enumext

ENUMERATE EXERCISE SHEETS

v2.7 2026-11-25*

©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
1.1 Description and usage 3
1.2 The concept of left margin 3
1.3 User interface 4
1.3.1 Public counters 4
1.3.2 Public dimension 4
1.3.3 Support for multicol 4
1.3.4 Support for minipage 4
1.3.5 The \label and \ref system . . . 5
1.3.6 Support for \footnote 5
2 The environments provided 5
2.1 The environment enumext 5
2.2 The environment enumext* 5
2.3 The command \item* 6
2.3.1 Keys for \item* 6
2.4 The command \item in enumext* 6
2.4.1 Items on the whole row: the item-break key 7
3 The command \setenumext 7
4 The command \setenumextmeta 7
5 The keyval system 8
5.1 Keys for label and ref 8
5.2 The command \AddEnumextCounter . 9
5.3 Keys for penalties 9
5.4 Keys for spaces 10
5.4.1 Vertical spaces 10
5.4.2 Horizontal spaces 11
5.5 Keys for add code 11
5.6 Keys for start, series and resume . . 12
5.7 Keys for reset 13
5.7.1 The command \resetenumext . . 13
5.8 Keys for multicols 13
5.9 Keys for minipage 13
5.9.1 The command \miniright 13
5.9.2 The key mini-right 14
6 The storage system 14
6.1 Keys for storage system 14
6.1.1 Keys for label and ref 15
6.1.2 Keys for wrap and marks 15
6.1.3 Keys for debug and checking 16
6.2 The command \anskey 16
6.2.1 Keys for \anskey 16
6.3 The environment anskey* 17
6.3.1 Keys for anskey* 17
6.4 The environment keyans 18
6.4.1 The \item* in keyans 18
6.5 The environment keyanspic 19
6.5.1 Keys for keyanspic 19
6.5.2 The command \anspic 20
6.6 Printing stored content 21
6.6.1 The command \getkeyans 21
6.6.2 The command \foreachkeyans . 21
6.6.3 The command \printkeyans . . . 22
7 Full examples 23
8 Tagged PDF examples 26
9 The way of non-enumerated lists 26
10 References 29
11 Change history 30
12 Index of Documentation 32
13 Implementation 34
14 Index of Implementation 178

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 LaTeX team for their great work and to the different members of the 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 v2.7, last revised 2026-11-25.

†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 (lppl), version 1.3 or later (https://www.latex-project.org/lppl.txt). The software has the status "maintained".

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

The minimum requirement is LATEX release 2026-11-01.

1 Introduction

In the LATEX 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) LATEX2e 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) LATEX2e 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
   B) correct
   C) value
   D) value

2. Second type of questions

   I. $2\alpha + 2\delta = 90°$
   II. $\alpha = \delta$
   III. $\angle EDF = 45°$

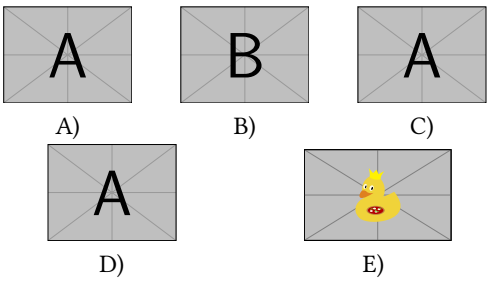
   A) I only
   B) II only
   C) I and II only
   D) I and III only
   E) I, II, and III

⋆ 3. Third type of questions

   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$

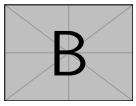
   A) value
   B) value
   C) value
   D) value
   E) value

4. Question with image and label below:



5. Question with image on right side:

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



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 LaTeX[4], 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`[5], 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 TeX Live and 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`[5] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

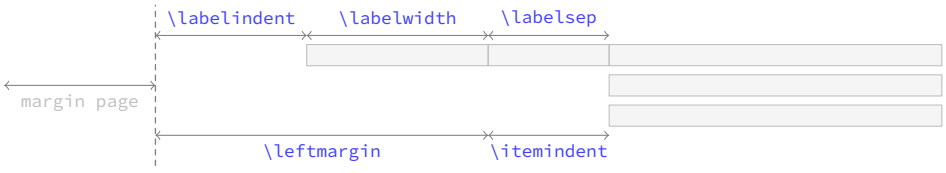


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

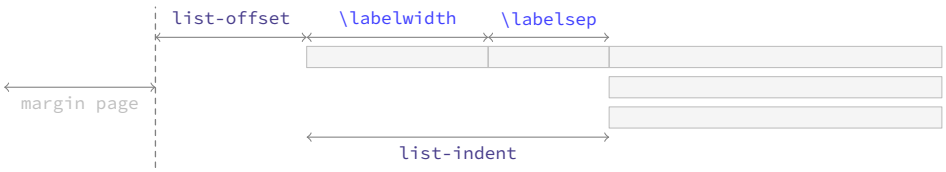


Figure 2: Representation of horizontal lengths concept in enumext.

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.

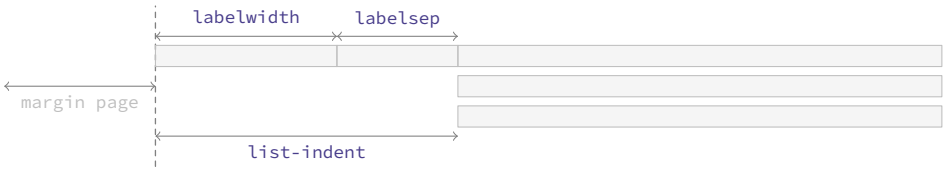


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in enumext.

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.

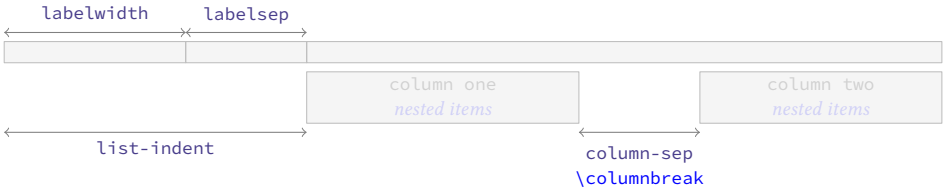


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 (§5.8).

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.

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` (§5.9).



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

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 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). Version 2.0 (2026/11/12) also supports *tagged* PDF. Load it as follows:

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

With *tagged* PDF active and `footnotehyper` v2.0 or later loaded, the package enumext uses it for the notes. If `footnotehyper` is not loaded, or it is an older version that is not compatible with *tagged* PDF, the package redefines `\footnote` internally and the "links" are not produced; in `enumext*` and `keyans*` the notes are processed inside the item and printed after the environment, which avoids the kernel issue 1337 (a `\footnotetext` after a list gives a `tagpdf` error). With an older version, load `footnotehyper` only when *tagged* PDF is not active, using `\IfDocumentMetadataF`.

2 The environments provided

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

`enumext`

```
\begin{enumext}[⟨keyval list⟩]
  \item ⟨item content⟩
  \item [⟨custom⟩] ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩] ⟨item content⟩
\end{enumext}
```

`enumext*`

```
\begin{enumext*}[⟨keyval list⟩]
  \item ⟨item content⟩
  \item [⟨custom⟩] ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩] ⟨item content⟩
\end{enumext*}
```

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:

©2024–2026 by Pablo González L

- 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 the links you must load footnotehyper 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 (§13.49.2).

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*[⟨*symbol*⟩][⟨*offset*⟩]

The \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩] work like the numbered \item, but places a ⟨*symbol*⟩ to the *"left"* of the ⟨*label*⟩ separated from it by the ⟨*offset*⟩ 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* = {⟨*symbol*⟩} default: *textborn*

Sets the *symbol* to be displayed to the *"left"* of the box containing the current ⟨*label*⟩ set by labelwidth key for \item* in enumext and enumext*. The *symbol* can be in *text* or *math* mode, for example item-sym*={$\star$}.

item-pos* = {⟨*rigid length*⟩} default: *by levels*

Sets the *offset* between the box containing the current ⟨*label*⟩ defined by labelwidth key and the ⟨*symbol*⟩ 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(⟨*columns*⟩)

The \item command for the enumext* environment provides a *"first optional argument"* \item(⟨*columns*⟩) which *"joins items"* between columns.

