leftmargin, rightmargin,
4194     listparindent }
4195   { \dim_set_eq:cc {####1} { \__enumext_####1 _ #1 _dim } }
4196 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
4197   { \skip_set_eq:cc {####1} { \__enumext_####1 _ #1 _skip } }
4198 \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
4199   { \int_set_eq:cc {@####1} { \__enumext_####1 _ #1 _int } }
4200 \skip_set_eq:Nc \parsep { \__enumext_itemsep_ #1 _skip }
4201 \skip_zero:N \partopsep
4202 \usecounter { enumX #1 }
4203 % \setcounter { enumX #1 }
4204 % { \int_eval:n { \int_use:c { \__enumext_start_ #1 _int } - 1 } }
4205 \__enumext_starred_ref:
4206 \str_if_eq:nnTF {#1} { vii }
4207   {
4208     \__enumext_fake_item_indent_vii:
4209     \bool_if:cT { \__enumext_show_length_vii_bool }
4210       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii }
4211         { enumext* } }
4212   }
4213   {
4214     \__enumext_fake_item_indent_viii:
4215     \bool_if:cT { \__enumext_show_length_ #1 _bool }
4216       { \msg_term:nnnn { enumext } { list-lengths-not-nested } {#1}
4217         { keyans* } }
4218   }
4219 }
4220 }
4221 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

13.43 The environment enumext

13.43.1 Definition of variables for enumext

__enumext_store_upper_level_X_bool

Internal variables used by the “*storing structure*” of the `enumext` environment when it is nested.

```

4222 \cs_set_protected:Nn \__enumext_tmp:n
4223   {
4224     \bool_new:c { \__enumext_store_upper_level_ #1 _bool }
4225   }
4226 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_store_upper_level_X_bool.)

13.43.2 Definition of error messages for enumext

list-too-deep

Message for the nesting depth of the environment `enumext`.

```

4227 \msg_new:nnnn { enumext } { list-too-deep }
4228   {
4229     Too~deep~nesting~for~'enumext'~\msg_line_context:.
4230   }
4231   {
4232     The~maximum~level~of~nesting~is~4.
4233   }

```

(End of definition for list-too-deep.)

13.43.3 Internal functions for enumext

__enumext_safe_exec:

The `\__enumext_safe_exec:` function first calls the function `\__enumext_is_not_nested:` which sets `\g__enumext_standard_bool` to `true` if it is NOT nested within `enumext*`, then calls the function `\__enumext_internal_mini_page:` to create the environment `\__enumext_mini_page`, increments `\l__enumext_level_int` to restrict nesting of the environment, sets `\l__enumext_standard_bool` to `true` and finally calls the function `\__enumext_is_on_first_level:` which sets `\l__enumext_standard_first_level` to `true` only if the environment is NOT nested and is at the “*first level*”.

```

4234 \cs_new_protected:Nn \__enumext_safe_exec:

```

```

4235 {
4236   \__enumext_is_not_nested:
4237   \__enumext_internal_mini_page:
4238   \int_incr:N \l__enumext_level_int
4239   \int_compare:nNnT { \l__enumext_level_int } > { 4 }
4240   { \msg_fatal:nn { enumext } { list-too-deep } }
4241   \bool_set_true:N \l__enumext_standard_bool
4242   \bool_set_false:N \l__enumext_starred_bool
4243   \__enumext_is_on_first_level:
4244 }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n

The __enumext_parse_keys:n function first clears the variable \l__enumext_series_name_str used by the key `series` and then checks if it is at the “first level”; if so, it processes the `<keys>` and then executes the function __enumext_parse_series:n used by the key `series` and calls the function __enumext_nested_base_line_fix: used by the key `base-fix`; otherwise it passes the `<keys>` to the inner levels of the environment, then executes the function __enumext_store_active_keys:n and reprocesses the `<keys>` to pass them to the `sequence` if the key `save-key` is not active.

```

4245 \cs_new_protected:Nn \__enumext_parse_keys:n
4246 {
4247   \tl_if_novalue:nF {#1}
4248   {
4249     \str_clear:N \l__enumext_series_name_str
4250     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
4251     {
4252       \keys_set:nn { enumext / level-1 } {#1}
4253       \bool_if:NF \l__enumext_print_keyans_cmd_bool
4254       {
4255         \__enumext_parse_series:n {#1}
4256       }
4257       \__enumext_nested_base_line_fix:
4258     }
4259     {
4260       \exp_args:Ne \keys_set:nn
4261       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
4262       \bool_if:NF \l__enumext_print_keyans_cmd_bool
4263       {
4264         \__enumext_parse_series:n {#1}
4265       }
4266     }
4267     \__enumext_store_active_keys:n {#1}
4268   }
4269 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level:

The __enumext_start_store_level: function activates the “storing structure” mechanism in the `sequence` for the command \anskey and the environment `anskey*`.

```

4270 \cs_new_protected:Nn \__enumext_start_store_level:
4271 {
4272   \bool_lazy_all:nT
4273   {
4274     { \bool_if_p:N \l__enumext_store_active_bool }
4275     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
4276     { \bool_if_p:N \g__enumext_standard_bool }
4277   }
4278   {
4279     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4280     {
4281       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
4282       \__enumext_store_level_open:
4283     }
4284   }

```

If `enumext` are nested in `enumext*` add __enumext_store_level_open: to preserve the “storing structure”.

```

4285   \bool_lazy_all:nT
4286   {
4287     { \bool_if_p:N \l__enumext_store_active_bool }
4288     { \bool_not_p:n { \l__enumext_keyans_env_bool } }

```

```

4289     { \int_compare:nNn { \l__enumext_level_h_int } = { 1 } }
4290   }
4291   {
4292     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4293     {
4294       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
4295       \__enumext_store_level_open:
4296     }
4297   }
4298 }

```

(End of definition for `\__enumext_start_store_level:`)

`\__enumext_stop_store_level:` The `\__enumext_stop_store_level:` function stops the “storing structure” mechanism in the sequence for the command `\anskey` and the environment `anskey*`.

```

4299 \cs_new_protected:Nn \__enumext_stop_store_level:
4300 {
4301   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
4302   {
4303     \__enumext_store_level_close:
4304   }
4305 }

```

(End of definition for `\__enumext_stop_store_level:`)

`\__enumext_multicols_start:` The function `\__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=opt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

4306 \cs_new_protected:Nn \__enumext_multicols_start:
4307 {
4308   \int_compare:nNnT
4309   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
4310   {
4311     \dim_compare:nNnT
4312     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
4313     {
4314       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
4315       {
4316         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
4317           + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
4318         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
4319         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
4320       }
4321     }
4322     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
4323     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4324     {
4325       \dim_zero:N \columnseprule
4326     }

```

The *vertical spacing* settings for the `multicols` environment are calculated using the function `\__enumext_multi_advspace:`, the “vertical adjust spacing” is applied, and then the `multicols` environment is started.

```

4327     \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
4328     {
4329       \skip_zero:N \multicolsep
4330       \__enumext_multi_advspace:
4331     }
4332     \raggedcolumns
4333     \begin{multicols}{ \int_use:c { \l__enumext_columns_ \__enumext_level: _int } }
4334   }
4335 }

```

(End of definition for `\__enumext_multicols_start:`)

`\__enumext_multicols_stop:` The function `\__enumext_multicols_stop:` will stop the `multicols` environment and apply the “vertical adjust” spacing. When the `mini-env` key is active the space is not added here: it is stored with `\__enumext_mini_capture_multicols:` to be applied by `\__enumext_mini_apply_end:` after the environment on the “right side”. For compatibility with *tagged* PDF, the closing of the `list` environment is executed here along with `\__enumext_stop_store_level:`.

```

4336 \cs_new_protected:Nn \__enumext_multicols_stop:

```

```

4337 {
4338   \int_compare:nNnTF
4339     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
4340   {
4341     \__enumext_stop_list:
4342     \__enumext_stop_store_level:
4343     \end{multicols}
4344     \__enumext_unskip_unkern:
4345     \__enumext_unskip_unkern:
4346     \par
4347     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
4348     {
4349       \__enumext_mini_capture_multicols:
4350     }
4351     {
4352       \addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
4353     }
4354   }
4355   {
4356     \__enumext_stop_list:
4357     \__enumext_stop_store_level:
4358   }
4359 }

```

(End of definition for `\__enumext_multicols_stop:`.)

`\__enumext_before_list:` The function `\__enumext_before_list:` first calls the function `\__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `\__enumext_before_args_exec:` used by the key `before*` and finally execute the function `\__enumext_check_ans_active:` for the check answer mechanism.

```

4360 \cs_new_protected:Nn \__enumext_before_list:
4361 {
4362   \__enumext_vspace_above:
4363   \__enumext_before_args_exec:
4364   \__enumext_check_ans_active:

```

When the `mini-env` key is active it will set the value of the `\__enumext_minipage_right_X_dim` to be the *width* of the `\__enumext_mini_page` environment on the “right side”, using this value together with the value of the `\__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\__enumext_minipage_left_X_dim` will be set, which will be the *width* of `\__enumext_mini_page` environment on the “left side”, always having a current `\linewidth` as *maximum width* between them.

```

4365 \dim_compare:nNnT
4366 { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
4367 {
4368   \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
4369   {
4370     \linewidth
4371     - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
4372     - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
4373   }

```

The boolean variable `\__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `\__enumext_minipage_add_space:` is called and the `\__enumext_mini_page` environment on the “left side” will be initialized followed by the “vertical spacing” applied to preserve the “baseline” between the *left* and *right* side environments. After these actions, the function `\__enumext_multicols_start:` is called to handle the `multicols` environment.

```

4374   \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
4375   \int_gincr:N \g__enumext_minipage_stat_int
4376   \__enumext_minipage_add_space:
4377   \noindent
4378   \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
4379 }
4380 \__enumext_multicols_start:
4381 }

```

(End of definition for `\__enumext_before_list:`.)

`\__enumext_second_part:` The function `\__enumext_second_part:` first checks the state of the boolean variable `\__enumext_minipage_active` if it is `true`, a small test is executed to check if the use of `\miniright` was omitted (the

`\__enumext_mini_page` environment has not been closed), then it closes `\__enumext_mini_page` and adds the stored *adjusted vertical space* using `\__enumext_mini_apply_end:`, otherwise it closes the `\multicols` environment.

```

4382 \cs_new_protected:Nn \__enumext_second_part:
4383 {
4384   \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
4385   {
4386     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
4387     {
4388       \msg_warning:nn { enumext } { missing-miniright }
4389       \miniright
4390     }
4391     \int_gzero:N \g__enumext_minipage_stat_int
4392     \__enumext_unskip_unkern: % remove topsep + [partopsep]
4393     \end__enumext_mini_page
4394     \__enumext_mini_apply_end:
4395   }
4396   {
4397     \__enumext_multicols_stop:
4398   }

```

Now the functions `\__enumext_after_stop_list:` used by the key `after`, `\__enumext_check_ans_key_hook:` used by the key `check-ans`, and `\__enumext_vspace_below:` used by the keys `below` and `below*` are executed. Finally it sets `\__enumext_standard_bool` to false and calls the function `\__enumext_standard_save_counter:` used by the `series`, `resume` and `resume*` keys.

```

4399   \__enumext_after_stop_list:
4400   \__enumext_check_ans_key_hook:
4401   \__enumext_vspace_below:
4402   \bool_set_false:N \__enumext_standard_bool
4403   \bool_if:NF \__enumext_print_keyans_cmd_bool
4404   {
4405     \__enumext_standard_save_counter:
4406   }
4407 }

```

(End of definition for `\__enumext_second_part:`.)

`\__enumext_set_item_width:`

The function `\__enumext_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key for each level of the environment.

```

4408 \cs_new_protected:Nn \__enumext_set_item_width:
4409 {
4410   \dim_set:Nn \itemwidth { \linewidth }
4411   \dim_compare:nT
4412   {
4413     \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
4414   }
4415   {
4416     \dim_sub:Nn \itemwidth
4417     {
4418       \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
4419     }
4420   }
4421 }

```

(End of definition for `\__enumext_set_item_width:`.)

enumext

Now create the `enumext` environment based on `list` environment by levels. It accepts an optional [`<key = val>`] argument in ‘#1’. It first checks where the environment is running (§13.43.3) and processes the `<keys>`. Then it applies the vertical space and the code of the keys `above` and `before*`, activates the “storing structure” and opens the list with the label of the current level. Finally it sets `\itemwidth` (§13.43.3) and executes the code set by the key `first`.

```

4422 \NewDocumentEnvironment{enumext}{0}{}
4423 {
4424   \__enumext_safe_exec:
4425   \__enumext_parse_keys:n {#1}
4426   \__enumext_before_list:
4427   \__enumext_start_store_level:
4428   \__enumext_start_list:nn
4429   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }

```

```

4430     {
4431         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
4432         \__enumext_before_keys_exec:
4433     }
4434     \__enumext_set_item_width:
4435     \__enumext_after_args_exec:
4436 }
4437 {
4438     \__enumext_second_part:
4439 }

```

(End of definition for enumext. This function is documented on page 5.)

Since the `check-ans` check must not be executed by levels but on the complete list, it is taken out of the `enumext` environment using the “hook” function `\__enumext_after_env:nn`.

```

4440 \__enumext_after_env:nn {enumext}
4441 {
4442     \__enumext_execute_after_env:
4443 }

```

13.44 The environment keyans

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

13.44.1 Definition of error messages for keyans

`keyans-nested` Message used by `keyans` environment.

```

4444 \msg_new:nnnn { enumext } { keyans-nested }
4445 {
4446     The~environment~'keyans'~can't~be~nested~\msg_line_context:.
4447 }
4448 {
4449     Close~the~first~'keyans'~environment~before~starting~
4450     another~one.
4451 }

```

(End of definition for keyans-nested.)

13.44.2 Internal functions for keyans

`\__enumext_keyans_safe_exec:` The `keyans` environment will only be available if the `save-ans` key is active and can only be used at the “first level” within the `enumext` environment. The environment must not be nested, so a maximum is set at this point. If the conditions are not met, an error message is returned.

```

4452 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
4453 {
4454     \bool_lazy_or:nnF { \__enumext_store_active_bool } { \__enumext_in_keyans_p: }
4455     {
4456         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
4457     }
4458     \int_incr:N \__enumext_keyans_level_int
4459     \bool_set_true:N \__enumext_keyans_env_bool
4460     \__enumext_keyans_name_and_start:
4461     % Set false for interfering with enumext nested in keyans
4462     % (yes, its possible and crayze)
4463     \bool_set_false:N \__enumext_store_active_bool
4464     \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
4465     {
4466         \msg_error:nn { enumext } { keyans-nested }
4467     }
4468     \int_compare:nNnT { \__enumext_level_int } > { 1 }
4469     {
4470         \msg_error:nn { enumext } { keyans-wrong-level }
4471     }
4472 }

```

(End of definition for __enumext_keyans_safe_exec:.)

`\__enumext_keyans_parse_keys:n` Parse [`<key = val>`] for `keyans` environment.

```

4473 \cs_new_protected:Nn \__enumext_keyans_parse_keys:n

```

```

4474 {
4475   \keys_set:nn { enumext / keyans } {#1}
4476 }

```

(End of definition for __enumext_keyans_parse_keys:n.)

```

\__enumext_before_list_v:
\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_second_part_v:

```

Same implementation as the one used in the enumext environment.

```

4477 \cs_new_protected:Nn \__enumext_before_list_v:
4478 {
4479   \__enumext_vspace_above_v:
4480   \__enumext_before_args_exec_v:
4481   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
4482   {
4483     \dim_set:Nn \l__enumext_minipage_left_v_dim
4484     {
4485       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
4486     }
4487     \bool_set_true:N \l__enumext_minipage_active_v_bool
4488     \int_gincr:N \g__enumext_minipage_stat_int
4489     \__enumext_keyans_minipage_add_space:
4490     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
4491   }
4492   \__enumext_keyans_multicols_start:
4493 }
4494 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
4495 {
4496   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
4497   {
4498     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
4499     {
4500       \dim_set:Nn \l__enumext_columns_sep_v_dim
4501       {
4502         (
4503           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
4504         ) / \l__enumext_columns_v_int
4505         - \l__enumext_listoffset_v_dim
4506       }
4507     }
4508     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
4509     \dim_zero:N \columnseprule % no rule here
4510     \bool_if:NF \l__enumext_minipage_active_v_bool
4511     {
4512       \skip_zero:N \multicolsep
4513       \__enumext_keyans_multi_addvspace:
4514     }
4515     \raggedcolumns
4516     \begin{multicols}{\l__enumext_columns_v_int }
4517   }
4518 }
4519 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
4520 {
4521   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
4522   {
4523     \__enumext_stop_list:
4524     \end{multicols}
4525     \__enumext_unskip_unkern:
4526     \__enumext_unskip_unkern:
4527     \par
4528     \bool_if:NTF \l__enumext_minipage_active_v_bool
4529     {
4530       \__enumext_mini_capture_multicols:
4531     }
4532     {
4533       \addvspace{ \l__enumext_multicols_below_v_skip }
4534     }
4535   }
4536   {
4537     \__enumext_stop_list:
4538   }
4539 }
4540 \cs_new_protected:Nn \__enumext_second_part_v:

```

```

4541 {
4542   \bool_if:NTF \l__enumext_minipage_active_v_bool
4543   {
4544     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4545     {
4546       \msg_warning:nn { enumext } { missing-miniright }
4547       \miniright
4548     }
4549     \int_gzero:N \g__enumext_minipage_stat_int
4550     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4551     \end__enumext_mini_page
4552     \__enumext_mini_apply_end:
4553   }
4554   {
4555     \__enumext_keyans_multicols_stop:
4556   }
4557   \bool_set_false:N \l__enumext_keyans_env_bool
4558   \__enumext_after_stop_list_v:
4559   \__enumext_vspace_below_v:
4560 }

```

(End of definition for __enumext_before_list_v: and others.)

__enumext_keyans_set_item_width:

The function `\__enumext_keyans_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key.

```

4561 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4562 {
4563   \dim_set:Nn \itemwidth { \linewidth }
4564   \dim_compare:nT
4565   {
4566     \l__enumext_listoffset_v_dim != \c_zero_dim
4567   }
4568   {
4569     \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4570   }
4571 }

```

(End of definition for __enumext_keyans_set_item_width:.)

keyans

Defines the environment `keyans`, also based on lists. It accepts an optional `[\langle key = val \rangle]` argument in ‘#1’. It first checks that the environment is available and where it is running, and processes the `\langle keys \rangle`. Then it opens the list, sets `\itemwidth` (§13.44.2) and executes the code set by the key `first`. In the second part it checks the use of `\item*` and finishes the environment.

```

4572 \NewDocumentEnvironment{keyans}{0}{ }
4573 {
4574   \__enumext_keyans_safe_exec:
4575   \__enumext_keyans_parse_keys:n {#1}
4576   \__enumext_before_list_v:
4577   \__enumext_start_list:nn
4578   { \tl_use:N \l__enumext_label_v_tl }
4579   {
4580     \__enumext_list_arg_two_v:
4581     \__enumext_before_keys_exec_v:
4582   }
4583   \__enumext_keyans_set_item_width:
4584   \__enumext_after_args_exec_v:
4585 }
4586 {
4587   \__enumext_check_starred_cmd:n { item }
4588   \__enumext_second_part_v:
4589 }

```

(End of definition for `keyans`. This function is documented on page 17.)

13.45 Tagging PDF support for non-standard list environments

The $\TeX$ release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the visual result

in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a `\list` environment is a `\list` environment, no matter what it looks like in the PDF file.

To maintain a correct `\list` structure when `\DocumentMetadata` is active, it is necessary to do some things manually using `tagpdf`[18] and `ltsockets`[19]. This implementation is an adaptation of the answer given by the author of this package thanks to Ulrike Fischer's comments in [How can I modify my \item redefinition to be compatible with tagging-pdf](#).