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

```
\begin{enumext*}[widest=10,columns=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)

©2024–2026 by Pablo González L

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

item-break = {⟨*true* | *false*⟩} 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

\setenumext

\setenumext{⟨*key = val*⟩}
\setenumext[⟨*enumext, level*⟩]{⟨*key = val*⟩}
\setenumext[⟨*enumext**⟩]{⟨*key = val*⟩}
\setenumext[⟨*keyans*⟩]{⟨*key = val*⟩}
\setenumext[⟨*keyans**⟩]{⟨*key = val*⟩}
\setenumext[⟨*print, level*⟩]{⟨*key = val*⟩}
\setenumext[⟨*print, **⟩]{⟨*key = val*⟩}
\setenumext[⟨*print**⟩]{⟨*key = val*⟩}

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

\setenumextmeta

\setenumextmeta[⟨1⟩]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[⟨2⟩]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[⟨3⟩]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[⟨4⟩]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta[⟨***⟩]{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}
\setenumextmeta*{⟨*key name*⟩}{⟨*key-one = val, key-two = val, ...*⟩}

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 adds 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.

©2024–2026 by Pablo González L

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: 0.3333*em*

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}}
```

©2024–2026 by Pablo González L

```
    {\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 The command \AddEnumextCounter

`\AddEnumextCounter` `\AddEnumextCounter{`⟨*counter style*⟩`}{`⟨*widest label*⟩`}`

The `label` key replaces `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` by the counter style followed by the counter of the current level. The command `\AddEnumextCounter` adds another {⟨*counter style*⟩} to that list, that is a command that receives the name of a counter, for example `\fnsymbol`. The {⟨*widest label*⟩} is a text typeset in a box to calculate the default value of the `labelwidth` key, as it is done with '`M`' for `\Alph*`, so it must be as wide as the widest label that will be used. Since the `widest` key converts a string to a number, with these styles pass it an ⟨*integer*⟩, because `\fnsymbol` only has nine symbols.

The command can be used in the preamble or in the body of the document, but before the style is used in the `label` key. An error is raised if the first argument is not a single control sequence or if the counter style is already registered, including the five styles of the package.

```
\AddEnumextCounter{\fnsymbol}{\textdaggerdbl\textdaggerdbl}
\begin{enumext}[label=\fnsymbol*]
  \item First item
  \item Second item
  \item Third item
\end{enumext}
```

* First item
† Second item
‡ Third item

5.3 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. 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 LaTeX or TeX manual about how penalties control page breaks.

5.4 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`.

5.4.1 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. TEX 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 LATEX 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 *"discardable"*.

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 discardable"*.

below = {⟨*rubber length* | *rigid length*⟩} default: *not used*

Set the *extra vertical space* 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 *"discardable"*.

below* = {⟨*rubber length* | *rigid length*⟩} default: *not used*

Set the *extra vertical space* 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 discardable"*.

5.4.2 Horizontal spaces

list-offset = {⟨*rigid length*⟩} default: 0*pt*

Sets the *horizontal translation* of the entire environment level, that is, the distance from the left margin of the enclosing environment to the left edge of the box defined by the labelwidth key. It is the counterpart of the \labelindent parameter of enumitem (§1.2). Together with the list-indent key it sets the values of \leftmargin and \itemindent for the current level as follows:

\leftmargin = list-offset + list-indent
\itemindent = labelwidth + labelsep − list-indent

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: 0*pt*

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 0pt 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: 0*pt*

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 0pt. Internally sets the value of \rightmargin for the current level.

listparindent = {⟨*rigid length*⟩} default: 0*pt*

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.

5.5 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 LaTeX 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*⟩}}.

©2024–2026 by Pablo González L

`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.

5.6 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 {⟨*series name*⟩} is used again at the *"same level"* at which the key was executed.

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

`resume` = {⟨*series name*⟩} 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=`{⟨*series name*⟩} key was executed, the *start value* will continue numbering according to the last execution of `resume=`{⟨*series name*⟩}. 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=`{⟨*series name*⟩} or `resume=`{⟨*series name*⟩} 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*` ⟨*value forbidden*⟩ 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=`{⟨*series name*⟩} or `resume=`{⟨*series name*⟩} 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=`{⟨*series name*⟩} or `resume*` you will have hierarchy in the ⟨*keys*⟩ that are stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} in the case of `resume*`. If you want to *reset* the value of a ⟨*key*⟩ that is already stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} this must be placed to the *right* of the key `resume=`{⟨*series name*⟩} or `resume*`.

When the `resume*` key is executed consecutively, it does not rewrite the ⟨*keys*⟩ stored in the internal version of {⟨*series name*⟩} and if the environment that precedes it does not have the *optional argument*, it will just continue with the numbering.

©2024–2026 by Pablo González L

5.7 Keys for reset

reset ⟨value forbidden⟩ 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* ⟨value forbidden⟩ 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.

5.7.1 The command \resetenumext

\resetenumext

\resetenumext[⟨1⟩]{⟨some counter⟩}
\resetenumext[⟨2⟩]{⟨some counter⟩}
\resetenumext[⟨3⟩]{⟨some counter⟩}
\resetenumext[⟨4⟩]{⟨some counter⟩}
\resetenumext[⟨*⟩]{⟨some counter⟩}
\resetenumext*{⟨some counter⟩}

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 {⟨some counter⟩}. 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 [1], [2], [3], [4] *"reset"* the values for levels 1, 2, 3 and 4 of the enumext environment, the form [*] *"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* '*' will *"reset"* the values for *"all levels"* of the enumext and enumext* environments.

5.8 Keys for multicols

columns = {⟨integer⟩} default: *1*

Set the *number of columns* to be used by the multicols 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.

5.9 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.

5.9.1 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 LaTeX justification is maintained in the minipage on the *"right side"*.

©2024–2026 by Pablo González L

5.9.2 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.

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{keyans}
          ...
    \end{keyans}
\end{enumext}
```

```
\begin{enumext}[save-ans={⟨store name⟩}]
  \item Text \anskey{answer}
  \item Text
    \begin{keyanspic}
          ...
    \end{keyanspic}
\end{enumext}
```

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

©2024–2026 by Pablo González L

'␣' 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}'.

©2024–2026 by Pablo González L

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 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*

©2024–2026 by Pablo González L

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 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 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 ⟨*keys*⟩ must be passed separated by commas and "without separation" of the start of the environment.

Comments '%' or *any character* after \begin{anskey*} or [⟨*key = val*⟩] on the same line are NOT supported, LaTeX will return an "error" message if this happens. In a similar way comments '%' or *any character* after \end{anskey*} on the same line LaTeX will return a "warning" message.

6.3.1 Keys for anskey*

The anskey* environment uses the same ⟨*keys*⟩ as the \anskey command next to the ⟨*keys*⟩ 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.

write-env = {⟨*file.ext*⟩} default: *not used*

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

overwrite = {⟨*true* | *false*⟩} default: *false*

Sets whether the ⟨*file.ext*⟩ generated by write-env from the anskey* environment will be rewritten.

force-eol = {⟨*true* | *false*⟩} default: *false*

Sets if the *last end of line* for the ⟨*stored content*⟩ 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 \end{Verbatim} or another environment that does not support a comments '%' after closing \end{Verbatim}%.

©2024–2026 by Pablo González L

Example

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

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

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

  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨fourth answer⟩
    \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 `\begin{keyans}[⟨key = val⟩] \item \item[⟨custom⟩] \item* \item*[⟨content⟩] \end{keyans}`
keyans* `\begin{keyans*}[⟨key = val⟩] \item \item[⟨custom⟩] \item* \item*[⟨content⟩] \end{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 \item and \item[⟨custom⟩] work in the usual and the command \item(⟨columns⟩) is available for the keyans* environment.

The behavior of \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.

©2024–2026 by Pablo González L

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}[⟨key = val⟩] \anspic*[⟨content⟩]{⟨drawing or tabular⟩} \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 ⟨*labels*⟩ 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.

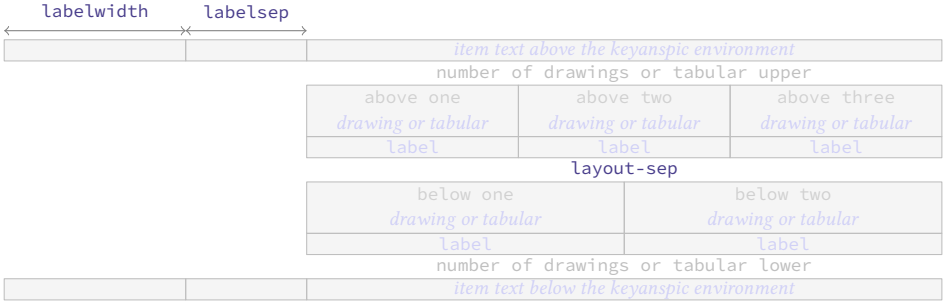


Figure 6: Representation of the keyanspic environment with layout-sty={⟨*3, 2*⟩} in enumext.

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 = {⟨*above* | *below*⟩} default: *below*

Set the *position* of ⟨*label*⟩ to be centered "above" or "below" *drawings* or *tabular* when the \anspic command is executed.

label-sep = {⟨*rubber length* | *rigid length*⟩} default: *1pt above* | *5pt below*

©2024–2026 by Pablo González L

Set the *vertical spacing* between the ⟨*label*⟩ centered "above" or "below" and *drawings* or *tabular* when running the `\anspic` command. The value is the exact distance between the ⟨*label*⟩ and the *drawing or tabular*, with any font size.

`layout-sty` = {⟨*n° upper* , *n° lower*⟩} 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 ⟨*n° lower*⟩ is omitted the *drawings* or *tabular* will be put on a *single line*.

`layout-sep` = {⟨*rubber length* | *rigid length*⟩} default: *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 sets the `parsep` value taken from the `keyans` environment. The value is the exact distance between the rows, with any font size.

`layout-top` = {⟨*rubber length* | *rigid length*⟩} default: *topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. The value is the exact distance between the environment and the neighbouring lines, with any font size. If it is not set, the space above and below is the same as in the `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{`⟨*drawing or tabular*⟩`}`
`\anspic*[`⟨*content*⟩`]{`⟨*drawing or tabular*⟩`}`

The `\anspic` command takes three arguments, the *starred argument* '`*`' store the current ⟨*label*⟩ next to the *optional argument* ⟨*content*⟩ in *sequence* and *prop list* {⟨*store name*⟩} 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

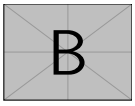
```
\begin{enumext}[save-ans=test,show-ans=true,nosep]
  \item Question with images and labels below.

    \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.

©2024–2026 by Pablo González L



A)



B)



C)

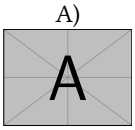


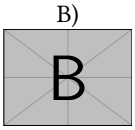
D)

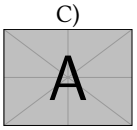


∗ E) [note]

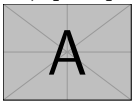
2. Question with images and labels above.

A)


B)


C)


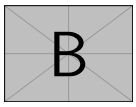
D)


∗ 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 `{`*store name* `:` *position*`}` 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* `{`*store name*`}`. If you pass without options run `\getkeyans` on all contents in *prop list* `{`*store name*`}`.

Options for command

`sep` = `{`*code*`}` default: *{;}*

Establishes the *separation* between *"each"* `{`*content*`}` stored in *prop list* `{`*store name*`}`. 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"* `{`*content*`}` stored in *prop list* `{`*store name*`}`. The value must be a ⟨*positive integer*⟩.

`start` = `{`*integer*`}` default: `1`

Sets the *position* of the *prop list* `{`*store name*`}` from which execution will start. The value must be a ⟨*positive integer*⟩.

`stop` = `{`*integer*`}` default: `0`

Sets the *position* of the *prop list* `{`*store name*`}` from which execution will finish. The value must be a ⟨*positive integer*⟩.

©2024–2026 by Pablo González L

before = {⟨code⟩} default: *empty*

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

after = {⟨code⟩} default: *empty*

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

wrapper = {⟨code `{#1}` more code⟩} default: *empty*

Wraps the {⟨*content*⟩} stored in *prop list* {⟨*store name*⟩} referenced by `{#1}`. The {⟨code⟩} 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* {⟨*store name*⟩} 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 {⟨*store name*⟩} does not exist the command will return an error.

The *optional argument* allows managing the ⟨*keys*⟩ in the *"first level"* of the environment in which the *"stored content"* of the *sequence* {⟨*store name*⟩} 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* {⟨*store name*⟩} 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* {⟨*store name*⟩} 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* {⟨*store name*⟩} already contains any `enumext*` environment an error will be returned, since nesting is not possible.
- If `\printkeyans*`{⟨*store name*⟩} is executed and the *sequence* {⟨*store name*⟩} contains any `enumext` environments, they will start with the ⟨*keys*⟩ 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 \LaTeX2e\ 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] $3(x + y + z)$

2. True False
   (a) LaTeX2e is cool?
   [2] Very True!
3. Related to Linux
   (a) You use Linux?
   [3] Yes
   (b) Rate the following package and class
      i. `xsim`
      [4] very good
      ii. `exsheets`
      [5] obsolete

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: $1\,\text{Å} = 1 \times 10^{-10}$ 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 $1\,\text{Å} = 1 \times 10^{5}$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^{3}$ 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: $1\,\text{Å} = 1 \times 10^{-10}$ 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 $1\,\text{Å} = 1 \times 10^{5}$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^{3}$ 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] The cool TeX automation tool: https://www.ctan.org/pkg/arara

©2024–2026 by Pablo González L

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: $1\,\text{Å} = 1 \times 10^{-10}$ 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 $1\,\text{Å} = 1 \times 10^5$ fm.
   - B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   - C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   - D $1\,\text{Å} = 1 \times 10^3$ 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: $1\,\text{Å} = 1 \times 10^{-10}$ 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 $1\,\text{Å} = 1 \times 10^5$ fm.
   - B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   - C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   - D $1\,\text{Å} = 1 \times 10^3$ fm.

1. B ※ 2. A ※
3. B ※ 4. A ※

Example 3

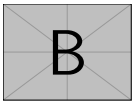
A *"simple multiple choice"* test. Source file.

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



Test keys

1. B, $x = 5$ ※ 4. E, A duck ※
2. D ※ 5. D, other note ※
3. C, some 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) LaTeX2e 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 for pursuing an unexpected course of action.

2

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 for pursuing an unexpected course of action.

4

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.

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

©2024–2026 by Pablo González L

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`, `luamml` 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 `multicols` 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.

With *tagged* PDF a list whose `label` does not use the counter of the level is declared as *unordered*, as it happens in `itemize`. It is not declared as *description*, so the *fake* `description` environments are also *unordered*. When the `label` uses one of the standard counter styles `\arabic`, `\alph`, `\Alph`, `\roman` or `\Roman` the list is declared as *Decimal*, *LowerAlpha*, *UpperAlpha*, *LowerRoman* or *UpperRoman* respectively; any other numbered `label` stays *Ordered*.

Fake `itemize` environment

The `label` key is set here using the default settings in LaTeX 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
  - Second level item
    - Third level item
      - Fourth level item
- First level item

- First level item
  - Second level item
    - Third level item
      - Fourth 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 `\makelabel` is redefined using `\makebox` for the `align` key. The width of this box is the larger of `\labelwidth` and the natural width of the label, so `\item[`⟨*content*⟩`]` wider than `\labelwidth` pushes the text to the right as in the default `description` style and there are no 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 *without* a label.
Something A short one-line description.
Something long A much longer description. Lorem ipsum dolor sit amet, consectetuer 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 *without* a label.
Something A short one-line description.
Something long A much longer description. Lorem ipsum dolor sit amet, consectetuer 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 *without* a label.
Something A short one-line description.
Something long A much longer description. Lorem ipsum dolor sit amet, consectetuer 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`.

©2024–2026 by Pablo González L

```
\NewDocumentCommand \labelbx { s +m }
  {%
    \SuspendTagging{\parbox}%
    \IfBooleanTF{#1}
      {\strut\smash{\parbox[t]{\labelwidth}{\leavevmode\raggedright{#2}}}}%
      {\strut\smash{\parbox[t]{\labelwidth}{\leavevmode\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 *without* a label.