13.45.1 Socket for tagging support in enumext* and keyans*

First, the necessary sockets and their behavior are defined for `enumext*` and `keyans*`.

```

start-list-tags
stop-start-tags
stop-list-tags
stop-list-tags-par
__enumext_start_list_tag:n
  __enumext_stop_start_list_tag:
__enumext_stop_list_tag:n
  __enumext_stop_item_break_tag:n
  __enumext_stop_start_break_tag:
stop-start-tags-break
stop-list-tags-break
4590 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4591 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {start-list-tags}
4592 {
4593   \tag_resume:n {#1}
4594   \tag_struct_begin:n {tag=LI}
4595   \tag_struct_begin:n {tag=Lbl}
4596   \tag_mc_begin:n {tag=Lbl}
4597 }
4598 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4599 {
4600   \tag_mc_end:
4601   \tag_struct_end:n {tag=Lbl}
4602   \tag_struct_begin:n {tag=LBody}
4603   % {tag=text-unit} now "semantic-para"
4604   % (https://github.com/latex3/latex2e/commit/83f054afc042e82a664085b6f2fd7c8d61705fb9)
4605   \tag_struct_begin:n {tag=semantic-para}
4606   % {tag=text} now "text-block"
4607   \tag_struct_begin:n {tag=text-block}
4608 }
4609 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4610 {
4611   \tag_struct_end:n {tag=text-block}
4612   \tag_struct_end:n {tag=semantic-para} % {tag=text-unit}
4613   \tag_struct_end:n {tag=LBody}
4614   \tag_struct_end:n {tag=LI}
4615   \tag_suspend:n {#1}
4616 }
4617 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags-par}
4618 {
4619   \tag_struct_end:n {tag=LBody}
4620   \tag_struct_end:n {tag=LI}
4621   \tag_suspend:n {#1}
4622 }
4623 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-start-tags-break}
4624 {
4625   \tag_mc_end:
4626   \tag_struct_end:n {tag=Lbl}
4627   \tag_struct_begin:n {tag=LBody}
4628   \tag_socket_use:nn {para/semantic/begin} { }
4629   \tag_socket_use:nn {para/textblock/begin} { }
4630 }
4631 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags-break}
4632 {
4633   \tag_socket_use:n {para/textblock/end}
4634   \tag_socket_use:nn {para/semantic/end} { }
4635   \tag_struct_end:n {tag=LBody}
4636   \tag_struct_end:n {tag=LI}
4637   \tag_suspend:n {#1}
4638 }

```

And now they are wrapped so that they are only active when `\DocumentMetadata` is present.

```

4639 \cs_new_protected_nopar:Nn __enumext_start_list_tag:n
4640 {
4641   \IfDocumentMetadataT
4642   {
4643     \socket_assign_plugin:nn {tagsupport/__enumext/starred} {start-list-tags}
4644     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4645   }
4646 }
4647 \cs_new_protected_nopar:Nn __enumext_stop_start_list_tag:
4648 {

```

```

4649 \IfDocumentMetadataT
4650 {
4651   \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4652   \socket_use:nn {tagsupport/__enumext/starred} { }
4653 }
4654 }
4655 \cs_new_protected_nopar:Nn \__enumext_stop_start_break_tag:
4656 {
4657   \IfDocumentMetadataT
4658   {
4659     \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags-break}
4660     \socket_use:nn {tagsupport/__enumext/starred} { }
4661   }
4662 }
4663 \cs_new_protected_nopar:Nn \__enumext_stop_list_tag:n
4664 {
4665   \IfDocumentMetadataT
4666   {
4667     \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4668     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4669   }
4670 }

```

The items of `item-break` have no `minipage`, so the paragraphs of the `<content>` are tagged by the *paragraph hooks*. The first paragraph of the item is opened while the tagging is still suspended, because the label is placed before the structures of the `<content>` are started, so its *begin* hook is never executed. For this reason, the plug `stop-start-tags-break` (used by `\__enumext_stop_start_break_tag:`) opens the structures of that paragraph using the sockets `para/semantic/begin` and `para/textblock/begin` of the $\TeX$ kernel. These sockets also take care of the counters of the *paragraph hooks*, the names of the tags and the class of the Layout attribute `TextAlign`. If the last paragraph of the item has already been ended by a `\par`, coming from an empty line before the next `\item` or before `\end` or from a `\par` inside a group, the hook has closed the text-block and semantic-para structures and the marked content, and we are in *vertical mode*. Then only `LBody` and `LI` remain to be closed, using the plug `stop-list-tags-par`. When the item ends with a standard list (`itemize`, for example) we are also in *vertical mode*, but the semantic-para that contains the list is still open, because the kernel closes it with the `\par` that ends the paragraph *continued* after the list. For this reason a `\par` is executed first in vertical mode: it closes that structure, and if there is nothing pending it does nothing. Otherwise the paragraph is still open and the plug `stop-list-tags-break` closes it with the sockets `para/textblock/end` and `para/semantic/end`, before the `\par` that ends the item. That `\par` runs with the tagging suspended and is not counted.

```

4671 \cs_new_protected_nopar:Nn \__enumext_stop_item_break_tag:n
4672 {
4673   \IfDocumentMetadataT
4674   {
4675     \mode_if_vertical:TF
4676     {
4677       \par
4678       \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags-par}
4679     }
4680     {
4681       \socket_assign_plug:nn
4682         {tagsupport/__enumext/starred} {stop-list-tags-break}
4683     }
4684     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4685   }
4686 }

```

(End of definition for *start-list-tags* and others.)

13.45.2 Socket for tagging support in `keyanspic`

`start-list-tags` First, the necessary sockets and their behavior are defined for the `keyanspic` environment.

```

stop-start-tags
stop-list-tags
\__enumext_anspic_start_list_tag:
\__enumext_anspic_stop_start_list_tag:
\__enumext_anspic_stop_list_tag:
4687 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4688 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4689 {
4690   \tag_resume:n {keyanspic}
4691   \tag_struct_begin:n {tag=LI}
4692   \tag_struct_begin:n {tag=Lbl}
4693   \tag_mc_begin:n {tag=Lbl}
4694 }
4695 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}

```

```
4696 {
4697     \tag_mc_end:
4698     \tag_struct_end:n {tag=Lbl}
4699     \tag_struct_begin:n {tag=LBody}
4700     \tag_struct_begin:n {tag=semantic-para}
4701     \tag_struct_begin:n {tag=text-block}
4702     \tag_mc_begin:n {tag=text-block}
4703 }
4704 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4705 {
4706     \tag_mc_end:
4707     \tag_struct_end:n {tag=text-block}
4708     \tag_struct_end:n {tag=semantic-para}
4709     \tag_struct_end:n {tag=LBody}
4710     \tag_struct_end:n {tag=LI}
4711     \tag_suspend:n {keyanspic}
4712 }
```

And now they are wrapped so that they are only active when `\DocumentMetadata` is present.

```
4713 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4714 {
4715     \IfDocumentMetadataT
4716     {
4717         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4718         \socket_use:n {tagsupport/__enumext/keyanspic}
4719     }
4720 }
4721 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4722 {
4723     \IfDocumentMetadataT
4724     {
4725         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4726         \socket_use:n {tagsupport/__enumext/keyanspic}
4727     }
4728 }
4729 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4730 {
4731     \IfDocumentMetadataT
4732     {
4733         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4734         \socket_use:n {tagsupport/__enumext/keyanspic}
4735     }
4736 }
```

(End of definition for `start-list-tags` and others.)

13.46 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a `list` based environment that uses the same configuration for “*spacing*” and $\langle label \rangle$ as the `keyans` environment, but it does not use `\item`. The $\langle contents \rangle$ are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the $\langle label \rangle$ centered “*above*” or “*below*”, adjusting *widths* and *position* according to the options passed to the environment.

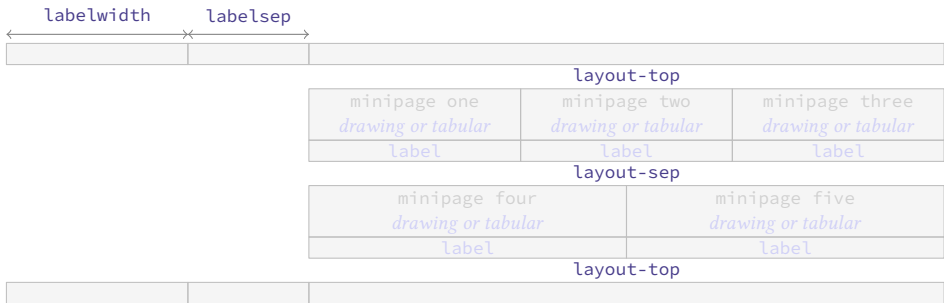


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, some variables must be set and exported in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a `\macro`?](#)

13.46.1 Definition of variables for keyanspic and \anspic

Internal variables used by `keyanspic` environment and `\anspic` command (§13.46.3).

```
\l__enumext_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_anspic_label_above_bool
\l__enumext_anspic_mini_pos_str
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim
```

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

13.46.2 Definition of error messages for keyanspic

Messages used by `keyanspic` environment.

```
keyanspic-nested
keyanspic-wrong-level
keyanspic-item-cmd
```

```
4747 \msg_new:nnnn { enumext } { keyanspic-nested }
4748 {
4749   The~environment~'keyanspic'~can't~be~nested~\msg_line_context:.
4750 }
4751 {
4752   Close~the~first~'keyanspic'~environment~before~starting~
4753   another~one.
4754 }
4755 \msg_new:nnnn { enumext } { keyanspic-wrong-level }
4756 {
4757   Wrong~level~position~for~'keyanspic'~\msg_line_context:.
4758 }
4759 {
4760   The~environment~'keyanspic'~can~only~be~in~the~first~level.
4761 }
4762 \msg_new:nnnn { enumext } { keyanspic-item-cmd }
4763 {
4764   Can't~use~'\token_to_str:N \item'~in~'keyanspic'~\msg_line_context:.
4765 }
4766 {
4767   Remove~'\token_to_str:N \item'~from~the~'keyanspic'~environment.
4768 }
```

(End of definition for `keyanspic-nested`, `keyanspic-wrong-level`, and `keyanspic-item-cmd`.)

13.46.3 The environment keyanspic

`label-pos` First the keys are defined. The position of the *(label)* centered “above” or “below” is handled by `label-pos`, and its vertical separation from the *drawing or tabular* by the key `label-sep`. The “*layout style*” is handled by the key `layout-sty`, which takes two values separated by comma $\{\langle n^{\circ} upper, n^{\circ} lower \rangle\}$ and determines the number of `minipage` environments in which all arguments of `\anspic` are printed at the “upper” and “lower” within the environments separated by the value of the key `layout-sep`. The vertical space “top” and “bottom” of the environment is handled by the key `layout-top`.

```
mark-ans
mark-pos
mark-sep
save-sep
wrap-opt
wrap-ans*
show-ans
show-pos
unknown
```

```
4769 \keys_define:nn { enumext / keyanspic }
4770 {
4771   label-pos .choices:nn =
4772     { above , below }
4773   {
4774     \int_case:nn { \l_keys_choice_int }
4775     {
4776       {1}
4777       {
4778         \bool_set_true:N \l__enumext_anspic_label_above_bool
4779         \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
4780       }
4781       {2}
4782       {
4783         \bool_set_false:N \l__enumext_anspic_label_above_bool
4784         \str_set:Nn \l__enumext_anspic_mini_pos_str { b }
4785       }
4786     }
4787   },
```

```

4788     label-pos  .initial:n = below,
4789     label-pos  .value_required:n = true,
4790     label-pos / unknown .code:n =
4791       \msg_error:nneee { enumext } { unknown-choice }
4792       { label-pos } { above,~ below } { \exp_not:n {#1} },
4793     label-sep  .skip_set:N = \l__enumext_anspic_label_sep_skip,
4794     label-sep  .value_required:n = true,
4795     layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4796     layout-sty .value_required:n = true,
4797     layout-sep .meta:nn = { enumext / keyans } { parsep = {#1} },
4798     layout-sep .value_required:n = true,
4799     layout-top .meta:nn = { enumext / keyans } { topsep = {#1} },
4800     layout-top .value_required:n = true,
4801     mark-ans   .meta:nn = { enumext / keyans } { mark-ans = {#1} },
4802     mark-ans   .value_required:n = true,
4803     mark-pos   .meta:nn = { enumext / keyans } { mark-pos = {#1} },
4804     mark-pos   .value_required:n = true,
4805     mark-sep   .meta:nn = { enumext / keyans } { mark-sep = {#1} },
4806     mark-sep   .value_required:n = true,
4807     save-sep   .meta:nn = { enumext / keyans } { save-sep = {#1} },
4808     save-sep   .value_required:n = true,
4809     wrap-opt   .meta:nn = { enumext / keyans } { wrap-opt = {#1} },
4810     wrap-opt   .value_required:n = true,
4811     wrap-ans*  .meta:nn = { enumext / keyans } { wrap-ans* = {#1} },
4812     wrap-ans*  .value_required:n = true,
4813     show-ans   .meta:nn = { enumext / keyans } { show-ans = {#1} },
4814     show-ans   .value_required:n = true,
4815     show-pos   .meta:nn = { enumext / keyans } { show-pos = {#1} },
4816     show-pos   .value_required:n = true,
4817     unknown    .code:n =
4818       {
4819         \tl_set:Nx \l__enumext_envir_name_tl { keyanspic }
4820         \__enumext_keyans_unknown_keys:n {#1}
4821       },
4822   }

```

(End of definition for `label-pos` and others.)

```

\__enumext_keyans_pic_safe_exec:
\__enumext_keyans_pic_parse_keys:n
\__enumext_keyans_pic_skip_abs:N
\__enumext_keyans_pic_arg_two:

```

The function `\__enumext_keyans_pic_safe_exec:` check the nested level position inside the `enumext` environment.

```

4823 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4824 {
4825   \int_incr:N \l__enumext_keyans_pic_level_int
4826   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4827   {
4828     \msg_error:nn { enumext } { keyanspic-nested }
4829   }
4830   \__enumext_keyans_name_and_start:
4831 }

```

Parse [`key = val`] for `keyanspic` environment.

```

4832 \cs_new_protected:Nn \__enumext_keyans_pic_parse_keys:n
4833 {
4834   \tl_if_novalue:nF {#1}
4835   {
4836     \keys_set:nn { enumext / keyanspic } {#1}
4837   }
4838 }

```

The function `\__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```

4839 \cs_new_protected:Nn \__enumext_keyans_pic_skip_abs:N
4840 {
4841   \dim_compare:nNnT {#1} < { \c_zero_dim }
4842   {
4843     \skip_set:Nn #1 { -#1 }
4844   }
4845 }

```

The `\__enumext_keyans_pic_arg_two:` function is used in the *second argument* of the `list` environment that defines the `keyanspic` environment, taking the configuration of the “spaces” and the keys `label`, `wrap-label`, `parsep` and `topsep` from the `keyans` environment. The first step is to set the boolean variable

`\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to `false`, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```

4846 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
4847 {
4848   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4849   \__enumext_list_arg_two_v:
4850   \__enumext_keyans_pic_skip_abs:N \parsep

```

Now the counter `enumXv` of the `keyans` environment is incremented and the *total height* of the `<label>` is saved in `\l__enumext_anspic_label_htdp_dim` used by `\anspic`, and the values of `\parsep` are adjusted only if the key `label-pos` is set to *below*.

```

4851   \bool_if:NF \l__enumext_anspic_label_above_bool
4852   {
4853     \stepcounter { enumXv }
4854     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4855     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4856     {
4857       \box_ht_plus_dp:N \l__enumext_anspic_label_box
4858     }
4859     \skip_add:Nn \parsep
4860     {
4861       \l__enumext_anspic_label_htdp_dim
4862       + \box_dp:N \strutbox
4863       + \l__enumext_anspic_label_sep_skip
4864     }
4865   }

```

Finally the value of `\leftmargin` and `\topsep` is *adjusted*, then `\listparindent`, `\partopsep` and `\itemsep` are set to zero so that the *horizontal* and *vertical* space is not affected.

```

4866   \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4867   \ignorespaces
4868   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4869   \dim_zero:N \listparindent
4870   \skip_zero:N \partopsep
4871   \skip_zero:N \itemsep
4872 }

```

(End of definition for `\__enumext_keyans_pic_safe_exec`: and others.)

keyanspic Defines the environment `keyanspic`. It accepts an optional [`<key = val>`] argument in ‘#1’ that is processed after checking the nested level. For compatibility with *tagged* PDF the `\begin{list}` form must be used, along with a lot of conditional code using `\IfDocumentMetadataTF`. First, the code for automatic *tagged* PDF for `list` environments is stopped, `\item` is redefined so that it cannot be used, and the code for automatic *tagged* PDF is stopped for the `keyanspic` environment.

```

4873 \NewDocumentEnvironment{keyanspic}{ o }
4874 {
4875   \__enumext_keyans_pic_safe_exec:
4876   \__enumext_keyans_pic_parse_keys:n {#1}
4877   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4878   \IfDocumentMetadataT
4879   {
4880     \tag_suspend:n {list}
4881   }
4882   \item[] \scan_stop:
4883   \RenewDocumentCommand \item {}
4884   {
4885     \msg_error:nn { enumext } { keyanspic-item-cmd }
4886   }
4887   \IfDocumentMetadataT
4888   {
4889     \tag_resume:n {keyanspic}
4890     \tag_socket_use:n { para/off }
4891     \tag_suspend:n {keyanspic}
4892   }
4893 }

```

In the second part of the environment definition, the code for *tagged* PDF is placed manually and the command is executed: `\anspic` is run using the `\__enumext_anspic_exec:` function.

```

4894 {

```

```

4895 \IfDocumentMetadataT
4896 {
4897   \tag_resume:n {keyanspic}
4898 }
4899 \__enumext_anspic_exec:
4900 \IfDocumentMetadataT
4901 {
4902   \tag_suspend:n {keyanspic}
4903 }
4904 \end{list}
4905 \IfDocumentMetadataT
4906 {
4907   \tag_struct_end:n {tag=L}
4908 }

```

Finally it checks if `\anspic*` has been used, sets the counter `enumXvi` to zero and applies the “adjusted” vertical space at the bottom.

```

4909 \__enumext_check_starred_cmd:n { anspic }
4910 \setcounter { enumXvi } { 0 }
4911 \bool_if:NTF \l__enumext_anspic_label_above_bool
4912 {
4913   \par\addvspace{ 0.5\box_dp:N \strutbox }
4914 }
4915 {
4916   \par
4917   \addvspace
4918   {
4919     \dim_eval:n
4920     {
4921       \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4922       + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4923     }
4924   }
4925 }
4926 }

```

(End of definition for `keyanspic`. This function is documented on page 19.)

13.46.4 The command `\anspic`

The `\anspic` command takes three arguments, the *starred versions* `\anspic*` [`<content>`] store the current `<label>` next to the *optional argument* [`<content>`] in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* [`<drawing or tabular>`] is NOT stored in the *sequence* or *prop list*.

- One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of `<label>`, the `\anspic` command processes the arguments in order, where ‘#1’ and ‘#2’ correspond to `<label>` and ‘#3’ to the mandatory argument and puts all this inside a `minipage` environment. If ‘#1’ and ‘#2’, that is `<label>`, is above ‘#3’ there are no problems with *tagged* PDF, but if ‘#3’ comes first the list created with *tagged* PDF will not be correct.

The `\__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4927 \cs_new_protected:Nn \__enumext_anspic_body_dim:n
4928 {
4929   \bool_if:NF \l__enumext_anspic_label_above_bool
4930   {
4931     \IfDocumentMetadataT
4932     {
4933       \tag_suspend:n {keyanspic}
4934     }
4935     \vbox_set:Nn \l__enumext_anspic_body_box {#1}
4936     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4937     {
4938       \box_ht_plus_dp:N \l__enumext_anspic_body_box
4939     }
4940     \IfDocumentMetadataT
4941     {
4942       \tag_resume:n {keyanspic}
4943     }
4944   }
4945 }

```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* ‘*’ and *optional argument* passed to the command. Here the `\label` and *optional argument* are stored in *prop list* and *sequence* and the `show-ans`, `show-pos`, `font`, `wrap-label`, `wrap-ans*` and `wrap-opt` keys are executed.

```

4946 \cs_new_protected:Nn \__enumext_anspic_label:nn
4947 {
4948   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4949   {
4950     \bool_if:nTF {#1}
4951     {
4952       \bool_set_true:N \l__enumext_item_wrap_key_bool
4953       \bool_set_true:N \l__enumext_wrap_label_v_bool
4954       \__enumext_keyans_save_item_opt:n {#2}
4955       \__enumext_keyans_addto_prop:n {#2}
4956       \__enumext_keyans_store_ref:
4957       \__enumext_keyans_addto_seq:n {#2}
4958       \int_gincr:N \g__enumext_check_starred_cmd_int
4959       \__enumext_keyans_show_ans:
4960       \__enumext_keyans_show_pos:
4961       \makebox[ \l__enumext_labelwidth_v_dim ][c]
4962       {
4963         \tl_use:N \l__enumext_label_font_style_v_tl
4964         \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4965       }
4966       \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4967       \__enumext_keyans_show_item_opt:
4968     }
4969     {
4970       \bool_set_false:N \l__enumext_item_wrap_key_bool
4971       \tl_use:N \l__enumext_label_font_style_v_tl
4972       \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4973     }
4974   }
4975 }

```

The function `\__enumext_anspic_label_pos:nnn` will be in charge of handling the “*counter*” and the position of the `\label`, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```

4976 \cs_new_protected:Nn \__enumext_anspic_label_pos:nnn
4977 {
4978   \stepcounter { enumXvi }
4979   \__enumext_anspic_body_dim:n {#3}
4980   \bool_if:NTF \l__enumext_anspic_label_above_bool
4981   {
4982     \__enumext_anspic_label:nn {#1} {#2}
4983   }
4984   {
4985     \raisebox
4986     {
4987       -\dim_eval:n
4988       {
4989         \l__enumext_anspic_label_htdp_dim
4990         + \l__enumext_anspic_body_htdp_dim
4991         + \box_dp:N \strutbox
4992         + \l__enumext_anspic_label_sep_skip
4993       }
4994     }
4995     [ opt ] [ opt ]
4996     {
4997       \__enumext_anspic_label:nn {#1} {#2}
4998     }
4999   }
5000 }
5001 %

```

The `\__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `\__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```

5002 \cs_new_protected:Nn \__enumext_anspic_args:nnn
5003 {
5004   \__enumext_anspic_start_list_tag:
5005   \__enumext_anspic_label_pos:nnn {#1} {#2} {#3}

```

```

5006     \__enumext_anspic_stop_start_list_tag:
5007     \bool_if:NTF \l__enumext_anspic_label_above_bool
5008     {
5009         \[\l__enumext_anspic_label_sep_skip] #3
5010     }
5011     {
5012         \[ #3
5013     }
5014     \__enumext_anspic_stop_list_tag:
5015 }

```

The value $\{\langle n^{\circ upper}, n^{\circ lower} \rangle\}$ passed to the `layout-sty` key is split by comma and is handled directly by the function `\__enumext_anspic_print:n` and passed to the function `\__enumext_anspic_row:n`.

```

5016 \cs_new_protected:Nn \__enumext_anspic_print:n
5017 {
5018     \clist_map_function:nN {#1} \__enumext_anspic_row:n
5019 }
5020 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }

```

The function `\__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic` saved in the `\l__enumext_anspic_args_seq` sequence inside them.

```

5021 \cs_new_protected:Nn \__enumext_anspic_row:n
5022 {
5023     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
5024     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
5025     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
5026     \int_step_inline:nnn
5027     { \l__enumext_anspic_above_int + 1 }
5028     { \l__enumext_anspic_below_int }
5029     {
5030         \IfDocumentMetadataT
5031         {
5032             \tag_suspend:n {minipage}
5033         }
5034         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{
5035             \l__enumext_anspic_mini_width_dim }
5036             \centering
5037             \seq_item:Nn \l__enumext_anspic_args_seq {##1}
5038         \end{minipage}
5039         \IfDocumentMetadataT
5040         {
5041             \tag_resume:n {minipage}
5042         }
5043     }
5044     \par
5045 }

```

The `\__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the `keyanspic` environment definition. If the key `layout-sty` is not set, everything will be printed on a *single line*.