Something A short one-line description.

Something long A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

SoMeThInG LoNg A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum 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 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 LuaTeX 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* and the commands \printkeyans, \getkeyans and \foreachkeyans, plus 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-017 and test-018: the \footnote command in enumext, enumext* and keyans* with tagging=on, with and without the footnotehyper package, where the structure tree must be the same in both cases;
- test-019: the visible vertical spaces of the environment keyanspic with the font sizes 10pt, 11pt and 12pt, measured between the boxes of the PDF, for the keys layout-top, layout-sep and label-sep;
- test-020: the ListNumbering value that the lists declare with tagging=on, according to the counter style of the label key;
- test-021: the counter styles registered with \AddEnumextCounter, 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. “LaTeX 2ε: 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, 2026.

©2024–2026 by Pablo González L

[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 LaTeX Project. “LaTeX for authors current version”. Available from CTAN, https://ctan.org/pkg/latex-base, 2026.

[10] The LaTeX Project. “Extensive support for hypertext in LaTeX”. Available from CTAN, https://www.ctan.org/pkg/hyperref, 2026.

[11] The LaTeX Project. “LaTeX News, Issue 41, June 2025”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[12] The LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, https://www.ctan.org/pkg/enumerate, 2025.

[13] The LaTeX Project. “The LaTeX3 Interfaces”. Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2026.

[14] The LaTeX Project. “The LaTeX 2_ε sources”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2026.

[15] The LaTeX Project. “latex-lab – 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 LaTeX contents in memory or files”. Available from CTAN, https://www.ctan.org/pkg/scontents, 2026.

[18] Fischer, Ulrike. “tagpdf – LaTeX kernel code for PDF tagging”. Available from CTAN, https://www.ctan.org/pkg/tagpdf, 2026.

[19] Mittelbach, Frank. “LaTeX’s socket management”. Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2026.

[20] The LaTeX Project. “The expl3 package”. Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2026.

11 Change history

v2.7 (ctan), 2026-11-25

- *Tagged* PDF: the ordered lists declare the numbering used in the label (`Decimal`, `LowerAlpha`, `UpperAlpha`, `LowerRoman` or `UpperRoman`) instead of the generic `Ordered`.
- *Tagged* PDF: `enumext*` and `keyans*` without the counter in the label are also declared as *unordered* lists.
- New command `\AddEnumextCounter` to register a counter style that the package does not provide, such as `\fnsymbol`, for the `label` key.
- The message written to the `.log` file by the `save-ans` key with the number of items and answers now starts with the name of the storage.

v2.6 (ctan), 2026-11-24

- When `hyperref` is loaded the `save-ref` key creates the link target with `\MakeLinkTarget` and the label with `\label`.
- *Tagged* PDF: the lists whose `label` does not use the counter, for example `label={}` or `label={⟨\textbullet⟩}`, are now declared as *unordered* lists instead of *ordered* lists.
- *Tagged* PDF: in `enumext` and `keyans` a label wider than `\labelwidth` pushes the text instead of overlapping it, and tall labels (images, `tikz` pictures) are no longer smashed and increase the height of the line again.
- The documentation is now a *tagged* PDF (PDF/UA-2).

v2.5 (ctan), 2026-11-14

- Support for `footnotehyper` v2.0 with *tagged* PDF: if it is loaded, it manages the `\footnote` in the environments.
- The `keyanspic` environment now adds the vertical space calculated with the real dimensions: `layout-top`, `layout-sep` and `label-sep` give the exact distance with any font size, and by default the space above and below is the same as in `keyans`. The default value of `label-sep` is `1pt` above and `5pt` below.

©2024–2026 by Pablo González L

- Fixed the tagpdf error for \footnote in enumext* and keyans* with *tagged* PDF when footnotehyper is not loaded.

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 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 LaTeX release 2026-06-01.
- Move \setcounter to \begin{list}{⟨*arg one*⟩}{⟨*arg two*⟩\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 LaTeX release 2025-11-01.
- Move \setcounter to \begin{list}{⟨*arg one*⟩}{⟨*arg two*⟩\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.

©2024–2026 by Pablo González L

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:
- \AddEnumextCounter . . . 9
- \anskey . . . 16, 17
- \anspic . . . 20
- \foreachkeyans . . . 21
- \getkeyans . . . 21
- \item* . . . 5, 6, 18, 19
- \item . . . 5, 6
- \miniright . . . 13
- \printkeyans . . . 22
- \resetenumext . . . 13
- \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 . . . 29
- multicols . . . 29

Environments provided by enumext:
- anskey* . . . 17
- enumext* . . . 4
- enumext . . . 4, 5
- keyans* . . . 18
- keyanspic . . . 19
- keyans . . . 18

I

\itemwidth . . . 4

K

Keys for \anskey provided by enumext:
- break-col . . . 16
- item-join . . . 16
- item-pos* . . . 17
- item-star . . . 17
- item-sym* . . . 17

Keys for \foreachkeyans provided by enumext:
- after . . . 22
- before . . . 22
- sep . . . 21
- start . . . 21
- step . . . 21
- stop . . . 21
- wrapper . . . 22

Keys for anskey* provided by enumext:
- break-col . . . 16
- force-eol . . . 17
- item-join . . . 16
- item-pos* . . . 17
- item-star . . . 17
- item-sym* . . . 17
- overwrite . . . 17
- write-env . . . 17

Keys for environments provided by enumext:
- above* . . . 11
- above . . . 10
- after . . . 12
- align . . . 8
- base-fix . . . 10
- before* . . . 12
- before . . . 11
- beginpenalty . . . 9
- below* . . . 11
- below . . . 11
- check-ans . . . 16
- columns-sep . . . 13
- columns . . . 13
- endpenalty . . . 9
- first . . . 12
- font . . . 8
- item-break . . . 6, 7, 29
- item-pos* . . . 6
- item-sym* . . . 6
- itemindent . . . 11
- itemsep . . . 10
- label-pos . . . 19
- label-sep . . . 19, 29
- labelsep . . . 8
- labelwidth . . . 8
- label . . . 8
- layout-sep . . . 20, 29
- layout-sty . . . 20
- layout-top . . . 20, 29
- list-indent . . . 11
- list-offset . . . 11
- listparindent . . . 11
- mark-ans* . . . 16
- mark-ans . . . 15
- mark-pos* . . . 16
- mark-pos . . . 15
- mark-ref . . . 15
- mark-sep* . . . 16
- mark-sep . . . 15
- midpenalty . . . 9
- mini-env . . . 13
- mini-right* . . . 14
- mini-right . . . 14
- mini-sep . . . 13
- mode-box . . . 8
- no-store . . . 15
- noitemsep . . . 10
- nosep . . . 10
- parsep . . . 10

©2024–2026 by Pablo González L

partopsep 10
ref 9
reset* 13
reset 13
resume* 12
resume 12
rightmargin 11
save-ans 14, 31
save-key 14
save-ref 15
save-sep 14
series 12
show-ans 15
show-length 10
show-pos 16
start* 12
start 12
topsep 10
widest 8
wrap-ans* 16
wrap-ans 15
wrap-label* 9
wrap-label 8
wrap-opt 16

L

Labels provided by enumext:

\Alph* 8
\Roman* 8
\alph* 8
\arabic* 8
\roman* 8
\labelsep 4
\labelwidth 4

P

Packages :

enumerate 28
enumitem 3
fancyvrb 17
footnotehyper 5
geometry 26
graphicx 26
hyperref 5
l3keys 8
l3prop 28
l3seq 28
luamml 26
multicol 1
scontents 31
shortlst 5
tasks 5
unicode-math 26
xsim 2

©2024–2026 by Pablo González L

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.

```
⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention) for l3doc class.