```

5046 \cs_new_protected:Nn \__enumext_anspic_exec:
5047 {
5048     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
5049     {
5050         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
5051     }
5052     {
5053         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
5054     }
5055 }

```

(End of definition for `\__enumext_anspic_body_dim:n` and others.)

\anspic The command accepts a *starred argument* ‘`*`’ in ‘`#1`’, an optional $\langle content \rangle$ in ‘`#2`’ and a mandatory $\langle drawing\ or\ tabular \rangle$ in ‘`#3`’. The command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise an error is returned. The three arguments are handled by the function `\__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

5056 \NewDocumentCommand \anspic { s o +m }
5057 {

```

```

5058 \bool_lazy_or:nnF { \__enumext_store_active_bool } { \__enumext_in_keyans_p: }
5059 {
5060   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
5061 }
5062 \int_compare:nNnT { \__enumext_level_int } > { 1 }
5063 {
5064   \msg_error:nn { enumext } { keyanspic-wrong-level }
5065 }
5066 \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
5067 {
5068   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
5069 }
5070 \seq_put_right:Nn \__enumext_anspic_args_seq
5071 {
5072   \__enumext_anspic_args:nnn {#1} {#2} {#3}
5073 }
5074 }

```

(End of definition for `\anspic`. This function is documented on page 19.)

13.47 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard $\TeX$ list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`[20]. It is not possible to redefine `\item` and `\makeLabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments*, the `\item` command and its $\langle content \rangle$ are captured in a *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`; the *optional argument* ($\langle number \rangle$) is also added to `\item` to be able to *join columns* horizontally. In simple terms, the goal is for `\item` to behave in the same way as in the `enumext` environment but with an additional *first optional argument* ($\langle number \rangle$).

Since `\item` is not redefined with `\RenewDocumentCommand`, the original definition of the kernel is replaced by the one of the horizontal environment. The original is not lost, a copy is made at the beginning of the document and `\item` is turned into a *switch* that decides which of the two definitions must be run, as explained in §13.48.2. Because of this, the *standard list environments* and any environment defined using base `list` or `\trivlist` can be used in the $\langle content \rangle$ of the items of `enumext*` and `keyans*`: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `list`, `center`, `flushleft`, `flushright`, `trivlist`, `tabbing`, `verbatim` and all environments created with `\newtheorem`.

- ♥ When `item-break` is active and the item occupies the whole row, the content of the item is not captured in a *horizontal box* or in a `minipage`, but it is typeset directly in the vertical list (§13.48.4).
- ♥ For compatibility with the *tagged PDF*, the environments are closed according to the presence or not of the `mini-env` key.

13.47.1 Definition of variables

```

\__enumext_joined_item_X_int
\__enumext_joined_item_aux_X_int
\__enumext_tmpa_X_int
\__enumext_item_width_X_dim
\g__enumext_minipage_active_X_bool
\__enumext_miniright_code_X_box

```

Internal variables used by `enumext*` and `keyans*` environments. The integers `\__enumext_joined_item_X_int`, `\__enumext_joined_item_aux_X_int` and `\__enumext_tmpa_X_int` and the dimension `\__enumext_item_width_X_dim` are used by the functions that calculate the item box width and join the item columns. The variables `\g__enumext_minipage_active_X_bool` and `\__enumext_miniright_code_X_box` are used by the `mini-env`, `mini-right` and `mini-right*` keys (§13.47.5).

```

5075 \cs_set_protected:Nn \__enumext_tmp:n
5076 {
5077   \int_new:c { \__enumext_joined_item_ #1 _int }
5078   \int_new:c { \__enumext_joined_item_aux_ #1 _int }
5079   \int_new:c { \__enumext_tmpa_ #1 _int }
5080   \dim_new:c { \__enumext_item_width_ #1 _dim }
5081   \bool_new:c { g__enumext_minipage_active_ #1 _bool }
5082   \box_new:c { \__enumext_miniright_code_ #1 _box }
5083 }
5084 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

13.47.2 Definition of error messages

```

nested-horizontal
item-joined
item-joined-columns

```

Messages used by `enumext*` and `keyans*` environments.

```

5085 \msg_new:nnnn { enumext } { nested-horizontal }
5086 {

```

```

5087   The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~
5088   \msg_line_context:.
5089 }
5090 {
5091   'enumext*'~and~'keyans*'~are~horizontal~lists~and~can't~be~
5092   inside~each~other.
5093 }
5094 \msg_new:nnn { enumext } { item-joined }
5095 {
5096   Items~joined~(#1)~>~#2~columns~\msg_line_context:.
5097 }
5098 \msg_new:nnn { enumext } { item-joined-columns }
5099 {
5100   Not~enough~space~to~join~items~(#1)~>~#2~\msg_line_context:.
5101 }

```

(End of definition for `nested-horizontal`, `item-joined`, and `item-joined-columns`.)

13.47.3 Functions for item box width

`\__enumext_starred_columns_set_vii:`
`\__enumext_starred_columns_set_viii:`

Sets the default value for the *width of the box* containing the *content* of the items for `enumext*` environment.

```

5102 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
5103 {
5104   \dim_compare:nNtT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
5105   {
5106     \dim_set:Nn \l__enumext_columns_sep_vii_dim
5107     {
5108       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
5109       / \l__enumext_columns_vii_int
5110     }
5111   }
5112   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
5113   \dim_set:Nn \l__enumext_item_width_vii_dim
5114   {
5115     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
5116     / \l__enumext_columns_vii_int
5117     - \l__enumext_labelwidth_vii_dim
5118     - \l__enumext_labelsep_vii_dim
5119   }

```

When the key `rightmargin` is active the values must be adjusted.

```

5120   \dim_compare:nNtT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
5121   {
5122     \dim_sub:Nn \l__enumext_item_width_vii_dim
5123     {
5124       ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
5125       / \l__enumext_columns_vii_int
5126     }
5127     \dim_add:Nn \l__enumext_columns_sep_vii_dim
5128     {
5129       \l__enumext_rightmargin_vii_dim
5130     }
5131   }
5132 }

```

Same implementation for the `keyans*` environment.

```

5133 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
5134 {
5135   \dim_compare:nNtT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
5136   {
5137     \dim_set:Nn \l__enumext_columns_sep_viii_dim
5138     {
5139       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
5140       / \l__enumext_columns_viii_int
5141     }
5142   }
5143   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
5144   \dim_set:Nn \l__enumext_item_width_viii_dim
5145   {
5146     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
5147     / \l__enumext_columns_viii_int

```

```

5148     - \l__enumext_labelwidth_viii_dim
5149     - \l__enumext_labelsep_viii_dim
5150   }
5151   \dim_compare:nNt { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
5152   {
5153     \dim_sub:Nn \l__enumext_item_width_viii_dim
5154     {
5155       ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
5156       / \l__enumext_columns_viii_int
5157     }
5158     \dim_add:Nn \l__enumext_columns_sep_viii_dim
5159     {
5160       \l__enumext_rightmargin_viii_dim
5161     }
5162   }
5163 }

```

(End of definition for `\__enumext_starred_columns_set_vii:` and `\__enumext_starred_columns_set_viii:`)

13.47.4 Functions for join item columns

`\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the *content* passed to `\item(<columns>)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

5164 \cs_new_protected:Nn \__enumext_starred_joined_item_vii:n
5165 {
5166   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
5167   \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
5168   {
5169     \msg_warning:nnee { enumext } { item-joined }
5170     { \int_use:N \l__enumext_joined_item_vii_int }
5171     { \int_use:N \l__enumext_columns_vii_int }
5172     \int_set:Nn \l__enumext_joined_item_vii_int
5173     {
5174       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
5175     }
5176   }
5177   \int_compare:nNt
5178   { \l__enumext_joined_item_vii_int }
5179   >
5180   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
5181   {
5182     \msg_warning:nnee { enumext } { item-joined-columns }
5183     { \int_use:N \l__enumext_joined_item_vii_int }
5184     {
5185       \int_eval:n
5186       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
5187     }
5188     \int_set:Nn \l__enumext_joined_item_vii_int
5189     {
5190       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
5191     }
5192   }
5193   \int_compare:nNtF { \l__enumext_joined_item_vii_int } > { 1 }
5194   {
5195     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
5196     \int_decr:N \l__enumext_joined_item_aux_vii_int
5197     \int_add:Nn \l__enumext_item_column_pos_vii_int
5198     { \l__enumext_joined_item_aux_vii_int }
5199     \int_gadd:Nn \g__enumext_item_count_all_vii_int
5200     { \l__enumext_joined_item_aux_vii_int }
5201     \dim_set:Nn \l__enumext_joined_width_vii_dim
5202     {
5203       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
5204       + (
5205         \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
5206         + \l__enumext_columns_sep_vii_dim
5207       ) * \l__enumext_joined_item_aux_vii_int
5208     }
5209     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
5210   }

```

```

5211     {
5212         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
5213         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
5214     }
5215 }

```

Same implementation for the **keyans*** environment.

```

5216 \cs_new_protected:Nn \__enumext_starred_joined_item_viii:n
5217 {
5218     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
5219     \int_compare:nNnT { \l__enumext_joined_item_viii_int } >
5220     { \l__enumext_columns_viii_int }
5221     {
5222         \msg_warning:nnee { enumext } { item-joined }
5223         { \int_use:N \l__enumext_joined_item_viii_int }
5224         { \int_use:N \l__enumext_columns_viii_int }
5225         \int_set:Nn \l__enumext_joined_item_viii_int
5226         {
5227             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
5228         }
5229     }
5230     \int_compare:nNnT
5231     { \l__enumext_joined_item_viii_int }
5232     >
5233     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
5234     {
5235         \msg_warning:nnee { enumext } { item-joined-columns }
5236         { \int_use:N \l__enumext_joined_item_viii_int }
5237         {
5238             \int_eval:n
5239             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
5240         }
5241         \int_set:Nn \l__enumext_joined_item_viii_int
5242         {
5243             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
5244         }
5245     }
5246     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
5247     {
5248         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
5249         \int_decr:N \l__enumext_joined_item_aux_viii_int
5250         \int_add:Nn \l__enumext_item_column_pos_viii_int
5251         { \l__enumext_joined_item_aux_viii_int }
5252         \int_gadd:Nn \g__enumext_item_count_all_viii_int
5253         { \l__enumext_joined_item_aux_viii_int }
5254         \dim_set:Nn \l__enumext_joined_width_viii_dim
5255         {
5256             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
5257             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
5258                 + \l__enumext_columns_sep_viii_dim
5259             ) * \l__enumext_joined_item_aux_viii_int
5260         }
5261         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
5262     }
5263     {
5264         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
5265         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
5266     }
5267 }

```

(End of definition for `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n`)

13.47.5 Functions for mini-env, mini-right and mini-right* keys

The implementation of the **mini-env** key support is almost identical to the one used in the **enumext** and **keyans** environments, the difference is that the `\__enumext_mini_page` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

5268 \cs_new_protected:Nn \__enumext_start_mini_vii:
5269 {
5270     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }

```

```

5271 {
5272   \dim_set:Nn \l__enumext_minipage_left_vii_dim
5273   {
5274     \linewidth
5275     - \l__enumext_minipage_right_vii_dim
5276     - \l__enumext_minipage_hsep_vii_dim
5277   }
5278   \bool_set_true:N \l__enumext_minipage_active_vii_bool
5279   \dim_gset_eq:NN
5280     \g__enumext_minipage_right_vii_dim
5281     \l__enumext_minipage_right_vii_dim
5282   \__enumext_mini_addvspace_vii:
5283   \nointerlineskip\noindent
5284   \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
5285 }
5286 }

```

The function `\__enumext_stop_mini_vii:` closes the `\__enumext_mini_page` environment on the “left side”, applies `\hfill` and sets the variable `\g__enumext_minipage_active_vii_bool` to `true` which will be used in the function `\__enumext_after_env:nn` to execute the `minipage` on the “right side”. At this point the `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` functions are executed, stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `\__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.48). Before closing the `\__enumext_mini_page` the function `\__enumext_mini_capture_end:` stores the space “below”.

```

5287 \cs_new_protected:Nn \__enumext_stop_mini_vii:
5288 {
5289   \bool_if:NTF \l__enumext_minipage_active_vii_bool
5290   {
5291     \__enumext_stop_list:
5292     \__enumext_stop_store_level_vii:
5293     \__enumext_mini_capture_end:
5294     \IfDocumentMetadataT { \tag_resume:n {enumext*} }
5295     \end__enumext_mini_page
5296     \hfill
5297     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
5298   }
5299   {
5300     \__enumext_stop_list:
5301     \__enumext_stop_store_level_vii:
5302   }
5303 }

```

(End of definition for `\__enumext_start_mini_vii:` and `\__enumext_stop_mini_vii:`.)

Finally the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `minipage` environment on the “right side” is executed. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, it is passed to a box and then printed. Once the `minipage` on the “right side” is closed, `\__enumext_mini_apply_end:` adds the stored space “below”. This minipage is opened after the environment has ended, so it is placed in a group; otherwise the assignments that $\text{\LaTeX}$ makes when *tagging* PDF is active (for example, the one that makes a heading warn that it is inside a box) would remain active after the list.

```

5304 \__enumext_after_env:nn {enumext*}
5305 {
5306   \bool_if:NT \g__enumext_minipage_active_vii_bool
5307   {
5308     \group_begin:
5309     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
5310     \legacy_if_gset_false:n { @minipage }
5311     \skip_vertical:N \c_zero_skip
5312     \par\addvspace { \g__enumext_minipage_right_skip }
5313     \bool_if:NF \g__enumext_minipage_center_vii_bool
5314     {
5315       \tl_gput_left:Nn \g__enumext_miniright_code_vii_tl
5316       {
5317         \centering
5318       }
5319     }
5320     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
5321     {

```

```

5322         \tl_use:N \g__enumext_miniright_code_vii_tl
5323     }
5324     \box_use_drop:N \l__enumext_miniright_code_vii_box
5325     \skip_vertical:N \c_zero_skip
5326     \__enumext_endminipage:
5327     \group_end:
5328     \__enumext_mini_apply_end:
5329 }
5330 \bool_gset_false:N \g__enumext_minipage_active_vii_bool
5331 \bool_gset_true:N \g__enumext_minipage_center_vii_bool
5332 \tl_gclear:N \g__enumext_miniright_code_vii_tl
5333 \dim_gzero:N \g__enumext_minipage_right_vii_dim
5334 \bool_gset_false:N \g__enumext_starred_bool
5335 }

```

`\__enumext_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

`\__enumext_stop_mini_viii:`

```

5336 \cs_new_protected:Nn \__enumext_start_mini_viii:
5337 {
5338     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
5339     {
5340         \dim_set:Nn \l__enumext_minipage_left_viii_dim
5341         {
5342             \linewidth
5343             - \l__enumext_minipage_right_viii_dim
5344             - \l__enumext_minipage_hsep_viii_dim
5345         }
5346         \bool_set_true:N \l__enumext_minipage_active_viii_bool
5347         \dim_gset_eq:NN
5348             \g__enumext_minipage_right_viii_dim
5349             \l__enumext_minipage_right_viii_dim
5350         \__enumext_mini_addvspace_viii:
5351         \nointerlineskip\noindent
5352         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
5353     }
5354 }
5355 \cs_new_protected:Nn \__enumext_stop_mini_viii:
5356 {
5357     \bool_if:NTF \l__enumext_minipage_active_viii_bool
5358     {
5359         \__enumext_stop_list:
5360         \__enumext_mini_capture_end:
5361         \IfDocumentMetadataT { \tag_resume:n {keyans*} }
5362         \end__enumext_mini_page
5363         \hfill
5364         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
5365     }
5366     {
5367         \__enumext_stop_list:
5368     }
5369 }
5370 \__enumext_after_env:n {keyans*}
5371 {
5372     \bool_if:NT \g__enumext_minipage_active_viii_bool
5373     {
5374         \group_begin:
5375         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
5376         \par\addvspace { \g__enumext_minipage_right_skip }
5377         \bool_if:NF \g__enumext_minipage_center_viii_bool
5378         {
5379             \tl_gput_left:Nn \g__enumext_miniright_code_viii_tl
5380             {
5381                 \centering
5382             }
5383         }
5384         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
5385         {
5386             \tl_use:N \g__enumext_miniright_code_viii_tl
5387         }
5388         \box_use_drop:N \l__enumext_miniright_code_viii_box
5389         \end__enumext_mini_page

```

```

5390     \group_end:
5391     \__enumext_mini_apply_end:
5392   }
5393   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
5394   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
5395   \tl_gclear:N \g__enumext_miniright_code_viii_tl
5396   \dim_gzero:N \g__enumext_minipage_right_viii_dim
5397 }

```

(End of definition for __enumext_start_mini_viii: and __enumext_stop_mini_viii:.)

13.48 The environment `enumext*`

`enumext*` The environment accepts an optional [`<key = val>`] argument in ‘#1’ that is processed after checking where it is running. The environment is generated and a temporary definition is given to `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_first_item_tmp_vii:` and next to `\item` equal to the switch `\__enumext_item_switch_vii:` (§13.48.2), which decides whether `\item` belongs to `enumext*` (`\__enumext_start_item_tmp_vii:`, redefined later) or to a standard list nested inside it; the nesting depth of the `enumext*` environment is stored in `\l__enumext_base_depth_int`. Unlike the implementation used by the `shortlst` package, the values of `\rightskip` and `\@rightskip` are not set equal to `\@flushglue`, whose value is `0.0pt plus 1.0 fil`; in the tests performed this fails in some circumstances and different results are obtained when using pdf $\TeX$ and Lua $\TeX$.

```

5398 \NewDocumentEnvironment{enumext*}{o}{
5399   {
5400     \__enumext_safe_exec_vii:
5401     \__enumext_parse_keys_vii:n {#1}
5402     \__enumext_before_list_vii:
5403     \__enumext_start_store_level_vii:
5404     \__enumext_nested_save:
5405     \__enumext_start_list:nn { }
5406     {
5407       \__enumext_list_arg_two_vii:
5408       \__enumext_before_keys_exec_vii:
5409     }
5410     \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
5411     \__enumext_starred_columns_set_vii:
5412     \item[] \scan_stop:

```

For compatibility with *tagged* PDF in $\TeX$ release 2026-06-01, `\setcounter` is set at this point in the environment definition, thereby avoiding the problem reported by Tim Prescott (@teepeeemm) in `enumext*` nested in `enumext` starts at 2 with `DocumentMetadata` this was caused by the `latex-lab-block`[15] update in this release.

```

5413     \setcounter { enumXvii }
5414     { \int_eval:n { \int_use:c { \l__enumext_start_vii_int } - 1 } }
5415     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
5416     \cs_set_eq:NN \item \__enumext_item_switch_vii:
5417     \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
5418     \bool_set_false:N \l__enumext_trivlist_bool
5419     \ignorespaces
5420   }
5421   {
5422     % {tag=text-unit}
5423     \IfDocumentMetadataT { \tag_struct_end:n {tag=semantic-para} }
5424     \__enumext_stop_item_tmp_vii:
5425     \__enumext_remove_extra_parsep_vii:
5426     \__enumext_after_list_vii:
5427   }

```

(End of definition for `enumext*`. This function is documented on page 5.)

`\__enumext_safe_exec_vii:` The function `\__enumext_is_not_nested:` is called first, which sets `\g__enumext_starred_bool` to true if it is NOT nested within `enumext`; then the function `\__enumext_internal_mini_page:` is called to create the environment `\__enumext_mini_page`, `\l__enumext_level_h_int` is incremented to restrict nesting of the environment, `\l__enumext_starred_bool` is set to true, and finally the function `\__enumext_is_on_first_level:` is called, which sets `\l__enumext_starred_first_bool` to true if it is not nested, allowing the “storage system” to be used.

```

5428 \cs_new_protected:Nn \__enumext_safe_exec_vii:
5429   {
5430     \__enumext_is_not_nested:

```

```

5431     \__enumext_internal_mini_page:
5432     \int_incr:N \__enumext_level_h_int
5433     \int_compare:nNnT { \__enumext_level_h_int } > { 1 }
5434     {
5435         \msg_error:nn { enumext } { nested }
5436     }
5437     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
5438     {
5439         \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
5440     }
5441     \bool_set_true:N \__enumext_starred_bool
5442     \bool_set_false:N \__enumext_standard_bool
5443     \__enumext_is_on_first_level:
5444 }

```

(End of definition for __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n

It first checks the state of the variable __enumext_resume_count_vii_bool set by the key `resume` and calls the function `\__enumext_resume_last_counter:` if it is `true`, then clears the variable `\__enumext_series_name_str` used by the key `series`, processes the environment [`<key = val>`] and executes the function `\__enumext_parse_series:n` used by the key `series`, then executes the function `\__enumext_store_active_keys_vii:n` and reprocesses the `<keys>` to pass them to the storage *sequence* if the key `save-key` is not active.

Here it is necessary to check the status of `\__enumext_resume_count_vii_bool` in case the key `resume` is set using `\setenumext{enumext*}{resume}`.

```

5445 \cs_new_protected:Nn \__enumext_parse_keys_vii:n
5446 {
5447     \bool_if:NT \__enumext_resume_count_vii_bool
5448     {
5449         \__enumext_resume_last_counter:
5450     }
5451     \tl_if_novalue:nF {#1}
5452     {
5453         \str_clear:N \__enumext_series_name_str
5454         \keys_set:nn { enumext / enumext* } {#1}
5455         \bool_if:NF \__enumext_print_keyans_cmd_bool
5456         {
5457             \__enumext_parse_series:n {#1}
5458         }
5459         \__enumext_store_active_keys_vii:n {#1}
5460     }
5461 }

```

(End of definition for __enumext_parse_keys_vii:n.)

__enumext_before_list_vii:

The function `\__enumext_before_list_vii:` first calls the function `\__enumext_vspace_above_vii:` used by the keys `above` and `above*`, then calls the function `\__enumext_check_ans_active:` for the check answer mechanism and finally calls the functions `\__enumext_before_args_exec_vii:` and `\__enumext_start_mini_vii:` used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```

5462 \cs_new_protected:Nn \__enumext_before_list_vii:
5463 {
5464     \__enumext_vspace_above_vii:
5465     \__enumext_check_ans_active:
5466     \__enumext_before_args_exec_vii:
5467     \__enumext_start_mini_vii:
5468 }

```

(End of definition for __enumext_before_list_vii:.)