```
⟨@@=enumext⟩
```

13.3 Declaration of the package

Checks for a minimum release version of LaTeX; otherwise it returns an "error".

```
\NeedsTeXFormat{LaTeX2e}[2026-11-01]
\IfFormatAtLeastF { 2026-11-01 }
  {
    \PackageError { enumext }
      { LaTeX~kernel~too~old~need~2026-11-01~or~later }
      {
        enumext~requires~a~LaTeX2e~release~2026-11-01~or~
        later.~Update~your~TeX~distribution.
      }
  }
```

Now declare the enumext package.

```
\ProvidesExplPackage {enumext} {2026-11-25} {2.7} {Enumerate exercise sheets}
```

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

```
\hook_gput_code:nnn {begindocument} {enumext}
  {
    \IfPackageLoadedTF { multicol }
      {
        \msg_info:nnn { enumext } { package-load } { multicol }
      }
      {
        \msg_info:nnn { enumext } { package-not-load } { multicol }
        \RequirePackage{multicol}[2025-10-21]
      }
  }
```

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

\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

Integer variables will control the nesting levels of the environments.

```
\int_new:N \l__enumext_level_int
\int_new:N \l__enumext_level_h_int
\int_new:N \l__enumext_keyans_level_int
\int_new:N \l__enumext_keyans_level_h_int
\int_new:N \l__enumext_keyans_pic_level_int
```

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

\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

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

```
\bool_new:N \l__enumext_starred_bool
\bool_new:N \g__enumext_starred_bool
\bool_new:N \l__enumext_starred_first_bool
\bool_new:N \l__enumext_standard_bool
\bool_new:N \g__enumext_standard_bool
\bool_new:N \l__enumext_standard_first_bool
\bool_new:N \l__enumext_keyans_env_bool
\tl_new:N   \g__enumext_start_line_tl
\tl_new:N   \g__enumext_envir_name_tl
\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

\l__enumext_current_widest_dim

The dimension `\l__enumext_current_widest_dim` stores the current label width. It is used by `widest` (§13.16) and `label` (§13.14) keys.

```
\dim_new:N \l__enumext_current_widest_dim
```

(*End of definition for* `\l__enumext_current_widest_dim`.)

\l__enumext_resume_count_bool
\l__enumext_resume_count_X_bool

Internal variables shared by the `start` and `start*` keys (§13.16), the `resume`, `resume*` and `series` keys (§13.28) and the environments.

```
\bool_new:N \l__enumext_resume_count_bool
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_resume_count_ #1 _bool }
  }
\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`.)

\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl

The integer `\l__enumext_start_X_int` is used by the `start` and `start*` keys (§13.16) and the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.20.1).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \int_new:c { l__enumext_start_ #1 _int           }
    \tl_new:c  { l__enumext_fake_item_indent_ #1 _tl }
  }
\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`.)

\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

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 (§13.14). 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 (§13.14).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_wrap_label_ #1 _bool     }
    \bool_new:c { l__enumext_wrap_label_opt_ #1 _bool }
    \tl_new:c   { l__enumext_label_fill_left_ #1 _tl  }
    \tl_new:c   { l__enumext_label_fill_right_ #1 _tl }
  }
\clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
\cs_set_protected:Nn \__enumext_tmp:n
```

©2024–2026 by Pablo González L

```
  {
    \str_new:c { l__enumext_align_label_pos_ #1 _str }
  }
\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

`\l__enumext_leftmargin_tmp_X_bool`
`\l__enumext_leftmargin_tmp_X_dim`
`\l__enumext_leftmargin_X_dim`

The key `list-indent` (§13.20) uses the boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimension `\l__enumext_leftmargin_tmp_X_dim`. The dimension `\l__enumext_leftmargin_X_dim` is used and set by the function `\__enumext_calc_hspace:NNNNNNN` (§13.43.4).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_leftmargin_tmp_ #1 _bool }
    \dim_new:c  { l__enumext_leftmargin_tmp_ #1 _dim  }
    \dim_new:c  { l__enumext_leftmargin_ #1 _dim      }
  }
\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`.)

`\l__enumext_list_class_tl`

When *tagged* PDF is active this variable stores the name of the attribute class that declares the numbering style of the current list (§13.43.6). It is empty when the style is not one of the standard counter styles.

```
\tl_new:N \l__enumext_list_class_tl
```

(*End of definition for* `\l__enumext_list_class_tl`.)

`\l__enumext_multicols_above_X_skip`
`\l__enumext_multicols_below_X_skip`

Internal variables used by `columns` key (§13.25).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \skip_new:c { l__enumext_multicols_above_ #1 _skip }
    \skip_new:c { l__enumext_multicols_below_ #1 _skip }
  }
\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`.)

`\g__enumext_minipage_stat_int`
`\g__enumext_minipage_right_skip`
`\l__enumext_minipage_left_X_dim`
`\l__enumext_minipage_active_X_bool`

Internal variables used by `\miniright` command (§13.26.7) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.24, §13.26).

```
\int_new:N  \g__enumext_minipage_stat_int
\skip_new:N \g__enumext_minipage_right_skip
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \dim_new:c  { l__enumext_minipage_left_ #1 _dim    }
    \bool_new:c { l__enumext_minipage_active_ #1 _bool }
  }
\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.*)

`\l__enumext_item_starred_X_bool`
`\l__enumext_item_break_X_bool`
`\l__enumext_item_column_pos_X_int`
`\g__enumext_item_count_all_X_int`
`\l__enumext_item_text_X_box`
`\l__enumext_joined_width_X_dim`
`\g__enumext_item_symbol_aux_X_tl`
`\l__enumext_align_label_X_str`
`\g__enumext_minipage_center_X_bool`
`\g__enumext_minipage_right_X_dim`

Internal variables used by `enumext*` and `keyans*` environments. The key `item-break` (§13.22) sets the boolean `\l__enumext_item_break_X_bool`.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_item_starred_ #1 _bool    }
    \bool_new:c { l__enumext_item_break_ #1 _bool      }
    \int_new:c  { l__enumext_item_column_pos_ #1 _int  }
    \int_new:c  { g__enumext_item_count_all_ #1 _int   }
    \box_new:c  { l__enumext_item_text_ #1 _box        }
    \dim_new:c  { l__enumext_joined_width_ #1 _dim     }
    \tl_new:c   { g__enumext_item_symbol_aux_ #1 _tl   }
    \str_new:c  { l__enumext_align_label_ #1 _str      }
    \bool_new:c { g__enumext_minipage_center_ #1 _bool }
    \dim_new:c  { g__enumext_minipage_right_ #1 _dim   }
  }
\clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

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

©2024–2026 by Pablo González L

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 (§13.31.3) 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 *current label* and *optional argument* of \item* (§13.42) and \anspic* (§13.47.4) are saved in the variables \l__enumext_store_current_label_tl and \l__enumext_store_current_opt_arg_tl for the keyans, keyans* and keyanspic environments.

```
\bool_new:N \l__enumext_store_active_bool
\tl_new:N   \l__enumext_store_name_tl
\tl_new:N   \g__enumext_store_name_tl
\tl_new:N   \l__enumext_store_current_label_tl
\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 (§13.31.5) 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.