__enumext_after_list_vii:

The function `\__enumext_after_list_vii:` first calls the function `\__enumext_stop_mini_vii:` which internally calls `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` (§13.47.5) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions `\__enumext_after_stop_list_vii:` used by the key `after`, `\__enumext_check_ans_key_hook:` used by the key `check-ans`, `\__enumext_vspace_below_vii:` used by the keys `below` and `below*`. Finally set `\__enumext_starred_bool` to false and call the `\__enumext_starred_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

5469 \cs_new_protected:Nn \__enumext_after_list_vii:

```

```

5470 {
5471   \__enumext_stop_mini_vii:
5472   \__enumext_after_stop_list_vii:
5473   \__enumext_check_ans_key_hook:
5474   \__enumext_vspace_below_vii:
5475   \bool_set_false:N \l__enumext_starred_bool
5476   \bool_if:NF \l__enumext_print_keyans_cmd_bool
5477   {
5478     \__enumext_starred_save_counter:
5479   }
5480 }

```

(End of definition for __enumext_after_list_vii:.)

```

\__enumext_start_store_level_vii:
\__enumext_stop_store_level_vii:

```

The `\__enumext_start_store_level_vii:` and `\__enumext_stop_store_level_vii:` functions activate the “*storing structure*” mechanism in *sequence* for `\anskey` command and `anskey*` environment if `enumext*` are nested in `enumext`.

```

5481 \cs_new_protected:Nn \__enumext_start_store_level_vii:
5482 {
5483   \bool_if:NT \l__enumext_store_active_bool
5484   {
5485     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
5486     {
5487       \__enumext_store_level_open_vii:
5488     }
5489   }
5490 }
5491 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
5492 {
5493   \bool_if:NT \l__enumext_store_active_bool
5494   {
5495     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
5496     {
5497       \__enumext_store_level_close_vii:
5498     }
5499   }
5500 }

```

(End of definition for __enumext_start_store_level_vii: and __enumext_stop_store_level_vii:.)

13.48.1 The command \item in enumext*

```
\__enumext_first_item_tmp_vii:
```

The `\__enumext_first_item_tmp_vii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the “*first*” `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_vii:` function inside the environment body definition.

```

5501 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
5502 {
5503   \skip_horizontal:n
5504   {
5505     -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
5506   }
5507   \ignorespaces
5508 }

```

(End of definition for __enumext_first_item_tmp_vii:.)

```
\__enumext_start_item_tmp_vii:
```

```
\__enumext_item_peek_args_vii:
```

```
\__enumext_joined_item_vii:w
```

```
\__enumext_standard_item_vii:w
```

```
\__enumext_starred_item_vii:w
```

First the function `\__enumext_stop_item_tmp_vii:` is called, which is redefined later, then the value of `\l__enumext_item_column_pos_vii_int`, which counts the items by rows, and the value of `\g__enumext_item_count_all_vii_int`, which counts the total of items in the environment, are incremented. After that the function `\__enumext_item_peek_args_vii:` is called, which handles the arguments passed to `\item`.

```

5509 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
5510 {
5511   \__enumext_stop_item_tmp_vii:
5512   \int_incr:N \l__enumext_item_column_pos_vii_int
5513   \int_gincr:N \g__enumext_item_count_all_vii_int
5514   \__enumext_item_peek_args_vii:
5515 }

```

The function `\__enumext_item_peek_args_vii:` handles the `\item(<number>)`. It looks for the argument “(” ; if it is present, it calls the function `\__enumext_joined_item_vii:w (<number>)`, which is in charge of joining the items in the same row; if it is not present, it sets the default value (1).

```
5516 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
5517 {
5518   \peek_meaning:NTF (
5519     { \__enumext_joined_item_vii:w }
5520     { \__enumext_joined_item_vii:w (1) }
5521   }
```

The function `\__enumext_joined_item_vii:w` will first call the function `\__enumext_starred_joined_item_vii:w` in charge of setting the *width* of the box that stores the content passed to `\item`. Then it looks for the argument “*”; if it is present, it calls the function `\__enumext_starred_item_vii:w`, otherwise it calls the function `\__enumext_standard_item_vii:w`.

```
5522 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
5523 {
5524   \__enumext_starred_joined_item_vii:n {#1}
5525   \peek_meaning_remove:NTF *
5526     { \__enumext_starred_item_vii:w }
5527     { \__enumext_standard_item_vii:w }
5528 }
```

The function `\__enumext_standard_item_vii:w` will first look for the argument ‘[’, if present it will set the state of the variable `\l__enumext_wrap_label_vii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_vii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[<custom>]` by means of the function `\__enumext_start_item_vii:w`; otherwise it sets the value of the variable `\l__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]`.

```
5529 \cs_new_protected:Npn \__enumext_standard_item_vii:w
5530 {
5531   \bool_set_false:N \l__enumext_item_starred_vii_bool
5532   \peek_meaning:NTF [
5533     {
5534       \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
5535       \__enumext_start_item_vii:w
5536     }
5537     {
5538       \bool_set_true:N \l__enumext_wrap_label_vii_bool
5539       \legacy_if_set_true:n { @noitemarg }
5540       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5541     }
5542 }
```

The function `\__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*(<symbol>)` and `\item*(<symbol>)[<offset>]`.

```
5543 \cs_new_protected:Npn \__enumext_starred_item_vii:w
5544 {
5545   \bool_set_true:N \l__enumext_item_starred_vii_bool
5546   \bool_set_true:N \l__enumext_wrap_label_vii_bool
5547   \peek_meaning:NTF [
5548     { \__enumext_starred_item_vii_aux_i:w }
5549     { \__enumext_starred_item_vii_aux_ii:w }
5550   }
5551   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
5552   {
5553     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
5554     \__enumext_starred_item_vii_aux_ii:w
5555   }
5556   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
5557   {
5558     \peek_meaning:NTF [
5559       { \__enumext_starred_item_vii_aux_iii:w }
5560       {
5561         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
5562         \legacy_if_set_true:n { @noitemarg }
5563         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5564       }
5565     }
```

```

5566 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
5567 {
5568   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
5569   \legacy_if_set_true:n { @noitemarg }
5570   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5571 }

```

(End of definition for `\__enumext_start_item_tmp_vii:` and others.)

`\__enumext_fake_make_label_vii:n` The `\__enumext_fake_make_label_vii:n` function will be in charge of handling the definition of `\item`. First it increments the counter `enumXvii` for the enumerated items and activates support for the *check answers* mechanism, followed by support for `\item*[(symbol)][(offset)]` if present, then the `wrap-label` and `wrap-label*` keys, executed using `\makebox` whose width is given by the `labelwidth` key and position by the `align` key; inside its argument it executes the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally it executes the `labelsep` key applying a `\skip_horizontal:N` and `\ignorespaces`.

For compatibility with *tagged* PDF and `hyperref` when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to `true`. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi} twice (or more) creates destination with the same identifier`. This patch is only needed if you are running `pdflatex` and not if you are running `lua1latex`

```

5572 \cs_new_protected_nopar:Nn \__enumext_fake_make_label_vii:n
5573 {
5574   \legacy_if:nT { @noitemarg }
5575   {
5576     \legacy_if_set_false:n { @noitemarg }
5577     \legacy_if:nT { @nmbrlist }
5578     {
5579       \IfDocumentMetadataT
5580       {
5581         \bool_if:NT \l__enumext_hyperref_bool
5582         {
5583           \legacy_if_set_true:n { @hyper@item }
5584         }
5585       }
5586       \refstepcounter{enumXvii}
5587       \bool_if:NT \l__enumext_check_answers_bool
5588       {
5589         \int_gincr:N \g__enumext_item_number_int
5590         \bool_set_true:N \l__enumext_item_number_bool
5591       }
5592     }
5593   }
5594   \bool_if:NT \l__enumext_item_starred_vii_bool
5595   {
5596     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
5597     {
5598       \tl_gset_eq:NN
5599       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
5600     }
5601     \mode_leave_vertical:
5602     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
5603     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5604     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
5605     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5606   }
5607   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
5608   {
5609     \tl_use:N \l__enumext_label_font_style_vii_tl
5610     \bool_if:NTF \l__enumext_wrap_label_vii_bool
5611     {
5612       \__enumext_wrapper_label_vii:n {#1}
5613     }
5614     {#1}
5615   }
5616   \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
5617 }

```

(End of definition for `\__enumext_fake_make_label_vii:n`.)

13.48.2 The `\item` command inside nested lists

In the `enumext*` and `keyans*` environments `\item` is a *streaming* command: it closes the box of the previous item and opens the next one, so the `\item` of the environment and the `\item` of a list nested in the “*item content*”, for example `itemize`, run inside the same `minipage`. To distinguish them two things are used, depending on how the nested environment is built.

The environments based on `list` increase the depth of lists `\@listdepth`. This depth is stored in `\l__enumext_base_depth_int` when the environment starts and again after opening the `minipage` of an item, where the kernel resets it to zero. While `\@listdepth` is greater than the stored value `\item` belongs to the nested list and the copy `\__enumext_item_std:w` is executed; otherwise the *real* `\item` of the environment is executed.

The environments based on `\trivlist`, like `center` or the theorems, do not increase `\@listdepth`. For them the boolean `\l__enumext_trivlist_bool` is set to true by a wrapper of `\trivlist`. The assignment is local to the group of the nested environment, so the boolean is true while it is open and false again when it is closed, and the switch executes the copy `\__enumext_item_std:w` in the same way. The boolean is set to false when the environment starts and after opening each item.

- The nested lists inherit `\labelwidth` and `\labelsep` from the group in which they are opened, which is the one of the list that defines `enumext*`, where they have the values of the keys `labelwidth` and `labelsep`. Without a correction the label of `itemize` would be displaced and the space after the label of `description` and of the theorems would change. The values that they have before starting the environment are saved in `\l__enumext_outer_labelwidth_dim` and `\l__enumext_outer_labelsep_dim`, and `\__enumext_nested_reset:` restores them after opening the `minipage` of the item, or inside the group of an item of `item-break`. The nested list is therefore typeset as if it were in a `minipage` at that point of the document. The other parameters of the list, like `\leftmargin` or `\itemindent`, are always set by the nested list.

```
\l__enumext_base_depth_int
\l__enumext_trivlist_bool
\__enumext_trivlist_orig:
\l__enumext_outer_labelwidth_dim
\l__enumext_outer_labelsep_dim
\__enumext_item_switch:N
\__enumext_item_switch_vii:
\__enumext_item_switch_viii:
\__enumext_nested_save:
\__enumext_nested_reset:
```

First the variables: the depth of lists of the environment, the boolean that is true inside an environment based on `\trivlist` and the values of `\labelwidth` and `\labelsep` before starting the environment.

```
5618 \int_new:N \l__enumext_base_depth_int
5619 \bool_new:N \l__enumext_trivlist_bool
5620 \dim_new:N \l__enumext_outer_labelwidth_dim
5621 \dim_new:N \l__enumext_outer_labelsep_dim
```

The function `\__enumext_item_switch:N` executes the copy of the kernel `\item` when the depth of lists is greater than the stored one or when the boolean is true, otherwise it executes the function in its argument, the *real* `\item` of the environment.

```
5622 \cs_new_protected_nopar:Npn \__enumext_item_switch:N #1
5623 {
5624   \bool_lazy_or:nnTF
5625   { \int_compare_p:nNn { \@listdepth } > { \l__enumext_base_depth_int } }
5626   { \l__enumext_trivlist_bool }
5627   { \__enumext_item_std:w }
5628   { #1 }
5629 }
```

The function `\__enumext_nested_save:` is run before the list of the environment is opened and `\__enumext_nested_reset:` is run inside the item, after the label has been typeset; it stores the depth of lists, sets the boolean to false and restores the saved values.

```
5630 \cs_new_protected_nopar:Nn \__enumext_nested_save:
5631 {
5632   \dim_set_eq:NN \l__enumext_outer_labelwidth_dim \labelwidth
5633   \dim_set_eq:NN \l__enumext_outer_labelsep_dim \labelsep
5634 }
5635 \cs_new_protected_nopar:Nn \__enumext_nested_reset:
5636 {
5637   \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
5638   \bool_set_false:N \l__enumext_trivlist_bool
5639   \dim_set_eq:NN \labelwidth \l__enumext_outer_labelwidth_dim
5640   \dim_set_eq:NN \labelsep \l__enumext_outer_labelsep_dim
5641 }
```

At the beginning of the document `\trivlist` is replaced by a version that runs the original one and then sets the boolean to true, inside the group of the environment. The `cmd` hooks of the kernel are not used for this because the patch of `\trivlist` fails when PDF tagging is active, with the error `Use of \__hook_make_ prefixes:w doesn't match its definition`.

```
5642 \__enumext_at_begin_document:n
5643 {
5644   \cs_new_eq:NN \__enumext_trivlist_orig: \trivlist
```

```

5645 \cs_set_protected_nopar:Npn \trivlist
5646 {
5647   \__enumext_trivlist_orig:
5648   \bool_set_true:N \l__enumext_trivlist_bool
5649 }
5650 }

```

Finally the definition of `\item` in `enumext*` and `keyans*` respectively.

```

5651 \cs_new_protected_nopar:Nn \__enumext_item_switch_vii:
5652 { \__enumext_item_switch:N \__enumext_start_item_tmp_vii: }
5653 \cs_new_protected_nopar:Nn \__enumext_item_switch_viii:
5654 { \__enumext_item_switch:N \__enumext_start_item_tmp_viii: }

```

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

13.48.3 Real definition of `\item` in `enumext*`

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst`, no extra group is added around the *horizontal box* and the plain form of the `lrbox` environment, that is `\hbox_set_to_wd:Nnw` and `\hbox_set_end:`, is used instead of the `lrbox` environment.

`\__enumext_start_item_vii:w` The function `\__enumext_start_item_vii:w` chooses how the item is built: if the key `item-break` is active and the item occupies the whole row, that is, the number of joined columns is equal to the value of `columns` after the adjustments of `\__enumext_starred_joined_item_vii:n`, it calls `\__enumext_start_item_break_vii:n` (§13.48.4), otherwise it calls `\__enumext_start_item_box_vii:w`, described next.

In `\__enumext_start_item_box_vii:w` first the value of `\__enumext_stop_item_tmp_vii:` is set equal to `\__enumext_stop_item_vii:`, which is defined later; after that it starts capturing `\item` and “*item content*” in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```

5655 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5656 {
5657   \bool_lazy_and:nnTF
5658   { \l__enumext_item_break_vii_bool }
5659   {
5660     \int_compare_p:nNn
5661     { \l__enumext_joined_item_vii_int } = { \l__enumext_columns_vii_int }
5662   }
5663   { \__enumext_start_item_break_vii:n {#1} }
5664   { \__enumext_start_item_box_vii:w [#1] }
5665 }
5666 \cs_new_protected_nopar:Npn \__enumext_start_item_box_vii:w [#1]
5667 {
5668   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5669   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
5670   {
5671     \l__enumext_joined_width_vii_dim
5672     + \l__enumext_labelwidth_vii_dim
5673     + \l__enumext_labelsep_vii_dim
5674   }

```

Redefine the `\footnote` command.

```

5675 \__enumext_renew_footnote_starred:

```

Now the *sockets* for *tagging* PDF support are inserted and `\item` is run.

```

5676 \__enumext_start_list_tag:n {enumext*}
5677 \__enumext_fake_make_label_vii:n {#1}
5678 \__enumext_stop_start_list_tag:

```

Finally the `minipage` environment is opened, the “*item content*” is captured, `\parindent` takes the value of the key `listparindent` and `\parskip` takes the value of the key `parsep`, then the keys `itemindent` and `first` are executed. Inside the `minipage` `\@listdepth` is reset to zero, so the nesting depth is stored again in `\l__enumext_base_depth_int` right after opening it; this allows the switch (§13.48.2) to recognize standard lists nested in the item.

Here the use of `\__enumext_unskip_unkern:` (run twice) and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[⟨opt⟩]` and the value passed to the key `itemindent` is incremented.

```

5679 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }

```

```

5680     \__enumext_nested_reset:
5681     \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5682     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5683     \__enumext_unskip_unkern:
5684     \__enumext_unskip_unkern:
5685     \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5686     \tl_use:N \l__enumext_fake_item_indent_vii_tl
5687     \tl_use:N \l__enumext_after_list_args_vii_tl
5688 }

```

The `\__enumext_stop_item_vii:` function will finish fetching `\item` and “*item content*” by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```

5689 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5690 {
5691     \__enumext_endminipage:
5692     \__enumext_stop_list_tag:n {enumext*}
5693     \hbox_set_end:

```

Here the *warnings* are reduced a bit by setting the value of `\hbadness` to `10000`, print `\item` and “*item content*” from the *horizontal box*.

```

5694     \int_set:Nn \hbadness { 10000 }
5695     \box_use_drop:N \l__enumext_item_text_vii_box

```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```

5696     \int_compare:nNnTF
5697     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
5698     {
5699         \par\noindent
5700         \int_zero:N \l__enumext_item_column_pos_vii_int
5701     }
5702     {
5703         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
5704     }
5705 }

```

(End of definition for `\__enumext_start_item_vii:w`, `\__enumext_start_item_box_vii:w`, and `\__enumext_stop_item_vii:`.)

13.48.4 Items without minipage: the `item-break` key

```

\__enumext_start_item_break_vii:n
\__enumext_stop_item_break_vii:

```

When the key `item-break` is active and the item occupies the whole row, the `<content>` is NOT captured in a *horizontal box*: it flows as the text of the list, so it can be broken between pages. The functions are the equivalent of `\__enumext_start_item_box_vii:w` and `\__enumext_stop_item_vii:` for this case.

The *label* is placed at the left of the row with a negative skip and the lines of the `<content>` are indented by the width of the `labelwidth` plus the `labelsep`. This is done the same way as the lists do: the width is added to `\@totalleftmargin`, subtracted from `\linewidth` and the paragraph shape is set again with `\parshape`. All the assignments to `\@totalleftmargin`, `\linewidth`, `\parshape`, `\parindent` and `\parskip` are local to the group that is opened here and closed in `\__enumext_stop_item_break_vii:` after the `\par` that ends the item. The *sockets* for *tagging* PDF are the same ones used by the items with box, but since there is no `minipage` to start the paragraph the marked content of the text is opened and closed by hand.

- The first idea was to make the indentation with `\leftskip`, but that is not safe, because commands such as `\centering` or `\raggedright` set it to zero and the indentation is lost. The alternative used here is the one of the `list` environment, but `\@totalleftmargin` and `\linewidth` are internal parameters of the $\TeX$ kernel and relying on them is not correct practice, so the code must be reviewed if the kernel changes them. The technique of adjusting these parameters, instead of using a surrounding list, comes from the `indent[16]` package.

```

5706 \cs_new_protected_nopar:Nn \__enumext_start_item_break_vii:n
5707 {
5708     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_break_vii:
5709     \group_begin:
5710         \dim_add:Nn \@totalleftmargin
5711         { \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim }
5712         \dim_sub:Nn \linewidth
5713         { \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim }
5714         \parshape \c_one_int \@totalleftmargin \linewidth
5715         \skip_horizontal:n
5716         { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
5717         \__enumext_start_list_tag:n {enumext*}
5718         \__enumext_fake_make_label_vii:n {#1}

```

```

5719     \__enumext_nested_reset:
5720     \__enumext_stop_start_break_tag:
5721     \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5722     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5723     \tl_use:N \l__enumext_fake_item_indent_vii_tl
5724     \tl_use:N \l__enumext_after_list_args_vii_tl
5725   }
5726   \cs_new_protected_nopar:Nn \__enumext_stop_item_break_vii:
5727   {
5728     \__enumext_stop_item_break_tag:n {enumext*}
5729     \par
5730     \group_end:
5731     \int_zero:N \l__enumext_item_column_pos_vii_int
5732     \noindent
5733   }

(End of definition for \__enumext_start_item_break_vii:n and \__enumext_stop_item_break_vii:.)

```

__enumext_remove_extra_parsep_vii:

Remove the extra *vertical space* equal to `\parsep=\itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the removal of the last `<skip>` with `\__enumext_unskip_unkern:`, `\unskip` or `\removelastskip` does not obtain the expected result, the extra space remains; the option is to add a negative `<skip>` with the value of `itemsep` using `\skip_vertical:n`, the `enumext` interface of `\vskip`, after ending the paragraph with `\para_end:`, so that $\text{\TeX}$ is in `<vertical mode>` and the skip is applied directly to the vertical list. A zero skip is added after it and, finally, the counter of *all* the items is reset to zero.

```

5734 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
5735 {
5736   \int_compare:nNnT
5737   {
5738     \int_mod:nn
5739     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
5740   }
5741   =
5742   { 0 }
5743   {
5744     \para_end:
5745     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
5746     \skip_vertical:N \c_zero_skip
5747     \int_gzero:N \g__enumext_item_count_all_vii_int
5748   }
5749 }

```

(End of definition for __enumext_remove_extra_parsep_vii:.)

Since the `check-ans` check must not be executed by levels but on the complete list, it is taken out of the `enumext*` environment using the “hook” function `\__enumext_after_env:nn`.

```

5750 \__enumext_after_env:nn {enumext*}
5751 {
5752   \__enumext_execute_after_env:
5753 }

```

13.49 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `\__enumext_check_starred_cmd:n` function added in the second part. It accepts an optional `[<key = val>]` argument in ‘#1’ that is processed after checking that the `save-ans` key is active and where the environment is running.

```

5754 \NewDocumentEnvironment{keyans*}{ o }
5755 {
5756   \__enumext_safe_exec_viii:
5757   \__enumext_parse_keys_viii:n {#1}
5758   \__enumext_before_list_viii:
5759   \__enumext_nested_save:
5760   \__enumext_start_list:nn { }
5761   {
5762     \__enumext_list_arg_two_viii:
5763     \__enumext_before_keys_exec_viii:
5764   }
5765   \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
5766   \__enumext_starred_columns_set_viii:

```

```

5767 \item[] \scan_stop:
5768 % Set start counter
5769 \setcounter { enumXviii }
5770 { \int_eval:n { \int_use:c { l__enumext_start_viii_int } - 1 } }
5771 \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5772 \cs_set_eq:NN \item \__enumext_item_switch_viii:
5773 \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
5774 \bool_set_false:N \l__enumext_trivlist_bool
5775 \ignorespaces
5776 }
5777 {
5778 % {tag=text-unit}
5779 \IfDocumentMetadataT { \tag_struct_end:n {tag=semantic-para} }
5780 \__enumext_stop_item_tmp_viii:
5781 \__enumext_remove_extra_parsep_viii:
5782 \__enumext_check_starred_cmd:n { item }
5783 \__enumext_after_list_viii:
5784 }

```

(End of definition for `keyans*`. This function is documented on page 17.)

`\__enumext_safe_exec_viii:` The `\__enumext_safe_exec_viii:` function first checks if the `save-ans` key is active; only when this is true is the environment available. It increments the value of `\l__enumext_keyans_level_h_int` and returns an error message when the environment is nested, then calls the `\__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then checks if `keyans*` is being nested in `enumext*`, returning an error, and sets `\l__enumext_starred_bool` to true; finally it checks if it is at the appropriate level within the `enumext` environment.

```

5785 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5786 {
5787   \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
5788   {
5789     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
5790   }
5791   \int_incr:N \l__enumext_keyans_level_h_int
5792   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5793   {
5794     \msg_error:nn { enumext } { nested }
5795   }
5796   \__enumext_keyans_name_and_start:
5797   \bool_if:NT \l__enumext_starred_bool
5798   {
5799     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5800   }
5801   \bool_set_true:N \l__enumext_starred_bool
5802   % Set false for interfering with enumext nested in keyans*
5803   % (yes, its possible and crayze)
5804   \bool_set_false:N \l__enumext_store_active_bool
5805   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5806   {
5807     \msg_error:nn { enumext } { keyans-wrong-level }
5808   }
5809 }

```

(End of definition for `\__enumext_safe_exec_viii:`.)

`\__enumext_parse_keys_viii:n` Parse [`<key = val>`] for `keyans*`.

```

5810 \cs_new_protected:Nn \__enumext_parse_keys_viii:n
5811 {
5812   \tl_if_novalue:nF {#1}
5813   {
5814     \keys_set:nn { enumext / keyans* } {#1}
5815   }
5816 }

```

(End of definition for `\__enumext_parse_keys_viii:n`.)

`\__enumext_before_list_viii:` The function `\__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{<code>}` defined by the `before*` key if it is active (function

`\__enumext_before_args_exec_viii:`), then call the function `\__enumext_start_mini_viii:` handled by `mini-env`.

```
5817 \cs_new_protected:Nn \__enumext_before_list_viii:
5818 {
5819     \__enumext_vspace_above_viii:
5820     \__enumext_before_args_exec_viii:
5821     \__enumext_start_mini_viii:
5822 }
```

(End of definition for `\__enumext_before_list_viii:`)

`\__enumext_after_list_viii:` The function `\__enumext_after_list_viii:` first calls the function `\__enumext_stop_mini_viii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```
5823 \cs_new_protected:Nn \__enumext_after_list_viii:
5824 {
5825     \__enumext_stop_mini_viii:
5826     \__enumext_after_stop_list_viii:
5827     \__enumext_vspace_below_viii:
5828 }
```

(End of definition for `\__enumext_after_list_viii:`)

13.49.1 The command `\item` in `keyans*`

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the *optional argument* (`\number`) which works in the same way as in the `enumext*` environment. In simple terms, the goal is to store the `\label` next to the `[\content]` if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[\content]`, `\item(\number)*` and `\item(\number)*[\content]` commands.

`\__enumext_first_item_tmp_viii:` The `\__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the “first” `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```
5829 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5830 {
5831     \skip_horizontal:n
5832     {
5833         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
5834     }
5835     \ignorespaces
5836 }
```

(End of definition for `\__enumext_first_item_tmp_viii:`)

`\__enumext_start_item_tmp_viii:` First the function `\__enumext_stop_item_tmp_viii:` is called, which is redefined later, then the value of `\l__enumext_item_column_pos_viii_int`, which counts the items by rows, and the value of `\g__enumext_item_count_all_viii_int`, which counts the total of items in the environment, are incremented. After that the function `\__enumext_item_peek_args_viii:` is called, which handles the arguments passed to `\item`.

```
5837 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5838 {
5839     \__enumext_stop_item_tmp_viii:
5840     \int_incr:N \l__enumext_item_column_pos_viii_int
5841     \int_gincr:N \g__enumext_item_count_all_viii_int
5842     \__enumext_item_peek_args_viii:
5843 }
```

The function `\__enumext_item_peek_args_viii:` handles the `\item(\number)`. It looks for the argument “(” ; if it is present, it calls the function `\__enumext_joined_item_viii:w(\number)`, which is in charge of joining the items in the same row; if it is not present, it sets the default value (1).

```
5844 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5845 {
5846     \peek_meaning:NTF (
5847         { \__enumext_joined_item_viii:w }
5848         { \__enumext_joined_item_viii:w (1) }
5849     }
```

The function `\__enumext_joined_item_viii:w` will first call the function `\__enumext_starred_joined_item_v` in charge of setting the *width* of the box that stores the content passed to `\item`. Then it looks for the argument “*”; if it is present, it calls the function `\__enumext_starred_item_viii:w`, otherwise it calls the function `\__enumext_standard_item_viii:w`.

```
5850 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5851 {
5852   \__enumext_starred_joined_item_viii:n {#1}
5853   \peek_meaning_remove:NTF *
5854   { \__enumext_starred_item_viii:w }
5855   { \__enumext_standard_item_viii:w }
5856 }
```

The function `\__enumext_standard_item_viii:w` will first look for the argument ‘[’, if present it will set the state of the variable `\l__enumext_wrap_label_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[⟨custom⟩]` by means of the function `\__enumext_start_item_viii:w`; otherwise it sets the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]`.

```
5857 \cs_new_protected:Npn \__enumext_standard_item_viii:w
5858 {
5859   \bool_set_false:N \l__enumext_item_starred_viii_bool
5860   \bool_set_false:N \l__enumext_item_wrap_key_bool
5861   \peek_meaning:NTF [
5862   {
5863     \bool_set_eq:NN \l__enumext_wrap_label_viii_bool
5864     \l__enumext_wrap_label_opt_viii_bool
5865     \__enumext_start_item_viii:w
5866   }
5867   {
5868     \bool_set_true:N \l__enumext_wrap_label_viii_bool
5869     \legacy_if_set_true:n { @noitemarg }
5870     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5871   }
5872 }
```

(End of definition for `\__enumext_start_item_tmp_viii:` and others.)

```
\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_keyans_starred_item_star:
```

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

```
5873 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5874 {
5875   \bool_set_true:N \l__enumext_item_starred_viii_bool
5876   \bool_set_true:N \l__enumext_item_wrap_key_bool
5877   \bool_set_true:N \l__enumext_wrap_label_viii_bool
5878   \peek_meaning:NTF [
5879   { \__enumext_starred_item_viii_aux_i:w }
5880   { \__enumext_starred_item_viii_aux_ii:w }
5881 }
```

The function `\__enumext_starred_item_viii_aux_i:w` will save the *optional argument* to `\item*` in `\l__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_current_label_tl` if present, then call the function `\__enumext_starred_item_viii_aux_ii:w`.

```
5882 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5883 {
5884   \tl_clear:N \l__enumext_store_current_label_tl
5885   \tl_if_no_value:nF {#1}
5886   {
5887     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5888     {
5889       \tl_put_right:NV \l__enumext_store_current_label_tl
5890       \l__enumext_store_keyans_item_opt_sep_viii_tl
5891       \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
5892     }
5893     \tl_set:Nn \l__enumext_store_current_opt_arg_tl {#1}
5894   }
5895   \__enumext_starred_item_viii_aux_ii:w
5896 }
```

```

5897 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5898 {
5899   \legacy_if_set_true:n { @noitemarg }
5900   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5901 }

```

The function `\__enumext_keyans_starred_item_star:` will be in charge of storing the current *label* for `\item*` followed by the `[<content>]` for `\item*<content>` if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos`, `mark-sep` and `save-ref` are implemented.

```

5902 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5903 {
5904   \tl_put_left:Ne \__enumext_store_current_label_tl { \__enumext_label_viii_tl }
5905   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
5906   \__enumext_keyans_store_ref:
5907   \tl_put_left:Nn \__enumext_store_current_label_tl { \item }
5908   \__enumext_keyans_addto_seq_link:
5909   \int_gincr:N \g__enumext_check_starred_cmd_int
5910   \dim_compare:nNnT { \__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5911   {
5912     \dim_set:Nn \__enumext_mark_sym_sep_viii_dim { \__enumext_labelsep_viii_dim }
5913   }
5914   \bool_if:NT \__enumext_show_answer_bool
5915   {
5916     \tl_set_eq:NN \__enumext_mark_answer_sym_tl \__enumext_mark_answer_sym_viii_tl
5917     \str_set_eq:NN \__enumext_mark_position_str \__enumext_mark_position_viii_str
5918     \__enumext_print_keyans_box:NN
5919     \__enumext_labelwidth_viii_dim \__enumext_mark_sym_sep_viii_dim
5920   }
5921   \bool_if:NT \__enumext_show_position_bool
5922   {
5923     \tl_set:Ne \__enumext_mark_answer_sym_tl
5924     {
5925       \group_begin:
5926       \exp_not:N \normalfont
5927       \exp_not:N \footnotesize [ \int_eval:n
5928       {
5929         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
5930       }
5931       ]
5932       \group_end:
5933     }
5934     \str_set_eq:NN \__enumext_mark_position_str \__enumext_mark_position_viii_str
5935     \__enumext_print_keyans_box:NN
5936     \__enumext_labelwidth_viii_dim \__enumext_mark_sym_sep_viii_dim
5937   }
5938 }

```

(End of definition for `\__enumext_starred_item_viii:w` and others.)

```

\__enumext_keyans_wrapper_label_viii:n
\__enumext_fake_make_label_viii:n
\__enumext_wrapper_label_viii:n

```

The implementation at this point is very similar to that of the `enumext*` environment.

```

5939 \cs_new_protected:Nn \__enumext_keyans_wrapper_label_viii:n
5940 {
5941   \bool_lazy_all:nT
5942   {
5943     { \bool_if_p:N \__enumext_wrap_label_viii_bool }
5944     { \bool_if_p:N \__enumext_show_answer_bool }
5945     { \bool_if_p:N \__enumext_item_wrap_key_bool }
5946     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5947   }
5948   {
5949     \cs_set_eq:NN
5950     \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5951   }
5952   \bool_if:NTF \__enumext_wrap_label_viii_bool
5953   {
5954     \__enumext_wrapper_label_viii:n {#1}
5955   }
5956   {#1}
5957 }
5958 \cs_new_protected_nopar:Nn \__enumext_fake_make_label_viii:n

```

```

5959 {
5960   \legacy_if:nT { @noitemarg }
5961   {
5962     \legacy_if_set_false:n { @noitemarg }
5963     \legacy_if:nT { @nmbrrlist }
5964     {
5965       \IfDocumentMetadataT
5966       {
5967         \bool_if:NT \l__enumext_hyperref_bool
5968         {
5969           \legacy_if_set_true:n { @hyper@item }
5970         }
5971       }
5972       \refstepcounter{enumXviii}
5973     }
5974   }
5975   \bool_if:NT \l__enumext_item_starred_viii_bool
5976   {
5977     \__enumext_keyans_starred_item_star:
5978   }
5979   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5980   {
5981     \tl_use:N \l__enumext_label_font_style_viii_tl
5982     \__enumext_keyans_wrapper_label_viii:n {#1}
5983   }
5984   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5985 }

```

(End of definition for `\__enumext_keyans_wrapper_label_viii:n`, `\__enumext_fake_make_label_viii:n`, and `\__enumext_wrapper_label_viii:n`.)

13.49.2 Real definition of `\item` in `keyans*`

The implementation at this point is very similar to that of the `enumext*` environment; the function `\__enumext_start_item_viii:w` chooses between `\__enumext_start_item_break_viii:n` and `\__enumext_start_item_box_viii:w` in the same way.

```

5986 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5987 {
5988   \bool_lazy_and:nnTF
5989   { \l__enumext_item_break_viii_bool }
5990   {
5991     \int_compare_p:nNn
5992     { \l__enumext_joined_item_viii_int } = { \l__enumext_columns_viii_int }
5993   }
5994   { \__enumext_start_item_break_viii:n {#1} }
5995   { \__enumext_start_item_box_viii:w [#1] }
5996 }
5997 \cs_new_protected_nopar:Npn \__enumext_start_item_box_viii:w [#1]
5998 {
5999   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
6000   \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
6001   {
6002     \l__enumext_joined_width_viii_dim
6003     + \l__enumext_labelwidth_viii_dim
6004     + \l__enumext_labelsep_viii_dim
6005   }
6006   \__enumext_renew_footnote_starred:
6007   \__enumext_start_list_tag:n {keyans*}
6008   \__enumext_fake_make_label_viii:n {#1}
6009   \__enumext_stop_start_list_tag:
6010   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
6011   \__enumext_nested_reset:
6012   \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
6013   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
6014   \__enumext_unskip_unkern:
6015   \__enumext_unskip_unkern:
6016   \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
6017   \tl_use:N \l__enumext_fake_item_indent_viii_tl
6018   \bool_if:NT \l__enumext_item_starred_viii_bool
6019   {
6020     \__enumext_keyans_show_item_opt_viii:
6021   }

```

```

6022         \tl_use:N \l__enumext_after_list_args_viii_tl
6023     }
6024 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
6025 {
6026     \__enumext_endminipage:
6027     \__enumext_stop_list_tag:n {keyans*}
6028     \hbox_set_end:
6029     \int_set:Nn \hbadness { 10000 }
6030     \box_use_drop:N \l__enumext_item_text_viii_box
6031     \int_compare:nNnTF
6032     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
6033     {
6034         \par\noindent
6035         \int_zero:N \l__enumext_item_column_pos_viii_int
6036     }
6037     {
6038         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
6039     }
6040 }

```

(End of definition for `\__enumext_start_item_viii:w`, `\__enumext_start_item_box_viii:w`, and `\__enumext_stop_item_viii:`.)

```

\__enumext_start_item_break_viii:n
\__enumext_stop_item_break_viii:

```

Same implementation as `\__enumext_start_item_break_vii:n` and `\__enumext_stop_item_break_vii:` for the `keyans*` environment. The item option shown by `\item*` is placed at the beginning of the `(content)`.

```

6041 \cs_new_protected_nopar:Nn \__enumext_start_item_break_viii:n
6042 {
6043     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_break_viii:
6044     \group_begin:
6045         \dim_add:Nn \@totalleftmargin
6046         { \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim }
6047         \dim_sub:Nn \linewidth
6048         { \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim }
6049         \parshape \c_one_int \@totalleftmargin \linewidth
6050         \skip_horizontal:n
6051         { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
6052         \__enumext_start_list_tag:n {keyans*}
6053         \__enumext_fake_make_label_viii:n {#1}
6054         \__enumext_nested_reset:
6055         \__enumext_stop_start_break_tag:
6056         \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
6057         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
6058         \tl_use:N \l__enumext_fake_item_indent_viii_tl
6059         \bool_if:NT \l__enumext_item_starred_viii_bool
6060         { \__enumext_keyans_show_item_opt_viii: }
6061         \tl_use:N \l__enumext_after_list_args_viii_tl
6062     }
6063 \cs_new_protected_nopar:Nn \__enumext_stop_item_break_viii:
6064 {
6065     \__enumext_stop_item_break_tag:n {keyans*}
6066     \par
6067     \group_end:
6068     \int_zero:N \l__enumext_item_column_pos_viii_int
6069     \noindent
6070 }

```

(End of definition for `\__enumext_start_item_break_viii:n` and `\__enumext_stop_item_break_viii:`.)

```
\__enumext_remove_extra_parsep_viii:
```

The implementation at this point is very similar to that of the `enumext*` environment.

```

6071 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
6072 {
6073     \int_compare:nNnT
6074     {
6075         \int_mod:nn
6076         { \g__enumext_item_count_all_viii_int }
6077         { \l__enumext_columns_viii_int }
6078     }
6079     =
6080     { 0 }
6081     {

```

```

6082     \para_end:
6083     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
6084     \skip_vertical:N \c_zero_skip
6085     \int_gzero:N \g__enumext_item_count_all_viii_int
6086   }
6087 }

```

(End of definition for \l__enumext_remove_extra_parsep_viii:.)

13.50 The command \getkeyans

13.50.1 Definition of error messages for \getkeyans

missing-colon Message used by \getkeyans command.

```

6088 \msg_new:nnnn { enumext } { missing-colon }
6089 {
6090   The~argument~'#1'~has~no~colon~\msg_line_context:.
6091 }
6092 {
6093   The~argument~of~'\token_to_str:N \getkeyans'~must~have~the~form~
6094   'store~name:position'.
6095 }

```

(End of definition for missing-colon.)

13.50.2 Internal functions for \getkeyans

\l__enumext_getkeyans_aux:n The internal function \l__enumext_getkeyans_aux:n is in charge of *splitting* the *mandatory argument* using “:”. If “:” is omitted it will return an error.

```

6096 \cs_new_protected:Nn \l__enumext_getkeyans_aux:n
6097 {
6098   \str_if_in:nnTF {#1} { : }
6099   {
6100     \use:e
6101     {
6102       \cs_set:Npn \exp_not:N \l__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
6103       { {##1} {##2} }
6104     }
6105     \exp_after:w \l__enumext_getkeyans:nn \l__enumext_tmp:w #1 \scan_stop:
6106   }
6107   { \msg_error:nnn { enumext } { missing-colon } {#1} }
6108 }

```

The internal function \l__enumext_getkeyans:nn will check for the existence of the *prop list*, if it does not exist it will return an error message, then it will fetch the content specified by the *second argument* from *prop list*.

```

6109 \cs_new_protected:Nn \l__enumext_getkeyans:nn
6110 {
6111   \prop_if_exist:cTF { g__enumext_ #1 _prop }
6112   {
6113     \prop_item:cn { g__enumext_ #1 _prop }{#2}
6114   }
6115   {
6116     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
6117   }
6118 }

```

(End of definition for \l__enumext_getkeyans_aux:n and \l__enumext_getkeyans:nn.)

\getkeyans The \getkeyans command takes a *mandatory argument* of the form {<*store name*:*position*>} in ‘#1’. The argument is expanded and converted to a string before it is processed. Retrieve a “*single content*” stored by \anskey, \anspic* and \item* and anskey* from *prop list* defined by save-ans key.

```

6119 \NewDocumentCommand \getkeyans { m }
6120 {
6121   \exp_args:Ne \l__enumext_getkeyans_aux:n
6122   { \tl_to_str:e { \text_expand:n {#1} } }
6123 }

```

(End of definition for \getkeyans. This function is documented on page 20.)

13.51 The command `\printkeyans`

The `\printkeyans` command prints “*all stored content*” in the *sequence* defined by the `save-ans` key.

13.51.1 Definition of variables for `\printkeyans`

The command `\printkeyans` must not depend on the values that `\setenumext` sets for the environments, so the default values of the package are saved when it is loaded. The constant `\c__enumext_print_envs_clist` lists the environments that `\printkeyans` can open and the constant `\c__enumext_default_vars_clist` lists, as pairs $\langle name \rangle \langle type \rangle$, the local variables of each environment whose default value is saved. Each value is saved in a global variable with the same name as the local one, but starting with `\g__enumext_default_` instead of `\l__enumext_`, for example `\g__enumext_default_labelsep_ii_dim` for `\l__enumext_labelsep_ii_dim`. The default $\langle label \rangle$ is saved by the `label` key in `\c__enumext_default_label_X_tl`.

```
6124 \clist_const:Nn \c__enumext_print_envs_clist
6125 {
6126   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
6127 }
6128 \clist_const:Nn \c__enumext_default_vars_clist
6129 {
6130   {labelsep}{dim}, {labelwidth}{dim}, {fake_item_indent}{dim},
6131   {listoffset}{dim}, {rightmargin}{dim}, {listparindent}{dim},
6132   {leftmargin_tmp}{dim}, {leftmargin_tmp}{bool}, {columns_sep}{dim},
6133   {minipage_hsep}{dim}, {topsep}{skip}, {partopsep}{skip}, {parsep}{skip},
6134   {itemsep}{skip}, {columns}{int}, {start}{int}, {beginparpenalty}{int},
6135   {itempenalty}{int}, {endparpenalty}{int}, {label_fill_left}{tl},
6136   {label_fill_right}{tl}, {align_label_pos}{str}, {label_font_style}{tl},
6137   {after_list_args}{tl}, {wrap_label_opt}{bool}, {item_break}{bool}
6138 }
```

(End of definition for `\c__enumext_print_envs_clist`, `\c__enumext_default_vars_clist`, and `\c__enumext_default_label_X_tl`.)

The token list `\l__enumext_print_keyans_starred_tl` is used by the command `\printkeyans` (§13.51).

```
6139 \tl_new:N \l__enumext_print_keyans_starred_tl
```

(End of definition for `\l__enumext_print_keyans_starred_tl`.)

13.51.2 Definition of error messages for `\printkeyans`

`print-starred` Messages used by `\printkeyans*` command.

```
6140 \msg_new:nnnn { enumext } { print-starred }
6141 {
6142   '\token_to_str:N \printkeyans*':~The~sequence~'#1'~already~contains~
6143   #2~environment~\msg_line_context:.
6144 }
6145 {
6146   '\token_to_str:N \printkeyans*'~opens~'enumext*';~use~
6147   '\token_to_str:N \printkeyans'~without~the~star.
6148 }
```

(End of definition for `print-starred`.)