```
\bool_new:N \l__enumext_check_answers_bool
\bool_new:N \g__enumext_check_ans_key_bool
\tl_new:N   \l__enumext_check_start_line_env_tl
\bool_new:N \l__enumext_item_wrap_key_bool
\int_new:N  \g__enumext_check_starred_cmd_int
\int_new:N  \g__enumext_item_anskey_int
\int_new:N  \g__enumext_item_number_int
\bool_new:N \l__enumext_item_number_bool
\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 (§13.52) 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 position of the marks set by the mark-pos key (§13.32) is stored in the strings \l__enumext_mark_position_str, \l__enumext_mark_position_v_str and \l__enumext_mark_position_viii_str.

```
\bool_new:N \l__enumext_print_keyans_star_bool
\bool_new:N \l__enumext_print_keyans_cmd_bool
\str_new:N  \l__enumext_mark_position_str
\str_new:N  \l__enumext_mark_position_v_str
\str_new:N  \l__enumext_mark_position_viii_str
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \tl_new:c { l__enumext_print_keyans_ #1 _tl }
  }
\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
\l__enumext_footnotehyper_tag_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. The boolean variable \l__enumext_footnotehyper_tag_bool determines if *tagged* PDF is active and the footnotehyper package, with support for it, is loaded.

```
\bool_new:N \l__enumext_hyperref_bool
\bool_new:N \l__enumext_footnotes_key_bool
\bool_new:N \l__enumext_footnotehyper_tag_bool
```

(*End of definition for* \l__enumext_hyperref_bool*,* \l__enumext_footnotes_key_bool *, and* \l__enumext_footnotehyper_-tag_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 (§13.32). 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.

```
\tl_new:N \l__enumext_newlabel_arg_one_tl
\tl_new:N \l__enumext_newlabel_arg_two_tl
\tl_new:N \l__enumext_write_aux_file_tl
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \tl_new:c { l__enumext_label_copy_ #1 _tl }
  }
\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.

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

(*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
item-answer-hook

Messages written to the .log file by the internal hooks that show the values of the global variables of the storage.

```
\msg_new:nnn { enumext } { prop-seq-int-hook }
  {
    Storage~'#1':~
    \c_backslash_str g__enumext_ #1 _prop~has~#2~elements,~
    \c_backslash_str g__enumext_ #1 _seq~has~#3~elements,~
    \c_backslash_str g__enumext_resume_ #1 _int~=~#4.
  }
\msg_new:nnn { enumext } { item-answer-hook }
  {
    Storage~'#1':~items~=~#2,~answers~=~#3,~difference~=~#4.
  }
```

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

13.5.2 General utility functions

\keys_precompile:neN
\seq_use:NV
\keys_set:en
\keys_set:ev
\str_if_in:vnT
\str_if_in:vnF

Non-standard kernel variants used by the \printkeyans command (§13.52) and \foreachkeyans command (§13.55), and by the functions that set the keys of each level.

```
\cs_generate_variant:Nn \keys_precompile:nnN { neN }
\cs_generate_variant:Nn \seq_use:Nn { NV }
\cs_generate_variant:Nn \keys_set:nn { en, ev }
\prg_generate_conditional_variant:Nnn \str_if_in:nn { v } { T, F }
```

(*End of definition for* \keys_precompile:neN *and others.*)

__enumext_at_begin_document:n

A internal *"hook"* function used for copying plain list and minipage environments definition and hyperref detection.

```
\cs_new_protected:Nn \__enumext_at_begin_document:n
  {
    \hook_gput_code:nnn {begindocument} {enumext} {#1}
  }
```

(*End of definition for* __enumext_at_begin_document:n.)

©2024–2026 by Pablo González L

__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.

```
\cs_new_protected:Nn \__enumext_after_env:nn
  {
    \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
  }
```

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

__enumext_level:

Function for check current level in `enumext`.

```
\cs_new:Nn \__enumext_level:
  {
    \int_to_roman:n { \l__enumext_level_int }
  }
```

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

__enumext_if_is_int:nT
__enumext_if_is_int:nF
__enumext_if_is_int:nTF

A conditional function to determine whether the variable being passed is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in How to test if an expl3 function argument is an integer expression?.

```
\prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
  {
    \regex_if_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
      { \prg_return_true: }
      { \prg_return_false: }
  }
```

(*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.

```
\cs_new:Nn \__enumext_show_length:nnn
  {
    *~#2
    \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
      =~\use:c { #1 _use:c } { l__enumext_ #2 _ #3 _ #1 } \\
  }
```

(*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 TEX 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 LATEX does in `\removelastskip`.

```
\cs_new_protected:Nn \__enumext_unskip_unkern:
  {
    \int_case:nn { \lastnodetype }
      {
        { 11 }
          {
            \mode_if_vertical:TF
              {
                \mode_if_inner:TF
                  { \unskip }
                  { \skip_vertical:n { -\lastskip } }
              }
              { \unskip }
          }
        { 12 }
          {
            \mode_if_vertical:TF
              {
                \mode_if_inner:TF
                  { \unkern }
                  { \kern -\lastkern \scan_stop: }
              }
              { \unkern }
          }
      }
  }
```

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

©2024–2026 by Pablo González L

13.5.3 Utilities for environments and levels

__enumext_is_not_nested:
__enumext_is_on_first_level:

The function __enumext_is_not_nested: sets the variables \g__enumext_standard_bool and \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.

```
\cs_new_protected:Nn \__enumext_is_not_nested:
  {
    \str_case:en { \@currenvir }
      {
        {enumext}
          {
            \tl_set:Ne \l__enumext_envir_name_tl { enumext }
            \bool_lazy_and:nnT
              { \bool_not_p:n { \g__enumext_standard_bool } }
              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
              {
                \bool_gset_true:N \g__enumext_standard_bool
              }
          }
        {enumext*}
          {
            \tl_set:Ne \l__enumext_envir_name_tl { enumext* }
            \bool_lazy_and:nnT
              { \bool_not_p:n { \g__enumext_starred_bool } }
              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
              {
                \bool_gset_true:N \g__enumext_starred_bool
              }
          }
      }
  }
```

The function __enumext_is_on_first_level: sets the variables \l__enumext_standard_first_bool (§13.31.3), \l__enumext_starred_first_bool (§13.31.3) 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.

```
\cs_new_protected:Nn \__enumext_is_on_first_level:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_standard_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      }
      {
        \bool_set_true:N \l__enumext_standard_first_bool
        \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
        \tl_gset:Ne \g__enumext_start_line_tl
          {
            on~line~\exp_not:V \inputlineno
          }
      }
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      }
      {
        \bool_set_true:N \l__enumext_starred_first_bool
        \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
        \tl_gset:Ne \g__enumext_start_line_tl
          {
            on~line~\exp_not:V \inputlineno
          }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_keyans_name_and_start:
  {
    \str_case:en { \@currenvir }
      {
        {keyans}
          {
            \tl_set:Ne \l__enumext_envir_name_tl { keyans }
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in~'keyans'~start~on~line~\exp_not:V \inputlineno
              }
          }
        {keyans*}
          {
            \tl_set:Ne \l__enumext_envir_name_tl { keyans* }
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in~'keyans*'~start~on~line~\exp_not:V \inputlineno
              }
          }
        {keyanspic}
          {
            \tl_set:Ne \l__enumext_envir_name_tl { keyanspic }
            \tl_set:Ne \l__enumext_check_start_line_env_tl
              {
                in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
              }
          }
      }
  }
```

(*End of definition for* __enumext_keyans_name_and_start:.)