13.51.3 Definition of keys for `\printkeyans`

`print*` First a set of $\langle filtered\ keys \rangle$ is defined, with which the options of the different nesting levels are controlled for the environments `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

`print-1` The default values only contain what the printed content changes with respect to the default values of the package: `nosep`, `first` and `font`, plus `columns` for the “*first level*”. The remaining values are restored by the function `\__enumext_default_restore:` before these $\langle keys \rangle$ are run (§13.51).

`print-2` The variable `\l__enumext_print_keyans_starred_tl` will have the default $\langle keys \rangle$ for `\printkeyans*` and will be set by `\setenumext[ $\langle print \rangle$ ]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[ $\langle print \rangle$ ,*]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[ $\langle print \rangle$ , level]`.

```
6149 \keys_define:nn { enumext / print }
6150 {
6151   print* .code:n =
```

```

6152     {
6153         \keys_precompile:neN { enumext / enumext* }
6154         { \__enumext_filter_save_key:n {#1} }
6155         \l__enumext_print_keyans_starred_tl
6156     }, % starred cmd
6157     print* .initial:n =
6158     {
6159         nosep,
6160         columns = 2,
6161         first = \small,
6162         font = \small
6163     },
6164     print-1 .code:n =
6165     {
6166         \keys_precompile:neN { enumext / level-1 }
6167         { \__enumext_filter_save_key:n {#1} }
6168         \l__enumext_print_keyans_i_tl
6169     },
6170     print-1 .initial:n =
6171     {
6172         nosep,
6173         columns = 2,
6174         first = \small,
6175         font = \small
6176     },
6177     print-2 .code:n =
6178     {
6179         \keys_precompile:neN { enumext / level-2 }
6180         { \__enumext_filter_save_key:n {#1} }
6181         \l__enumext_print_keyans_ii_tl
6182     },
6183     print-2 .initial:n =
6184     {
6185         nosep,
6186         first = \small,
6187         font = \small
6188     },
6189     print-3 .code:n =
6190     {
6191         \keys_precompile:neN { enumext / level-3 }
6192         { \__enumext_filter_save_key:n {#1} }
6193         \l__enumext_print_keyans_iii_tl
6194     },
6195     print-3 .initial:n =
6196     {
6197         nosep,
6198         first = \small,
6199         font = \small
6200     },
6201     print-4 .code:n =
6202     {
6203         \keys_precompile:neN { enumext / level-4 }
6204         { \__enumext_filter_save_key:n {#1} }
6205         \l__enumext_print_keyans_iv_tl
6206     },
6207     print-4 .initial:n =
6208     {
6209         nosep,
6210         first = \small,
6211         font = \small
6212     },
6213     print-* .code:n =
6214     {
6215         \keys_precompile:neN { enumext / enumext* }
6216         { \__enumext_filter_save_key:n {#1} }
6217         \l__enumext_print_keyans_vii_tl
6218     }, % starred nested
6219     print-* .initial:n =
6220     {
6221         nosep,
6222         first = \small,

```

```

6223         font = \small
6224     },
6225 }

```

(End of definition for `print*` and others.)

13.51.4 Internal functions for `\printkeyans`

The function `\__enumext_default_save:nnn` saves the value of a local variable, if it exists, in the global variable with the same name that starts with `\g__enumext_default_`, and the function `\__enumext_default_restore:nnn` copies it back.

```

6226 \cs_new_protected:Nn \__enumext_default_save:nnn
6227 {
6228     \cs_if_exist:cT { l__enumext_ #1 _ #3 _ #2 }
6229     {
6230         \use:c { #2 _new:c } { g__enumext_default_ #1 _ #3 _ #2 }
6231         \use:c { #2 _gset_eq:cc }
6232         { g__enumext_default_ #1 _ #3 _ #2 } { l__enumext_ #1 _ #3 _ #2 }
6233     }
6234 }
6235 \cs_new_protected:Nn \__enumext_default_restore:nnn
6236 {
6237     \cs_if_exist:cT { g__enumext_default_ #1 _ #3 _ #2 }
6238     {
6239         \use:c { #2 _set_eq:cc }
6240         { l__enumext_ #1 _ #3 _ #2 } { g__enumext_default_ #1 _ #3 _ #2 }
6241     }
6242 }

```

The function `\__enumext_default_save_env:nn` saves all the variables of `\c__enumext_default_vars_clist` for one environment, together with the function of the `wrap-label` key. The function `\__enumext_default_restore_env:nn` restores them and runs the `label` key with the saved *label*, since this key also defines the counter style and the width of the label.

```

6243 \cs_new_protected:Nn \__enumext_default_save_env:nn
6244 {
6245     \clist_map_inline:Nn \c__enumext_default_vars_clist
6246     { \__enumext_default_save:nnn ##1 {#2} }
6247     \cs_gset_eq:cc { __enumext_default_wrapper_label_ #2 :n }
6248     { __enumext_wrapper_label_ #2 :n }
6249 }
6250 \cs_new_protected:Nn \__enumext_default_restore_env:nn
6251 {
6252     \clist_map_inline:Nn \c__enumext_default_vars_clist
6253     { \__enumext_default_restore:nnn ##1 {#2} }
6254     \cs_set_eq:cc { __enumext_wrapper_label_ #2 :n }
6255     { __enumext_default_wrapper_label_ #2 :n }
6256     \keys_set:ne { enumext / #1 }
6257     { label = { \exp_not:v { c__enumext_default_label_ #2 _tl } } }
6258 }

```

Finally the function `\__enumext_default_restore:` restores, for all the environments, the values saved when the package was loaded.

```

6259 \cs_new_protected:Nn \__enumext_default_restore:
6260 {
6261     \clist_map_inline:Nn \c__enumext_print_envs_clist
6262     { \__enumext_default_restore_env:nn ##1 }
6263 }
6264 \clist_map_inline:Nn \c__enumext_print_envs_clist
6265 { \__enumext_default_save_env:nn #1 }

```

(End of definition for `\__enumext_default_save:nnn` and others.)

```
\__enumext_printkeyans:nnn
```

The internal function `\__enumext_printkeyans:nnn` will check for the existence of the *sequence*, if it does not exist it will return an error message, then it will check if not empty.

```

6266 \cs_new_protected:Nn \__enumext_printkeyans:nnn
6267 {
6268     \seq_if_exist:cTF { g__enumext_ #3 _seq }
6269     {
6270         \seq_if_empty:cF { g__enumext_ #3 _seq }
6271         {

```

If the *starred argument* ‘*’ is present, it checks that the environment `enumext*` is not saved in the *sequence*, then executes the variable `\__enumext_print_keyans_starred_tl` that contains the default `<keys>` for the environment `enumext*`, sets `\__enumext_base_line_fix_bool` and `\__enumext_print_keyans_star_bool` to true for *baseline correction*, opens the `enumext*` environment passing the *optional argument* and maps the *sequence*, then sets `\__enumext_base_line_fix_bool` and `\__enumext_print_keyans_star_bool` to false.

```

6272         \bool_if:nTF {#1}
6273         {
6274             \seq_if_in:cnTF { g__enumext_ #3 _seq } { \end{enumext*} }
6275             {
6276                 \msg_error:nnnn { enumext } { print-starred } {#3}
6277                 { enumext* }
6278             }
6279             {
6280                 \tl_use:N \__enumext_print_keyans_starred_tl
6281                 \bool_set_true:N \__enumext_base_line_fix_bool
6282                 \bool_set_true:N \__enumext_print_keyans_star_bool
6283                 \begin{enumext*}[#2]
6284                     \seq_map_inline:cn { g__enumext_ #3 _seq } {##1}
6285                     \end{enumext*}
6286                     \bool_set_false:N \__enumext_base_line_fix_bool
6287                     \bool_set_false:N \__enumext_print_keyans_star_bool
6288                 }
6289             }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “first level” then map the *sequence*.

```

6290         {
6291             \begin{enumext}[#2]
6292             \seq_map_inline:cn { g__enumext_ #3 _seq } {##1}
6293             \end{enumext}
6294         }
6295     }
6296 }
6297 {
6298     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
6299 }
6300 }

```

(End of definition for `\__enumext_printkeyans:nnn`.)

`\printkeyans` The command accepts a *starred argument* ‘*’ in ‘#1’, an optional `[<key = val>]` in ‘#2’ and a mandatory `{<store name>}` in ‘#3’. Create a user command to print “all stored content” in *sequence* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group the default values saved when the package was loaded are restored, then the “precompiled keys” are run and finally the stored content is printed.

```

6301 \NewDocumentCommand \printkeyans { s O{} m }
6302 {
6303     \group_begin:
6304         \__enumext_default_restore:
6305         \bool_set_true:N \__enumext_print_keyans_cmd_bool
6306         \tl_use:N \__enumext_print_keyans_i_tl
6307         \tl_use:N \__enumext_print_keyans_ii_tl
6308         \tl_use:N \__enumext_print_keyans_iii_tl
6309         \tl_use:N \__enumext_print_keyans_iv_tl
6310         \tl_use:N \__enumext_print_keyans_vii_tl
6311         \__enumext_printkeyans:nnn {#1} {#2} {#3}
6312         \bool_set_false:N \__enumext_print_keyans_cmd_bool
6313     \group_end:
6314 }

```

(End of definition for `\printkeyans`. This function is documented on page 21.)

- The reason for storing `<keys>` in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

13.52 The command `\setenumext`

The command `\setenumext` is in charge of managing the `<keys>` passed to all environments and to the `\printkeyans` command. Precautions must be taken with the `enumext*` and `enumext` environments so as not to capture `<keys>` that cause problems.

13.52.1 Definition of variables for \setenumext

Internal variables used by the command `\setenumext` (§13.52).

```

6315 \tl_new:N \__enumext_setkey_tmpa_tl
6316 \tl_new:N \__enumext_setkey_tmpb_tl
6317 \int_new:N \__enumext_setkey_tmpa_int
6318 \seq_new:N \__enumext_setkey_tmpa_seq
6319 \seq_new:N \__enumext_setkey_tmpb_seq

```

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

13.52.2 Definition of error messages for \setenumext

Messages used by the `\setenumext` command.

```

invalid-level
unknown-key-family
6320 \msg_new:nnnn { enumext } { invalid-level }
6321 {
6322   The~level~'#1'~is~not~valid~\msg_line_context:.
6323 }
6324 {
6325   The~optional~argument~must~be~a~family~followed~by~the~levels~
6326   1,~2,~3,~4~or~'!',~for~example~'print,1'.
6327 }
6328 \msg_new:nnnn { enumext } { unknown-key-family }
6329 {
6330   Unknown~key~family~'\l_keys_key_str'~for~enumext~\msg_line_context:.
6331 }
6332 {
6333   The~first~element~of~the~optional~argument~must~be~
6334   'enumext',~'enumext*',~'keyans',~'keyans*'~or~'print'.
6335 }

```

(End of definition for `invalid-level` and `unknown-key-family`.)

13.52.3 Definition of keys for \setenumext

Now define a “meta families” of *(keys)* to access from `\setenumext`.

```

enumext-1
enumext-2
enumext-3
enumext-4
enumext*
6336 \keys_define:nn { enumext / meta-families }
6337 {
6338   enumext-1 .code:n =
6339   {
6340     \keys_set:ne { enumext / level-1 }
6341     {
6342       \__enumext_filter_level:n {#1}
6343     }
6344   },
6345   enumext-2 .code:n =
6346   {
6347     \keys_set:ne { enumext / level-2 }
6348     {
6349       \__enumext_filter_level:n {#1}
6350     }
6351   },
6352   enumext-3 .code:n =
6353   {
6354     \keys_set:ne { enumext / level-3 }
6355     {
6356       \__enumext_filter_level:n {#1}
6357     }
6358   },
6359   enumext-4 .code:n =
6360   {
6361     \keys_set:ne { enumext / level-4 }
6362     {
6363       \__enumext_filter_level:n {#1}
6364     }
6365   },
6366   enumext* .code:n =
6367   {
6368     \keys_set:ne { enumext / enumext* }
6369     {
6370       \__enumext_filter_level:n {#1}

```

```

6371     }
6372   },
6373   keyans      .meta:nn = { enumext / keyans } {#1},
6374   keyans*     .meta:nn = { enumext / keyans* } {#1},
6375   print*      .meta:nn = { enumext / print } { print* = {#1} },
6376   print-1     .meta:nn = { enumext / print } { print-1 = {#1} },
6377   print-2     .meta:nn = { enumext / print } { print-2 = {#1} },
6378   print-3     .meta:nn = { enumext / print } { print-3 = {#1} },
6379   print-4     .meta:nn = { enumext / print } { print-4 = {#1} },
6380   print-*     .meta:nn = { enumext / print } { print-* = {#1} },
6381   unknown     .code:n  = { \msg_error:nn { enumext } { unknown-key-family } },
6382 }

```

(End of definition for `enumext-1` and others.)

13.52.4 Internal functions for `\setenumext`

```

__enumext_filter_level:n
__enumext_filter_level_key:n
__enumext_filter_level_pair:nn

```

The function `__enumext_filter_level:n` will be in charge of filtering the *keys* passed to the `enumext` and `enumext*` environments.

```

6383 \cs_new:Nn __enumext_filter_level:n
6384 {
6385   \use:e
6386   {
6387     \keyval_parse:NNn
6388       __enumext_filter_level_key:n
6389       __enumext_filter_level_pair:nn {#1}
6390   }
6391 }

```

The function `__enumext_filter_level_key:n` will be responsible for filtering the *keys* that are passed “without value” by excluding the keys `resume*`, `reset` and `reset*` passed to the `enumext` and `enumext*` environments.

```

6392 \cs_new:Nn __enumext_filter_level_key:n
6393 {
6394   \str_case:nnF {#1}
6395   {
6396     { resume* } { } { reset } { } { reset* } { }
6397   }
6398   { , { \exp_not:n {#1} } }
6399 }

```

The function `__enumext_filter_level_pair:nn` will be responsible for filtering the *keys* that are passed “with value” by excluding the `series`, `resume` and `save-ans` keys passed to the `enumext` and `enumext*` environments.

```

6400 \cs_new:Nn __enumext_filter_level_pair:nn
6401 {
6402   \str_case:nnF {#1}
6403   {
6404     { series } { } { save-ans } { } { resume } { }
6405   }
6406   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
6407 }

```

(End of definition for `__enumext_filter_level:n`, `__enumext_filter_level_key:n`, and `__enumext_filter_level_pair:nn`.)

```

__enumext_set_parse:n
__enumext_set_error:n

```

Internal functions used by the `\setenumext` command.

```

6408 \cs_new_protected:Nn __enumext_set_parse:n
6409 {
6410   \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
6411   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
6412   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
6413   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
6414   {
6415     \seq_put_right:Nc \l__enumext_setkey_tmpa_seq
6416     { \tl_trim_spaces:n {#1} }
6417   }
6418   { __enumext_set_error:n {#1} }
6419 }
6420 \cs_new_protected:Nn __enumext_set_error:n
6421 { \msg_error:nnn { enumext } { invalid-level } {#1} }

```

(End of definition for `\__enumext_set_parse:n` and `\__enumext_set_error:n`.)

`\setenumext` Defines the user command `\setenumext`. It accepts an optional argument in ‘#1’ that selects the environments and levels, with default value `enumext`, `1`, and a mandatory [`(key = val)`] argument in ‘#2’. The optional argument is split into the families to be modified and the `(keys)` are passed to each one of them.

```

6422 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
6423 {
6424   \seq_clear:N \l__enumext_setkey_tmpa_seq
6425   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
6426   \int_set:Nn \l__enumext_setkey_tmpa_int
6427   {
6428     \seq_count:N \l__enumext_setkey_tmpb_seq
6429   }
6430   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
6431   {
6432     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
6433     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
6434     \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
6435     {
6436       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
6437     }
6438   }
6439   {
6440     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
6441   }
6442   \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq
6443   {
6444     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
6445   }
6446 }

```

(End of definition for `\setenumext`. This function is documented on page 7.)

13.53 The command `\setenumextmeta`

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Simplify syntax for command that adds .meta key to existing keys \(l3keys\)](#).

13.53.1 Definition of variables for `\setenumextmeta`

`\l__enumext_meta_path_str` The string `\l__enumext_meta_path_str` is used by the `\setenumextmeta` command (§13.53).

```

6447 \str_new:N \l__enumext_meta_path_str

```

(End of definition for `\l__enumext_meta_path_str`.)

13.53.2 Definition of error messages for `\setenumextmeta`

already-defined
prohibited-unknown

Messages used by `\setenumextmeta` command.

```

6448 \msg_new:nnnn { enumext } { already-defined }
6449 {
6450   The~key~'##1'~is~already~defined~\msg_line_context:.
6451 }
6452 {
6453   Choose~another~name~for~the~meta~key.
6454 }
6455 \msg_new:nnnn { enumext } { prohibited-unknown }
6456 {
6457   The~name~'unknown'~can't~be~chosen~for~a~meta~key~\msg_line_context:.
6458 }
6459 {
6460   'unknown'~is~reserved~by~the~package;~choose~another~name.
6461 }

```

(End of definition for `already-defined` and `prohibited-unknown`.)

13.53.3 Definition of keys for `\setenumextmeta`

`enumext`
`enumext*` These `(keys)` are the selectors that can be passed in the *optional argument* of `\setenumextmeta` (§13.53) and `\setenumext` (§13.52). First, the selectors for each level, `1`, `2`, `3` and `4`, of type `.code:n` are created for “all levels” of the `enumext` environment.

```

6462 \int_step_inline:nn { 4 }

```

```

6463 {
6464   \keys_define:nn { enumext }
6465   {
6466     #1 .code:n = { \str_set:Nn \l__enumext_meta_path_str { level-#1 } },
6467     #1 .value_forbidden:n = true,
6468   }
6469 }

```

And now the *(keys)* are defined for the environments using `.code:n` for the `enumext` environment and `.meta:n` for the `enumext*` environment.

```

6470 \clist_map_inline:nn { enumext }
6471 {
6472   \keys_define:nn { enumext }
6473   {
6474     % ignored for now, might do something useful in the future
6475     #1 .code:n = { },
6476     #1 .value_forbidden:n = true,
6477     #1* .code:n = { \str_set:Nn \l__enumext_meta_path_str { #1* } },
6478     #1* .value_forbidden:n = true,
6479   }
6480 }
6481 \keys_define:nn { enumext }
6482 {
6483   * .meta:n = enumext*,
6484   * .value_forbidden:n = true,
6485 }

```

(End of definition for `enumext` and `enumext*`.)

13.53.4 Internal functions for `\setenumextmeta`

The internal functions `\__enumext_key_set_meta:nnn` and `\__enumext_key_def_meta:nnn` will check the *optional argument* and create the “*meta-key*”.

```

6486 \cs_new_protected:Npn \__enumext_key_set_meta:nnn #1
6487 {
6488   \keys_set:nn { enumext } {#1}
6489   \__enumext_key_def_meta:Vnn \l__enumext_meta_path_str
6490 }
6491 \cs_new_protected:Nn \__enumext_key_def_meta:nnn
6492 {
6493   \bool_lazy_or:nnTF
6494   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
6495   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
6496   { \msg_error:nnn { enumext } { already-defined } {#2} }
6497   {
6498     \keys_define:nn { enumext / #1 }
6499     {
6500       #2 .meta:n = {#3},
6501       #2 .value_forbidden:n = true
6502     }
6503   }
6504 }
6505 \cs_generate_variant:Nn \__enumext_key_def_meta:nnn { V }

```

(End of definition for `\__enumext_key_set_meta:nnn` and `\__enumext_key_def_meta:nnn`.)

`\setenumextmeta`

Now the user command is created, taking care that unknown cannot be passed as an argument. It accepts a *starred argument* ‘`*`’ in ‘`#1`’, an optional argument in ‘`#2`’ that selects the environments and levels, with default value `enumext,1`, the name of the new “*meta-key*” in ‘`#3`’ and the *(keys)* it sets in ‘`#4`’. With the *starred argument* the “*meta-key*” is defined for the four levels of `enumext` and for `enumext*`.

```

6506 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
6507 {
6508   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
6509   { \msg_error:nn { enumext } { prohibited-unknown } }
6510   {
6511     \bool_if:nTF {#1}
6512     {
6513       \int_step_inline:nn { 4 }
6514       { \__enumext_key_set_meta:nnn { enumext, ##1 } {#3} {#4} }
6515       \__enumext_key_set_meta:nnn { enumext* } {#3} {#4}

```

```

6516     }
6517     { \__enumext_key_set_meta:nnn {#2} {#3} {#4} }
6518   }
6519 }

```

(End of definition for `\setenumextmeta`. This function is documented on page 7.)

13.54 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function](#).

13.54.1 Definition of variables for `\foreachkeyans`

Internal variables used by the `\foreachkeyans` command (§13.54).

```

\__enumext_foreach_print_seq
\__enumext_foreach_name_prop_tl
\__enumext_foreach_default_keys_tl
6520 \seq_new:N \__enumext_foreach_print_seq
6521 \tl_new:N \__enumext_foreach_name_prop_tl
6522 \tl_new:N \__enumext_foreach_default_keys_tl

```

(End of definition for `\__enumext_foreach_print_seq`, `\__enumext_foreach_name_prop_tl`, and `\__enumext_foreach_default_keys_tl`.)

13.54.2 Definition of error messages for `\foreachkeyans`

Message used by unknown *keys* in `\foreachkeyans`.

```

for-key-unknown
for-key-value-unknown
6523 \msg_new:nnnn { enumext } { for-key-unknown }
6524 {
6525   The~key~'#1'~is~unknown~by~'\token_to_str:N \foreachkeyans'~and~is~being~
6526   ignored~\msg_line_context:.
6527 }
6528 {
6529   The~command~'\token_to_str:N \foreachkeyans'~does~not~have~a~key~called~
6530   '#1'.~Check~that~you~have~spelled~the~key~name~correctly.
6531 }
6532 \msg_new:nnnn { enumext } { for-key-value-unknown }
6533 {
6534   The~key~'#1=#2'~is~unknown~by~'\token_to_str:N \foreachkeyans'~and~is~
6535   being~ignored~\msg_line_context:.
6536 }
6537 {
6538   The~command~'\token_to_str:N \foreachkeyans'~does~not~have~a~key~called~
6539   '#1'.~Check~that~you~have~spelled~the~key~name~correctly.
6540 }

```

(End of definition for `for-key-unknown` and `for-key-value-unknown`.)

13.54.3 Definition of keys for `\foreachkeyans`

A set of *keys* is defined for the command and the default values of these are saved in `\__enumext_foreach_default_keys_tl` to avoid the use of a group.

```

before
after
sep
start
step
stop
wrapper
unknown
6541 \keys_define:nn { enumext / foreach }
6542 {
6543   before .tl_set:N = \__enumext_foreach_before_tl,
6544   before .value_required:n = true,
6545   after .tl_set:N = \__enumext_foreach_after_tl,
6546   after .value_required:n = true,
6547   start .int_set:N = \__enumext_foreach_start_int,
6548   start .value_required:n = true,
6549   stop .int_set:N = \__enumext_foreach_stop_int,
6550   stop .value_required:n = true,
6551   step .int_set:N = \__enumext_foreach_step_int,
6552   step .value_required:n = true,
6553   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
6554   wrapper .value_required:n = true,
6555   sep .tl_set:N = \__enumext_foreach_sep_tl,
6556   sep .value_required:n = true,
6557   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
6558 }
6559 \keys_precompile:nnN { enumext / foreach }
6560 {
6561   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }

```

```

6562   }
6563   \l__enumext_foreach_default_keys_tl

```

(End of definition for before and others.)

13.54.4 Internal functions for \foreachkeyans

Functions for handling unknown $\langle keys \rangle$.

```

\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n

```

```

6564 \cs_new_protected:Nn \__enumext_parse_foreach_keys:nn
6565 {
6566   \tl_if_blank:nTF {#2}
6567   {
6568     \msg_error:nnn { enumext } { for-key-unknown } {#1}
6569   }
6570   {
6571     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
6572   }
6573 }
6574 \cs_new_protected:Nn \__enumext_parse_foreach_keys:n
6575 {
6576   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
6577 }

```

(End of definition for __enumext_parse_foreach_keys:nn and __enumext_parse_foreach_keys:n.)

```

\__enumext_foreach_keyans:nn
\__enumext_foreach_add_body:n

```

The internal functions __enumext_foreach_keyans:nn and __enumext_foreach_add_body:n will loop through the prop list and print the contents.

```

6578 \cs_new_protected:Nn \__enumext_foreach_keyans:nn
6579 {
6580   \tl_use:N \l__enumext_foreach_default_keys_tl
6581   \keys_set:nn { enumext / foreach } {#1}
6582   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
6583   \prop_if_exist:cF { g__enumext_ #2 _prop }
6584   {
6585     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
6586   }
6587   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
6588   {
6589     \int_set:Nn \l__enumext_foreach_stop_int
6590     { \prop_count:c { g__enumext_ #2 _prop } }
6591   }
6592   \seq_clear:N \l__enumext_foreach_print_seq
6593   \int_step_function:nnnN
6594   { \l__enumext_foreach_start_int }
6595   { \l__enumext_foreach_step_int }
6596   { \l__enumext_foreach_stop_int }
6597   \__enumext_foreach_add_body:n
6598   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
6599 }
6600 \cs_new_protected:Nn \__enumext_foreach_add_body:n
6601 {
6602   \seq_put_right:Ne \l__enumext_foreach_print_seq
6603   {
6604     \exp_not:V \l__enumext_foreach_before_tl
6605     \__enumext_foreach_wrapper:n
6606     {
6607       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop } {#1}
6608     }
6609     \exp_not:V \l__enumext_foreach_after_tl
6610   }
6611 }

```

(End of definition for __enumext_foreach_keyans:nn and __enumext_foreach_add_body:n.)

\foreachkeyans

The command accepts an optional $[\langle key = val \rangle]$ argument in ‘#1’ and a mandatory $\{\langle store name \rangle\}$ in ‘#2’, and runs the loop over the content stored in it.

```

6612 \NewDocumentCommand \foreachkeyans { +0{} m }
6613 {
6614   \__enumext_foreach_keyans:nn {#1} {#2}
6615 }

```

(End of definition for \foreachkeyans. This function is documented on page 21.)

13.55 General messages

Messages shared by several sections of the implementation; the messages used by only one command, key or environment are defined in its own section.

package-load
package-not-load

Messages used when checking the `multicol` package in §13.3 and the `hyperref` and `footnotehyper` packages in §13.7.

```
6616 \msg_new:nnn { enumext } { package-load }
6617 {
6618   The~'#1'~package~is~already~loaded.
6619 }
6620 \msg_new:nnn { enumext } { package-not-load }
6621 {
6622   The~'#1'~package~will~be~loaded~as~a~dependency.
6623 }
```

(End of definition for `package-load` and `package-not-load`.)

unknown-choice

Message used by the keys `align` in §13.13, `mark-pos` in §13.31 and `label-pos` in §13.46.3.

```
6624 \msg_new:nnnn { enumext } { unknown-choice }
6625 {
6626   Invalid~value~'#3'~for~key~'#1';~use~one~of:~#2~\msg_line_context:.
6627 }
6628 {
6629   Use~one~of~the~listed~values.
6630 }
```

(End of definition for `unknown-choice`.)

command-wrong-place

Message used by the commands `\anskey` in §13.34 and `\anspic` in §13.46.4 when they are used in the wrong place.

```
6631 \msg_new:nnnn { enumext } { command-wrong-place }
6632 {
6633   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.
6634 }
6635 {
6636   '\c_backslash_str #1'~works~outside~the~environment~'#2'.
6637 }
```

(End of definition for `command-wrong-place`.)

undefined-storage-anskey

Message used by the commands `\getkeyans` in §13.50, `\printkeyans` in §13.51 and `\foreachkeyans` in §13.54 when the storage is not defined.

```
6638 \msg_new:nnnn { enumext } { undefined-storage-anskey }
6639 {
6640   Storage~named~'#1'~is~not~defined~\msg_line_context:.
6641 }
6642 {
6643   Store~content~first~with~the~key~'save-ans=#1'.
6644 }
```

(End of definition for `undefined-storage-anskey`.)

missing-miniright

Message used by `enumext` in §13.43 and `keyans` in §13.44 when the key `mini-env` needs `\miniright` from §13.25.7.

```
6645 \msg_new:nnn { enumext } { missing-miniright }
6646 {
6647   Missing~'\token_to_str:N \miniright'~\msg_line_context:.\
6648   The~key~'mini-env'~needs~'\token_to_str:N \miniright'.
6649 }
```

(End of definition for `missing-miniright`.)

nested

Message used when an environment is nested in itself, raised by `enumext*` in §13.48 and `keyans*` in §13.49.

```
6650 \msg_new:nnnn { enumext } { nested }
6651 {
6652   The~environment~'\l__enumext_envir_name_tl \c_space_tl'~can't~be~nested~
6653   \msg_line_context:.
```

```
6654     }
6655     {
6656         Starred~environments~are~horizontal~lists~and~can't~be~nested.
6657     }
```

(End of definition for nested.)

13.56 Finish package

Finish package implementation.