13.5.4 Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

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

```
\cs_new_protected:Nn \__enumext_reset_global_vars:
  {
    \__enumext_reset_global_int:
    \__enumext_reset_global_bool:
    \__enumext_reset_global_tl:
  }
\cs_new_protected:Nn \__enumext_reset_global_int:
  {
    \int_gzero:N \g__enumext_item_number_int
    \int_gzero:N \g__enumext_item_anskey_int
    \int_gzero:N \g__enumext_item_answer_diff_int
  }
\cs_new_protected:Nn \__enumext_reset_global_bool:
  {
    \bool_gset_false:N \g__enumext_check_ans_key_bool
    \bool_gset_false:N \g__enumext_standard_bool
    \bool_gset_false:N \g__enumext_starred_bool
  }
\cs_new_protected:Nn \__enumext_reset_global_tl:
  {
    \tl_gclear:N \g__enumext_store_name_tl
    \tl_gclear:N \g__enumext_start_line_tl
    \tl_gclear:N \g__enumext_envir_name_tl
  }
```

(*End of definition for* __enumext_reset_global_vars: *and others.*)

__enumext_log_global_vars:
__enumext_log_answer_vars:

The function __enumext_log_global_vars: will be passed to __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.

```
\cs_new_protected:Nn \__enumext_log_global_vars:
```

```
  {
    \msg_info:nneeee { enumext } { prop-seq-int-hook }
      { \g__enumext_store_name_tl }
      { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
      { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
      { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
  }
```

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

```
\cs_new_protected:Nn \__enumext_log_answer_vars:
  {
    \msg_info:nneeee { enumext } { item-answer-hook }
      { \g__enumext_store_name_tl }
      { \int_use:N \g__enumext_item_number_int }
      { \int_use:N \g__enumext_item_anskey_int }
      { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
  }
```

(*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 LaTeX has the following plain form[14]:

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

And `minipage` environment provided by 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 `ltcmd` (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.

```
\__enumext_at_begin_document:n
  {
    \cs_new_eq:NN   \__enumext_start_list:nn \list
    \cs_new_eq:NN   \__enumext_stop_list: \endlist
    \NewCommandCopy \__enumext_item_std:w \item
  }
```

The functions `\__enumext_minipage:w` and `\__enumext_endminipage:` are copies of `\minipage` and `\endminipage` taken from the plain definition of the `minipage` environment.

```
\__enumext_at_begin_document:n
  {
    \cs_new_eq:NN \__enumext_minipage:w \minipage
    \cs_new_eq:NN \__enumext_endminipage: \endminipage
  }
```

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

13.7 Compatibility with hyperref and footnotehyper

`\__enumext_after_hyperref:`

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

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

The function __enumext_after_hyperref: sets 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.

```
\cs_new_protected:Nn \__enumext_after_hyperref:
  {
    \IfPackageLoadedT { hyperref }
      {
        \msg_info:nnn { enumext } { package-load } { hyperref }
        \bool_set_true:N \l__enumext_hyperref_bool
        \IfHyperBoolean{hyperfootnotes}
          {
            \bool_set_true:N \l__enumext_footnotes_key_bool
          }
          { }
      }
```

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. When *tagged* PDF is active and the package footnotehyper is loaded in version 2.0 (2026/11/12) or later, which supports it, the boolean variable \l__enumext_footnotehyper_tag_bool is set to true and \footnote is not redefined.

```
\bool_if:NT \l__enumext_footnotes_key_bool
  {
    \IfPackageLoadedTF { footnotehyper }
      {
        \msg_info:nnn { enumext } { package-load } { footnotehyper }
      }
      {
        \bool_set_false:N \l__enumext_footnotes_key_bool
      }
  }
\IfDocumentMetadataT
  {
    \IfPackageAtLeastT { footnotehyper } { 2026/11/12 }
      {
        \bool_set_true:N \l__enumext_footnotehyper_tag_bool
      }
  }
  }
```

(*End of definition for* __enumext_after_hyperref:.)

__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

If the hyperref package is loaded the link target is created with \MakeLinkTarget* and the label with \label, which writes the \newlabel with the page, the *name* and the *anchor*, using the \@currentlabel and \@currentlabelname set here to \l__enumext_newlabel_arg_two_tl. Without the package a plain \newlabel{#1}{#2} is written, since \label adds spaces that would change the typeset boxes.

```
\cs_new_protected:Nn \__enumext_newlabel:nn
  {
    \bool_if:NTF \l__enumext_hyperref_bool
      {
        \group_begin:
          \tl_set:Nn \@currentlabel {#2}
          \tl_set:Nn \@currentlabelname {#2}
          \MakeLinkTarget* {#1}
          \label {#1}
        \group_end:
      }
      {
        \protected@write \@auxout { }
          { \token_to_str:N \newlabel {#1} { {#2} { } } }
      }
  }
```

(*End of definition for* __enumext_newlabel:nn.)

©2024–2026 by Pablo González L

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
\g__enumext_footnote_standard_int_seq
\g__enumext_footnote_starred_int_seq
\g__enumext_footnote_box
\l__enumext_footnote_wd_dim

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

```
\int_new:N \g__enumext_footnote_standard_int
\int_new:N \g__enumext_footnote_starred_int
\seq_new:N \g__enumext_footnote_standard_arg_seq
\seq_new:N \g__enumext_footnote_starred_arg_seq
\seq_new:N \g__enumext_footnote_standard_int_seq
\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:
__enumext_renew_footnote_mini_tagged:
__enumext_print_footnote_mini_tagged:

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

```
\cs_new_protected:Nn \__enumext_footnotetext:nn
  {
    \footnotetext[#1]{#2}
  }
\cs_new_protected:Nn \__enumext_renew_footnote:
  {
    \RenewDocumentCommand \footnote { o +m }
      {
        \tl_if_novalue:nTF {##1}
          {
            \stepcounter{footnote}
            \int_gset_eq:Nc \g__enumext_footnote_standard_int { c@footnote }
          }
          {
            \int_gset:Nn \g__enumext_footnote_standard_int {##1}
          }
        \footnotemark [ \g__enumext_footnote_standard_int ]
        \seq_gput_right:Nn \g__enumext_footnote_standard_arg_seq {##2}
        \seq_gput_right:NV
          \g__enumext_footnote_standard_int_seq \g__enumext_footnote_standard_int
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote:
  {
    \seq_if_empty:NF \g__enumext_footnote_standard_int_seq
      {
        \seq_map_pairwise_function:NNN
          \g__enumext_footnote_standard_int_seq
          \g__enumext_footnote_standard_arg_seq
          \__enumext_footnotetext:nn
      }
    \seq_gclear:N \g__enumext_footnote_standard_arg_seq
    \seq_gclear:N \g__enumext_footnote_standard_int_seq
  }
```

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.

```
\cs_new_protected:Nn \__enumext_renew_footnote_mini:
  {
    \RenewDocumentCommand \footnote { o +m }
      {
        \tl_if_novalue:nTF {##1}
          {
            \stepcounter{footnote}
            \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
          }
          {
            \int_gset:Nn \g__enumext_footnote_starred_int {##1}
          }
        \footnotemark [ \g__enumext_footnote_starred_int ]
        \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq {##2}
        \seq_gput_right:NV
```

©2024–2026 by Pablo González L

```
448           \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
449       }
450   }
451 \cs_new_protected:Nn \__enumext_print_footnote_mini:
452   {
453     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
454       {
455         \seq_map_pairwise_function:NNN
456           \g__enumext_footnote_starred_int_seq
457           \g__enumext_footnote_starred_arg_seq
458           \__enumext_footnotetext:nn
459       }
460     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
461     \seq_gclear:N \g__enumext_footnote_starred_int_seq
462   }
```

With *tagged* PDF, a `\footnotetext` placed after a list gives a `tagpdf` error (see `tagging-project` issue 1337), so in the `enumext*` and `keyans*` environments the notes are processed inside the item, as the `footnotehyper` package does. The plug `enumextsave` of the socket `fntext/process` saves the note in the box `\g__enumext_footnote_box` instead of inserting it, with the width stored in `\l__enumext_footnote_wd_dim`.