```
6658 \file_input_stop:
6659 </package>
```

14 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

A	
above	<u>1643</u>
above*	<u>1643</u>
after	<u>1168</u> , <u>6541</u>
align	<u>652</u>
already-defined	<u>6448</u>
\anskey	<u>16</u> , <u>95</u> , <u>97</u> , <u>3200</u>
anskey*	<u>16</u> , <u>3402</u>
anskey-cmd-key-unknown	<u>3093</u>
anskey-cmd-key-value-unknown	<u>3093</u>
anskey-empty-arg	<u>2960</u>
anskey-env-error	<u>3228</u>
anskey-env-key-unknown	<u>3237</u>
anskey-env-key-value-unknown	<u>3237</u>
anskey-env-wrong	<u>3237</u>
anskey-math-mode	<u>2960</u>
anskey-nested	<u>3093</u>
anskey-unnumber-item	<u>2960</u>
anskey-wrong-place	<u>3093</u>
\anspic	<u>19</u> , <u>131</u> , <u>135</u> , <u>5056</u>
\anspic*	<u>87</u>
B	
base-fix	<u>1004</u>
before	<u>1168</u> , <u>6541</u>
before*	<u>1168</u>
beginpenalty	<u>946</u>
below	<u>1643</u>
below*	<u>1643</u>
break-col	<u>3126</u> , <u>3275</u>
C	
c@ internal commands:	
\c@__enumext_resume_i_int	<u>575</u>
\c@__enumext_resume_ii_int	<u>575</u>
\c@__enumext_resume_iii_int	<u>575</u>
\c@__enumext_resume_iv_int	<u>575</u>
\c@__enumext_resume_vii_int	<u>575</u>
check-ans	<u>2413</u>
Document class :	
article	<u>54</u>
l3doc	<u>33</u>
columns	<u>1239</u>
columns-sep	<u>1239</u>
command-wrong-place	<u>6631</u>
Commands provided by enumext:	
\anskey	<u>36</u>
\anspic*	<u>36</u>
\anspic	<u>132</u>
\foreachkeyans	<u>168</u>
\getkeyans	<u>159</u>
\item*	<u>36</u>
\item	<u>110</u>
\miniright	<u>35</u>
\printkeyans*	<u>160</u>
\printkeyans	<u>36</u> , <u>160</u>
\resetenumext	<u>79</u> , <u>80</u>
\setenumextmeta	<u>166</u>
\setenumext	<u>160</u> , <u>166</u>
Counters defined by enumext:	
enumXiii	<u>45</u>
enumXii	<u>45</u>
enumXiv	<u>45</u>
enumXi	<u>45</u>
enumXviii	<u>45</u>
enumXvii	<u>45</u>
enumXvi	<u>45</u>
enumXv	<u>45</u>
counters	<u>549</u>
E	
endpenalty	<u>946</u>
enumext	<u>5</u> , <u>4422</u> , <u>6462</u>
enumext internal commands:	
__enumext_add_pre_parsep:	<u>1283</u>
__enumext_after_args_exec:	<u>1183</u>
__enumext_after_args_exec_v:	<u>1199</u>
__enumext_after_env:nn	<u>157</u>
__enumext_after_hyperref:	<u>340</u>
__enumext_after_list_vii:	<u>5469</u>
__enumext_after_list_viii:	<u>5823</u>
__enumext_after_stop_list:	<u>1183</u>
__enumext_after_stop_list_v:	<u>1199</u>
__enumext_after_stop_list_vii:	<u>1215</u>
__enumext_after_stop_list_viii:	<u>1215</u>
\l_enumext_align_label_pos_X_str	<u>53</u>
\l_enumext_align_label_X_str	<u>87</u>
\c_enumext_all_envs_clist	<u>135</u>
__enumext_anskey_env_file_if_	
writable:n	<u>3317</u>
__enumext_anskey_env_file_if_	
writable:nTF	<u>3317</u>
__enumext_anskey_env_file_write:nn	<u>3317</u>
\c_enumext_anskey_env_hidden_space_	
str	<u>3225</u>
__enumext_anskey_env_safe_inner:	<u>3351</u>
__enumext_anskey_env_safe_outer:	<u>3351</u>
__enumext_anskey_env_unknown:n	<u>3303</u>
__enumext_anskey_env_unknown:nn	<u>3303</u>
\l_enumext_anskey_level_int	<u>3092</u>
__enumext_anskey_safe_inner:	<u>3163</u>
__enumext_anskey_safe_outer:	<u>3163</u>
__enumext_anskey_show_wrap_arg:n	<u>3045</u>
__enumext_anskey_show_wrap_left:n	<u>3070</u>
__enumext_anskey_unknown:n	<u>3142</u>
__enumext_anskey_unknown:nn	<u>3142</u>
\l_enumext_anspic_above_int	<u>4737</u>
__enumext_anspic_args:nnn	<u>4927</u>
\l_enumext_anspic_args_seq	<u>4737</u>
\l_enumext_anspic_below_int	<u>4737</u>
\l_enumext_anspic_body_box	<u>4737</u>
__enumext_anspic_body_dim:n	<u>4927</u>
\l_enumext_anspic_body_htdp_dim	<u>4737</u>
__enumext_anspic_exec:	<u>4927</u>

_enumext_anspic_label:nn	4927	_enumext_define_counter:Nn	558
\l_enumext_anspic_label_above_bool	4737	_enumext_endminipage:	329
\l_enumext_anspic_label_box	4737	\g_enumext_envir_name_tl	30
\l_enumext_anspic_label_htdp_dim	4737	\l_enumext_envir_name_tl	30
_enumext_anspic_label_pos:nnn	4927	_enumext_execute_after_env:	3421
\l_enumext_anspic_mini_pos_str	4737	_enumext_fake_item_indent:	1095
\l_enumext_anspic_mini_width_dim	4737	_enumext_fake_item_indent_vii:	1095
_enumext_anspic_print:n	4927	_enumext_fake_item_indent_viii:	1095
_enumext_anspic_row:n	4927	\l_enumext_fake_item_indent_X_tl	47
_enumext_anspic_start_list_tag:	4687	_enumext_fake_make_label_vii:n	5572
_enumext_anspic_stop_list_tag:	4687	_enumext_fake_make_label_viii:n	5939
_enumext_anspic_stop_start_list_tag:	4687	_enumext_filter_level:n	6383
_enumext_at_begin_document:n	153	_enumext_filter_level_key:n	6383
\l_enumext_base_depth_int	5618	_enumext_filter_level_pair:nn	6383
_enumext_before_args_exec:	1183	_enumext_filter_save_key:n	2754
_enumext_before_args_exec_v:	1199	_enumext_filter_save_key_key:n	2754
_enumext_before_args_exec_vii:	1215	_enumext_filter_save_key_pair:nn	2754
_enumext_before_args_exec_viii:	1215	_enumext_filter_series:n	1965
_enumext_before_keys_exec:	1183	_enumext_filter_series_key:n	1965
_enumext_before_keys_exec_v:	1199	_enumext_filter_series_pair:nn	1965
_enumext_before_keys_exec_vii:	1215	_enumext_first_item_tmp_vii:	5501
_enumext_before_keys_exec_viii:	1215	_enumext_first_item_tmp_viii:	5829
_enumext_before_list:	4360	\g_enumext_footnote_standard_arg_seq	389
_enumext_before_list_v:	4477	\g_enumext_footnote_standard_int	389
_enumext_before_list_vii:	5462	\g_enumext_footnote_standard_int_seq	389
_enumext_before_list_viii:	5817	\g_enumext_footnote_starred_arg_seq	389
_enumext_calc_hspace:NNNNNNN	4103	\g_enumext_footnote_starred_int	389
_enumext_check_ans_active:	2433	\g_enumext_footnote_starred_int_seq	389
\g_enumext_check_ans_key_bool	106	\l_enumext_footnotes_key_bool	125
_enumext_check_ans_key_hook:	2491	_enumext_footnotetext:nn	395
_enumext_check_ans_level:	2433	_enumext_foreach_add_body:n	6578
_enumext_check_ans_log:	2538	\l_enumext_foreach_default_keys_tl	6520
_enumext_check_ans_log_msg_greater:	2538	_enumext_foreach_keyans:nn	6578
_enumext_check_ans_log_msg_less:	2538	\l_enumext_foreach_name_prop_tl	6520
_enumext_check_ans_log_msg_same_ok:	2538	\l_enumext_foreach_print_seq	6520
_enumext_check_ans_msg_greater:	2513	_enumext_getkeyans:nn	6096
_enumext_check_ans_msg_less:	2513	_enumext_getkeyans_aux:n	6096
_enumext_check_ans_msg_same_ok:	2513	\l_enumext_hyperref_bool	125
_enumext_check_ans_show:	2513	_enumext_hypertarget:nn	340
\l_enumext_check_answers_bool	106	_enumext_if_is_int:nTF	165
_enumext_check_starred_cmd:n	2562	_enumext_in_keyans:	3156
\g_enumext_check_starred_cmd_int	106	_enumext_internal_mini_page:	525
\l_enumext_check_start_line_env_tl	106	_enumext_is_not_nested:	203
\l_enumext_counter_i_tl	544	_enumext_is_on_first_level:	203
\l_enumext_counter_ii_tl	544	\g_enumext_item_anskey_int	106
\l_enumext_counter_iii_tl	544	_enumext_item_answer_diff:	2506
\l_enumext_counter_iv_tl	544	\g_enumext_item_answer_diff_int	106
\g_enumext_counter_styles_tl	584	\l_enumext_item_break_X_bool	87
\l_enumext_counter_v_tl	544	\l_enumext_item_column_pos_X_int	87
\l_enumext_counter_vi_tl	544	\g_enumext_item_count_all_X_int	87
\l_enumext_counter_vii_tl	544	\l_enumext_item_number_bool	106
\l_enumext_counter_viii_tl	544	\g_enumext_item_number_int	106
\l_enumext_current_widest_dim	40	_enumext_item_peek_args_vii:	5509
_enumext_default_item:n	3682	_enumext_item_peek_args_viii:	5837
\c_enumext_default_label_X_tl	6124	_enumext_item_starred_exec:	3701
_enumext_default_restore:	6226	_enumext_item_starred_exec:nn	3701
_enumext_default_restore:nnn	6226	\l_enumext_item_starred_X_bool	87
_enumext_default_restore_env:nn	6226	_enumext_item_std:w	329
_enumext_default_save:nnn	6226	_enumext_item_switch:N	5618
_enumext_default_save_env:nn	6226	_enumext_item_switch_vii:	5618
\c_enumext_default_vars_clist	6124		

<code>__enumext_item_switch_viii:</code>	5618	<code>\l__enumext_label_fill_right_X_tl</code>	53
<code>\g__enumext_item_symbol_aux_tl</code>	3681	<code>\l__enumext_label_i_tl</code>	731
<code>\g__enumext_item_symbol_aux_X_tl</code>	87	<code>\l__enumext_label_ii_tl</code>	731
<code>\l__enumext_item_text_X_box</code>	87	<code>\l__enumext_label_iii_tl</code>	731
<code>\l__enumext_item_width_X_dim</code>	5075	<code>\l__enumext_label_iv_tl</code>	731
<code>\l__enumext_item_wrap_key_bool</code>	106	<code>__enumext_label_style:Nnn</code>	603
<code>\l__enumext_itemindent_X_dim</code>	4061	<code>\l__enumext_label_v_tl</code>	847
<code>\l__enumext_joined_item_aux_X_int</code>	5075	<code>\l__enumext_label_vi_tl</code>	847
<code>__enumext_joined_item_vii:w</code>	5509	<code>\l__enumext_label_vii_tl</code>	776
<code>__enumext_joined_item_viii:w</code>	5837	<code>\l__enumext_label_viii_tl</code>	776
<code>\l__enumext_joined_item_X_int</code>	5075	<code>\l__enumext_label_width_by_box</code>	584
<code>\l__enumext_joined_width_X_dim</code>	87	<code>__enumext_label_width_by_box:Nn</code>	597
<code>__enumext_key_def_meta:nnn</code>	6486	<code>\l__enumext_leftmargin_tmp_X_bool</code>	66
<code>__enumext_key_set_meta:nnn</code>	6486	<code>\l__enumext_leftmargin_tmp_X_dim</code>	66
<code>__enumext_keyans_addto_prop:n</code>	3457	<code>\l__enumext_leftmargin_X_dim</code>	66
<code>__enumext_keyans_addto_seq:n</code>	3531	<code>__enumext_level:</code>	161
<code>__enumext_keyans_addto_seq_link:</code>	3531	<code>\l__enumext_level_h_int</code>	25
<code>__enumext_keyans_default_item:n</code>	3958	<code>\l__enumext_level_int</code>	25
<code>\l__enumext_keyans_env_bool</code>	30	<code>__enumext_list_arg_two_i:</code>	4135
<code>__enumext_keyans_fake_item_indent:</code> ...	1095	<code>__enumext_list_arg_two_ii:</code>	4135
<code>\l__enumext_keyans_level_h_int</code>	25	<code>__enumext_list_arg_two_iii:</code>	4135
<code>\l__enumext_keyans_level_int</code>	25	<code>__enumext_list_arg_two_iv:</code>	4135
<code>__enumext_keyans_make_label:</code>	3999	<code>__enumext_list_arg_two_v:</code>	4135
<code>__enumext_keyans_make_label_box:</code>	3999	<code>__enumext_list_arg_two_vii:</code>	4181
<code>__enumext_keyans_make_label_std:</code>	3999	<code>__enumext_list_arg_two_viii:</code>	4181
<code>__enumext_keyans_mini_right_cmd:n</code> ...	1609	<code>__enumext_log_answer_vars:</code>	314
<code>__enumext_keyans_minipage_add_space:</code> .	1493	<code>__enumext_log_global_vars:</code>	314
<code>__enumext_keyans_minipage_set_skip:</code> ..	1493	<code>__enumext_make_label:</code>	3751
<code>__enumext_keyans_multi_addvspace:</code> ...	1336	<code>__enumext_make_label_box:</code>	3751
<code>__enumext_keyans_multi_set_vskip:</code> ...	1336	<code>__enumext_make_label_std:</code>	3751
<code>__enumext_keyans_multicols_start:</code> ...	4477	<code>\l__enumext_mark_position_str</code>	115
<code>__enumext_keyans_multicols_stop:</code>	4477	<code>\l__enumext_mark_position_v_str</code>	115
<code>__enumext_keyans_name_and_start:</code>	260	<code>\l__enumext_mark_position_viii_str</code> ...	115
<code>__enumext_keyans_parse_keys:n</code>	4473	<code>\l__enumext_mark_sep_tmpa_dim</code>	3440
<code>__enumext_keyans_pic_arg_two:</code>	4823	<code>\l__enumext_mark_sep_tmpb_dim</code>	3440
<code>\l__enumext_keyans_pic_level_int</code>	25	<code>\l__enumext_meta_path_str</code>	6447
<code>__enumext_keyans_pic_parse_keys:n</code> ...	4823	<code>__enumext_mini_addvspace_vii:</code>	1536
<code>__enumext_keyans_pic_safe_exec:</code>	4823	<code>__enumext_mini_addvspace_viii:</code>	1536
<code>__enumext_keyans_pic_skip_abs:N</code>	4823	<code>__enumext_mini_apply_end:</code>	1421
<code>__enumext_keyans_pos_mark_set:</code>	3607	<code>__enumext_mini_baselineskip:</code>	1407
<code>__enumext_keyans_redefine_item:</code>	3981	<code>__enumext_mini_capture_end:</code>	1421
<code>__enumext_keyans_ref:</code>	863	<code>__enumext_mini_capture_multicols:</code> ...	1421
<code>__enumext_keyans_ref:n</code>	863	<code>\g__enumext_mini_dp_dim</code>	1368
<code>__enumext_keyans_safe_exec:</code>	4452	<code>\g__enumext_mini_end_bool</code>	1368
<code>__enumext_keyans_save_item_opt:n</code>	3574	<code>\g__enumext_mini_end_skip</code>	1368
<code>__enumext_keyans_set_item_width:</code>	4561	<code>\l__enumext_mini_gap_dim</code>	1368
<code>__enumext_keyans_show_ans:</code>	3607	<code>\l__enumext_mini_last_box</code>	1368
<code>__enumext_keyans_show_item_opt:</code>	3574	<code>__enumext_mini_page</code>	525
<code>__enumext_keyans_show_item_opt_viii:</code> .	3574	<code>\l__enumext_mini_prev_dim</code>	1368
<code>__enumext_keyans_show_pos:</code>	3607	<code>__enumext_mini_right_cmd:n</code>	1579
<code>__enumext_keyans_starred_item:n</code>	3970	<code>__enumext_mini_set_vskip_vii:</code>	1528
<code>__enumext_keyans_starred_item_star:</code> ..	5873	<code>__enumext_mini_set_vskip_viii:</code>	1528
<code>__enumext_keyans_store_ref:</code>	3478	<code>__enumext_minipage:w</code>	329
<code>__enumext_keyans_store_ref_aux_i:</code> ...	3478	<code>\g__enumext_minipage_active_X_bool</code> ...	5075
<code>__enumext_keyans_store_ref_aux_ii:</code> ...	3478	<code>\l__enumext_minipage_active_X_bool</code>	79
<code>__enumext_keyans_unknown_keys:n</code>	3883	<code>__enumext_minipage_add_space:</code>	1457
<code>__enumext_keyans_unknown_keys:nn</code>	3883	<code>\g__enumext_minipage_center_X_bool</code>	87
<code>__enumext_keyans_wrapper_label:n</code>	3999	<code>\l__enumext_minipage_left_X_dim</code>	79
<code>__enumext_keyans_wrapper_label_viii:n</code>	5939	<code>\g__enumext_minipage_right_skip</code>	79
<code>\l__enumext_label_copy_X_tl</code>	127	<code>\l__enumext_minipage_right_skip</code>	1368
<code>\l__enumext_label_fill_left_X_tl</code>	53	<code>\g__enumext_minipage_right_X_dim</code>	87

<code>__enumext_minipage_set_skip:</code>	1457	<code>\g__enumext_resume_last_keys_X_tl</code>	1760
<code>\g__enumext_minipage_stat_int</code>	79	<code>__enumext_resume_series:n</code>	2097
<code>\l__enumext_miniright_code_X_box</code>	5075	<code>\l__enumext_resume_series_X_bool</code>	1760
<code>__enumext_multi_addvspace:</code>	1319	<code>__enumext_resume_star:</code>	2162
<code>__enumext_multi_set_vskip:</code>	1271	<code>\l__enumext_resume_star_key_X_bool</code> . . .	1760
<code>\l__enumext_multicols_above_X_skip</code> . . .	73	<code>__enumext_safe_exec:</code>	4234
<code>\l__enumext_multicols_below_X_skip</code> . . .	73	<code>__enumext_safe_exec_vii:</code>	5428
<code>__enumext_multicols_start:</code>	4306	<code>__enumext_safe_exec_viii:</code>	5785
<code>__enumext_multicols_stop:</code>	4336	<code>__enumext_save_last_keys:n</code>	1992
<code>__enumext_nested_base_line_fix:</code>	1010	<code>__enumext_second_part:</code>	4382
<code>__enumext_nested_reset:</code>	5618	<code>__enumext_second_part_v:</code>	4477
<code>__enumext_nested_save:</code>	5618	<code>\l__enumext_series_name_tl</code>	1760
<code>__enumext_newlabel:nn</code>	374	<code>__enumext_set_error:n</code>	6408
<code>\l__enumext_newlabel_arg_one_tl</code>	127	<code>__enumext_set_item_width:</code>	4408
<code>\l__enumext_newlabel_arg_two_tl</code>	127	<code>__enumext_set_parse:n</code>	6408
<code>\l__enumext_outer_labelsep_dim</code>	5618	<code>\l__enumext_setkey_tmpa_int</code>	6315
<code>\l__enumext_outer_labelwidth_dim</code>	5618	<code>\l__enumext_setkey_tmpa_seq</code>	6315
<code>__enumext_parse_foreach_keys:n</code>	6564	<code>\l__enumext_setkey_tmpa_tl</code>	6315
<code>__enumext_parse_foreach_keys:nn</code>	6564	<code>\l__enumext_setkey_tmpb_seq</code>	6315
<code>__enumext_parse_keys:n</code>	4245	<code>\l__enumext_setkey_tmpb_tl</code>	6315
<code>__enumext_parse_keys_vii:n</code>	5445	<code>__enumext_show_length:nnn</code>	171
<code>__enumext_parse_keys_viii:n</code>	5810	<code>\l__enumext_show_pos_tmp_int</code>	3440
<code>__enumext_parse_save_key:n</code>	2741	<code>\g__enumext_standard_bool</code>	30
<code>__enumext_parse_save_key_vii:n</code>	2741	<code>\l__enumext_standard_bool</code>	30
<code>__enumext_parse_series:n</code>	2026	<code>\l__enumext_standard_first_bool</code>	30
<code>__enumext_phantomsection:</code>	340	<code>__enumext_standard_item_vii:w</code>	5509
<code>\c__enumext_print_envs_clist</code>	6124	<code>__enumext_standard_item_viii:w</code>	5837
<code>__enumext_print_footnote:</code>	395	<code>__enumext_standard_ref:</code>	753
<code>__enumext_print_footnote_mini:</code>	395	<code>__enumext_standard_ref:nn</code>	753
<code>__enumext_print_footnote_standard:</code> . .	459	<code>__enumext_standard_reset:nn</code>	2200
<code>__enumext_print_footnote_starred:</code> . . .	459	<code>__enumext_standard_reset_key:</code>	2273
<code>__enumext_print_keyans_box:NN</code>	2870	<code>__enumext_standard_reset_key_star:</code> . .	2273
<code>\l__enumext_print_keyans_cmd_bool</code> . . .	115	<code>__enumext_standard_save_counter:</code> . . .	1829
<code>\l__enumext_print_keyans_star_bool</code> . . .	115	<code>__enumext_standard_save_counter_aux:</code> .	1829
<code>\l__enumext_print_keyans_starred_tl</code> . .	6139	<code>__enumext_standard_unknown_keys:n</code> . .	3931
<code>\l__enumext_print_keyans_X_tl</code>	115	<code>__enumext_standard_unknown_keys:nn</code> . .	3931
<code>__enumext_printkeyans:nnn</code>	6266	<code>\g__enumext_starred_bool</code>	30
<code>__enumext_redefine_item:</code>	3740	<code>\l__enumext_starred_bool</code>	30
<code>\l__enumext_ref_key_arg_tl</code>	715	<code>__enumext_starred_columns_set_vii:</code> . .	5102
<code>\l__enumext_ref_the_count_tl</code>	715	<code>__enumext_starred_columns_set_viii:</code> . .	5102
<code>__enumext_register_default_label_wd:Nn</code>	587	<code>\l__enumext_starred_first_bool</code>	30
<code>__enumext_remove_extra_parsep_vii:</code> . .	5734	<code>__enumext_starred_item_vii:w</code>	5509
<code>__enumext_remove_extra_parsep_viii:</code> . .	6071	<code>__enumext_starred_item_vii_aux_i:w</code> . .	5509
<code>\l__enumext_renew_counter_X_tl</code>	715	<code>__enumext_starred_item_vii_aux_ii:w</code> . .	5509
<code>__enumext_renew_footnote:</code>	395	<code>__enumext_starred_item_vii_aux_iii:w</code> .	5509
<code>__enumext_renew_footnote_mini:</code>	395	<code>__enumext_starred_item_viii:w</code>	5873
<code>__enumext_renew_footnote_standard:</code> . .	459	<code>__enumext_starred_item_viii_aux_i:w</code> . .	5873
<code>__enumext_renew_footnote_starred:</code> . . .	459	<code>__enumext_starred_item_viii_aux_ii:w</code> .	5873
<code>__enumext_reset_count_resume:nn</code>	2200	<code>__enumext_starred_joined_item_vii:n</code> . .	5164
<code>__enumext_reset_count_resume_all:n</code> . .	2200	<code>__enumext_starred_joined_item_viii:n</code> .	5164
<code>__enumext_reset_count_resume_levels:n</code>	2200	<code>__enumext_starred_ref:</code>	796
<code>__enumext_reset_global_bool:</code>	290	<code>__enumext_starred_ref:n</code>	796
<code>__enumext_reset_global_int:</code>	290	<code>__enumext_starred_reset:n</code>	2200
<code>__enumext_reset_global_tl:</code>	290	<code>__enumext_starred_reset_key:</code>	2273
<code>__enumext_reset_global_vars:</code>	290	<code>__enumext_starred_save_counter:</code>	1829
<code>__enumext_resume:n</code>	2069	<code>__enumext_starred_save_counter_aux:</code> . .	1829
<code>\l__enumext_resume_count_bool</code>	41	<code>__enumext_starred_unknown_keys:n</code> . . .	3917
<code>\l__enumext_resume_count_X_bool</code>	41	<code>__enumext_starred_unknown_keys:nn</code> . .	3917
<code>__enumext_resume_counter:</code>	1916	<code>__enumext_start_from:NNn</code>	886
<code>__enumext_resume_integer_series:</code>	2097	<code>__enumext_start_item_box_vii:w</code>	5655
<code>__enumext_resume_last_counter:</code>	1992	<code>__enumext_start_item_box_viii:w</code>	5986

__enumext_vspace_above_viii:	1716
\l__enumext_vspace_b_star_X_bool	1637
__enumext_vspace_below:	1680
__enumext_vspace_below_v:	1705
__enumext_vspace_below_vii:	1738
__enumext_vspace_below_viii:	1738
__enumext_widest_from:nNNn	901
\g__enumext_widest_label_tl	584
\l__enumext_wrap_label_opt_X_bool	53
\l__enumext_wrap_label_X_bool	53
__enumext_wrapper_label_v:n	3999
__enumext_wrapper_label_viii:n	5939
\l__enumext_write_anskey_env_bool	3221
\l__enumext_write_anskey_env_file_iow	3221
\l__enumext_write_anskey_env_file_name_- tl	3221
\l__enumext_write_aux_file_tl	127
enumext*	5, 5398, 6336, 6462
enumext-1	6336
enumext-2	6336
enumext-3	6336
enumext-4	6336
enumXi	558
enumXii	558
enumXiii	558
enumXiv	558
enumXv	558
enumXvi	558
enumXvii	558
enumXviii	558
Environments provided by enumext :	
enumext*	33
enumext	125
keyans*	152
keyanspic	131, 134
keyans	128
F	
first	1168
font	629
\footnote	42
for-key-unknown	6523
for-key-value-unknown	6523
force-eol	3275
\foreachkeyans	21, 168, 6612
G	
\getkeyans	20, 159, 6119
I	
invalid-level	6320
\item	6, 59, 109, 115, 146, 149, 150, 154, 157
\item*	6, 18, 87, 3981
item-answer-hook	140
item-break	1158
item-greater-answer	2338
item-join	3126, 3275
item-joined	5085
item-joined-columns	5085
item-less-answer	2338
item-pos*	3126, 3275, 3805
item-star	3126, 3275
item-sym*	3126, 3275, 3805

<code>\itemindent</code>	118	<code>layout-sty</code>	132, 137
<code>itemindent</code>	1049	<code>layout-top</code>	132
<code>items-same-answer</code>	2338	<code>list-indent</code>	56, 134
<code>itemsep</code>	962	<code>list-offset</code>	56, 125, 128
K			
<code>key-ref-empty</code>	724	<code>listparindent</code>	56, 120, 150
<code>keyans</code>	17, 4572, 6336	<code>mark-ans*</code>	88, 91, 109
<code>keyans*</code>	17, 5754, 6336	<code>mark-ans</code>	88, 91, 97
<code>keyans-nested</code>	4444	<code>mark-pos*</code>	88, 91, 109
<code>keyans-unknown-key</code>	3863	<code>mark-pos</code>	88, 91, 170
<code>keyans-unknown-key-value</code>	3863	<code>mark-ref</code>	88, 91, 96
<code>keyans-wrong-level</code>	3443	<code>mark-sep*</code>	88, 91, 108
<code>keyanspic</code>	19, 4873	<code>mark-sep</code>	88, 91, 97, 156
<code>keyanspic-item-cmd</code>	4747	<code>mini-env</code> 43–45, 60, 67, 91, 123, 124, 141, 143, 145, 154, 170	
<code>keyanspic-nested</code>	4747	<code>mini-right*</code>	37, 60, 91, 143, 145
<code>keyanspic-wrong-level</code>	4747	<code>mini-right</code>	37, 60, 67, 91, 143, 145
Keys for <code>\anskey</code> provided by <code>enumext</code> :			
<code>break-col</code>	96, 98	<code>mini-sep</code>	60, 91, 124
<code>force-eol</code>	101	<code>mode-box</code>	48, 111, 116
<code>item-join</code>	96, 98	<code>no-store</code>	83–85, 91, 100, 104, 110
<code>item-pos*</code>	96, 98, 110, 112	<code>noitemsep</code>	54
<code>item-star</code>	96, 98	<code>nosep</code>	54, 160
<code>item-sym*</code>	96, 98, 110, 112	<code>parsep</code>	54, 120, 133, 150
<code>overwrite</code>	101	<code>partopsep</code>	54
<code>write-env</code>	101	<code>ref</code>	49, 50, 52
Keys for <code>\foreachkeyans</code> provided by <code>enumext</code> :			
<code>after</code>	58, 59, 125, 145, 154	<code>reset*</code>	75, 80, 81, 91, 165
<code>before</code>	58, 59	<code>reset</code>	75, 80, 81, 91, 165
<code>start</code>	38, 52, 53, 75, 76	<code>resume*</code> ... 46, 72, 74–76, 78, 80, 83, 91, 125, 145, 165	
Keys for <code>anskey*</code> provided by <code>enumext</code> :			
<code>break-col</code>	96, 98	<code>resume</code> 40, 46, 53, 71, 72, 74–78, 80, 83, 91, 125, 145, 165	
<code>force-eol</code>	101	<code>rightmargin</code>	56, 139
<code>item-join</code>	96, 98	<code>save-ans</code> 40, 72–75, 78, 81–83, 85, 86, 90–92, 99, 103–105, 107, 126, 137, 148, 152, 153, 156, 159, 165	
<code>item-pos*</code>	96, 98, 110, 112	<code>save-key</code>	75, 90, 91, 122, 145
<code>item-star</code>	96, 98	<code>save-ref</code>	42, 88, 91, 95, 106, 107, 116, 156
<code>item-sym*</code>	96, 98, 110, 112	<code>save-sep</code>	88, 91, 105, 155
<code>overwrite</code>	101, 102	<code>series</code>	72, 75, 76, 78, 91, 122, 125, 145, 165
<code>write-env</code>	101, 102	<code>show-ans</code>	88, 91, 93, 95, 97, 108, 109, 136, 156
Keys for environments provided by <code>enumext</code> :			
<code>above*</code>	55, 69, 70, 124, 145	<code>show-length</code>	38, 58, 118
<code>above</code>	55, 69, 70, 124, 125, 145, 153	<code>show-pos</code>	88, 91, 93, 95, 97, 108, 136, 156
<code>after</code>	58, 59, 125, 145, 154	<code>start*</code>	52, 53, 75, 76
<code>align</code>	48, 108, 111, 116, 148, 170	<code>start</code>	38, 52, 53, 75, 76
<code>base-fix</code>	55, 75, 91, 122	<code>topsep</code>	54, 55, 133
<code>before*</code>	58, 124, 125, 145, 153	<code>widest</code>	38, 50, 53
<code>before</code>	58, 59	<code>wrap-ans*</code>	88, 91, 116, 136
<code>below*</code>	69–71, 125, 145	<code>wrap-ans</code>	88, 91, 93, 97
<code>below</code>	69–71, 125, 145, 154	<code>wrap-label*</code> 48, 110, 111, 115, 116, 147, 148, 155	
<code>check-ans</code> . 37, 39, 82–84, 86, 87, 91, 104, 107, 125, 145		<code>wrap-label</code> 48, 110, 111, 115, 116, 133, 136, 147, 148, 155, 162	
<code>columns-sep</code>	60, 123, 151	<code>wrap-opt</code>	88, 91, 108, 115, 136
<code>columns</code>	60, 123, 150, 160	keys commands:	
<code>first</code>	58, 59, 125, 128, 150, 160	<code>\l_keys_key_str</code>	98, 102
<code>font</code>	48, 111, 116, 136, 148, 160	<code>\keys_precompile:nnN</code>	163, 151
<code>item-break</code>	35, 58, 130, 150, 151	L	
<code>item-pos*</code>	110, 112	<code>label</code>	731, 776, 847
<code>item-sym*</code>	110, 112	<code>label-pos</code>	4769
<code>itemindent</code>	56, 57, 110, 115, 116, 150	<code>label-sep</code>	4769
<code>itemsep</code>	54, 120, 151, 152	Labels provided by <code>enumext</code> :	
<code>label-pos</code>	132, 134, 135, 170	<code>\Alph*</code>	46
<code>label-sep</code>	132	<code>\Roman*</code>	46
<code>labelsep</code>	48, 119, 148	<code>\alph*</code>	46
<code>labelwidth</code>	47, 48, 50, 52, 53, 119, 148	<code>\arabic*</code>	46
<code>label</code>	47, 52, 53, 133, 160, 162	<code>\roman*</code>	46
<code>layout-sep</code>	132	<code>labelsep</code>	629
		<code>labelwidth</code>	629
		<code>layout-sep</code>	4769

layout-sty 4769

layout-top 4769

\leftmargin 118

list-indent 1049

list-lengths 4071

list-lengths-not-nested 4087

list-offset 1049

list-too-deep 4227

listparindent 1049

M

\makelabel 109, 115

many-miniright-used 1375

many-starred 2310

mark-ans 2626, 4769

mark-ans* 2585, 2626

mark-pos 2626, 4769

mark-pos* 2585, 2626

mark-ref 2626

mark-sep 2626, 4769

mark-sep* 2585, 2626

midpenalty 946

mini-env 1239

mini-right 1239

mini-right* 1239

mini-sep 1239

\miniright 13, 67, 1548

missing-colon 6088

missing-miniright 6645

missing-starred 2310

mode-box 619

N

nested 6650

nested-horizontal 5085

no-store 2413

noitemsep 962

nosep 962

not-writing 3263

O

out-of-range 2193

overwrite 3275

overwrite-file 3263

P

package-load 6616

package-not-load 6616

Packages :

caption 142

enumitem 46

expl3 138

footnotehyper 42

hyperref 36

indent 151

latex-lab-block 41

ltxcmd 41

ltsockets 129

multicol 33

scontents 100

shortlst 138

tagpdf 129

parsep 962

partopsep 962

print* 6149, 6336

print-* 6149, 6336

print-1 6149, 6336

print-2 6149, 6336

print-3 6149, 6336

print-4 6149, 6336

print-starred 6140

\printkeyans 21, 160, 6301

prohibited-unknown 6448

prop-seq-int-hook 140

R

ref 731, 776, 847

reset 2255

reset* 2255

\resetenumext 12, 79, 2228

resume 1787

resume* 1787

rightmargin 1049

S

save-ans 2353

save-ans-empty 2303

save-ans-log 2318

save-ans-log-hook 2318

save-key 2727

save-ref 2626

save-sep 2585, 2626, 4769

sep 6541

seq commands:

\seq_use:Nn 151

series 1787

\setenumext 7, 163, 6422

\setenumextmeta 7, 166, 6506

show-ans 2585, 2626, 4769

show-length 1148

show-pos 2585, 2626, 4769

standard-unknown-key 3841

standard-unknown-key-value 3841

starred-unknown-key 3821

starred-unknown-key-value 3821

start 917, 6541

start* 917

start-list-tags 4590, 4687

step 6541

stop 6541

stop-list-tags 4590, 4687

stop-list-tags-break 4590

stop-list-tags-par 4590

stop-start-tags 4590, 4687

stop-start-tags-break 4590

store-int 2326

store-prop 2326

store-seq 2326

T

topsep 962

U

undefined-storage-anskey 6638

unknown ... 3126, 3275, 3883, 3909, 3931, 4769, 6336, 6541

unknown-choice 6624

unknown-key-family 6320

unknown-series-standard	1768	wrap-label*	629
unknown-series-starred	1768	wrap-opt	2585 , 2626 , 4769
W			
widest	917	wrapper	6541
width-negative	4066	write-env	3275
wrap-ans	2626	writing-file	3263
wrap-ans*	2585 , 2626 , 4769	wrong-miniright-place	1375
wrap-label	629	wrong-miniright-starred	1375
		wrong-miniright-use	1375
		wrong-place	3443