```
463 \box_new:N \g__enumext_footnote_box
464 \dim_new:N \l__enumext_footnote_wd_dim
465 \IfDocumentMetadataT
466   {
467     \NewSocketPlug { fntext/process } { enumextsave }
468       {
469         \vbox_gset:Nn \g__enumext_footnote_box
470           {
471             \vbox_unpack_drop:N \g__enumext_footnote_box
472             \dim_set_eq:NN \columnwidth \l__enumext_footnote_wd_dim
473             #1
474           }
475       }
476   }
```

The function `\__enumext_renew_footnote_mini_tagged:` redefines `\footnote` to use that plug in a group, with the counter and the mark of the document (not those of the `minipage`), and `\__enumext_print_footnote_mini_tagged:` delivers the saved notes after the environment with the default plug of the socket.

```
477 \cs_new_protected:Nn \__enumext_renew_footnote_mini_tagged:
478   {
479     \dim_set_eq:NN \l__enumext_footnote_wd_dim \columnwidth
480     \RenewDocumentCommand \footnote { o +m }
481       {
482         \tl_if_novalue:nTF {##1}
483           {
484             \stepcounter{footnote}
485             \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
486           }
487           {
488             \int_gset:Nn \g__enumext_footnote_starred_int {##1}
489           }
490         \footnotemark [ \g__enumext_footnote_starred_int ]
491         \group_begin:
492           \cs_set:Npn \@mpfn { footnote }
493           \cs_set_eq:NN \thempfn \thefootnote
494           \cs_set_eq:NN \@footnotetext \fnote_footnotetext:n
495           \AssignSocketPlug { fntext/process } { enumextsave }
496           \footnotetext [ \g__enumext_footnote_starred_int ] {##2}
497         \group_end:
498       }
499   }
500 \cs_new_protected:Nn \__enumext_print_footnote_mini_tagged:
501   {
502     \box_if_empty:NF \g__enumext_footnote_box
503       {
504         \socket_use:nn { fntext/process }
505           {
506             \reset@font
507             \footnotesize
508             \int_set_eq:NN \interlinepenalty \interfootnotelinepenalty
```

```
            \skip_set:Nn \splittopskip { \footnotesep }
            \dim_set:Nn \splitmaxdepth { \box_dp:N \strutbox }
            \int_set_eq:NN \floatingpenalty \@MM
            \vbox_unpack_drop:N \g__enumext_footnote_box
          }
      }
  }
```

(*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 the `footnotehyper` package does not manage the notes: when it is not loaded or, with *tagged* PDF active, when its version does not support it. In this last case the notes of the `enumext*` and `keyans*` environments are printed with `\__enumext_print_footnote_mini_tagged:`.

```
\cs_new_protected:Nn \__enumext_renew_footnote_standard:
  {
    \bool_if:NT \g__enumext_standard_bool
      {
        \IfDocumentMetadataTF
          {
            \bool_if:NF \l__enumext_footnotehyper_tag_bool
              {
                \__enumext_renew_footnote:
              }
          }
          {
            \bool_if:NF \l__enumext_footnotes_key_bool
              {
                \__enumext_renew_footnote:
              }
          }
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote_standard:
  {
    \bool_if:NT \g__enumext_standard_bool
      {
        \IfDocumentMetadataTF
          {
            \bool_if:NF \l__enumext_footnotehyper_tag_bool
              {
                \__enumext_print_footnote:
              }
          }
          {
            \bool_if:NF \l__enumext_footnotes_key_bool
              {
                \__enumext_print_footnote:
              }
          }
      }
  }
```

The redefinition of `\footnote` is encapsulated to pass it to the `enumext*` and `keyans*` environments. It is executed under the same conditions.

```
\cs_new_protected:Nn \__enumext_renew_footnote_starred:
  {
    \IfDocumentMetadataTF
      {
        \bool_if:NF \l__enumext_footnotehyper_tag_bool
          {
            \__enumext_renew_footnote_mini_tagged:
          }
      }
      {
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_renew_footnote_mini:
          }
```

```
      }
  }
\cs_new_protected:Nn \__enumext_print_footnote_starred:
  {
    \IfDocumentMetadataTF
      {
        \bool_if:NF \l__enumext_footnotehyper_tag_bool
          {
            \__enumext_print_footnote_mini_tagged:
          }
      }
      {
        \bool_if:NF \l__enumext_footnotes_key_bool
          {
            \__enumext_print_footnote_mini:
          }
      }
  }
```

In enumext* and keyans* environments it is necessary to use *"hooks"* to print \footnote with support for *tagged* PDF when the redefinition above is used.

```
\__enumext_after_env:nn { enumext* }
  {
    \__enumext_print_footnote_starred:
  }
\__enumext_after_env:nn { keyans* }
  {
    \__enumext_print_footnote_starred:
  }
```

(*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 (§13.44) and __enumext_safe_exec_vii: in the enumext* environment definition (§13.49).

```
\cs_new_protected:Nn \__enumext_internal_mini_page:
  {
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
          {
            \__enumext_renew_footnote_standard:
            \__enumext_minipage:w [ t ] {##1}
              \legacy_if_gset_false:n { @minipage }
              \skip_vertical:N \c_zero_skip
          }
          {
              \skip_vertical:N \c_zero_skip
            \__enumext_endminipage:
            \__enumext_print_footnote_standard:
          }
      }
  }
```

(*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.

```
\dim_zero_new:N \itemwidth
```

©2024–2026 by Pablo González L

13.11 Definition of counters

13.11.1 Definition of variables

\l__enumext_counter_i_tl
\l__enumext_counter_ii_tl
\l__enumext_counter_iii_tl
\l__enumext_counter_iv_tl
\l__enumext_counter_v_tl
\l__enumext_counter_vi_tl
\l__enumext_counter_vii_tl
\l__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 (§13.15).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \tl_new:c { l__enumext_counter_ #1 _tl }
  }
\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.

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

(*End of definition for* counters.)

13.11.3 Internal functions for the counters

__enumext_define_counter:Nn
enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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:

#1 : A token list \l__enumext_counter_X_tl to *"store"* the counter's name.
#2 : The counter's name.

```
\cs_new_protected:Nn \__enumext_define_counter:Nn
  {
    \cs_if_exist:cTF { c@ #2 }
      { \msg_fatal:nnn { enumext } { counters }{#2} }
      {
        \tl_set:Nn #1 {#2}
        \newcounter {#2}
      }
  }
```

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.

```
\__enumext_define_counter:Nn \l__enumext_counter_i_tl    { enumXi    }
\__enumext_define_counter:Nn \l__enumext_counter_ii_tl   { enumXii   }
\__enumext_define_counter:Nn \l__enumext_counter_iii_tl  { enumXiii  }
\__enumext_define_counter:Nn \l__enumext_counter_iv_tl   { enumXiv   }
\__enumext_define_counter:Nn \l__enumext_counter_v_tl    { enumXv    }
\__enumext_define_counter:Nn \l__enumext_counter_vi_tl   { enumXvi   }
\__enumext_define_counter:Nn \l__enumext_counter_vii_tl  { enumXvii  }
\__enumext_define_counter:Nn \l__enumext_counter_viii_tl { enumXviii }
```

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

\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

In version 1.6 the command \resetenumext (§13.29) 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*.

```
\cs_set_protected:Nn \__enumext_tmp:n
```

```
645   {
646     \cs_if_exist:cTF { c@ __enumext_resume_ #1 _int }
647       { \msg_fatal:nne { enumext } { counters }{ __enumext_resume_ #1 _int } }
648       {
649         \newcounter { __enumext_resume_ #1 _int }
650       }
651   }
652 \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 LaTeX 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` (§13.16) and `label` (§13.14) keys.

```
653 \tl_new:N  \g__enumext_counter_styles_tl
654 \tl_new:N  \g__enumext_widest_label_tl
655 \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.

```
656 \cs_new_protected:Nn \__enumext_register_default_label_wd:Nn
657   {
658     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
659     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
660   }
661 \__enumext_register_default_label_wd:Nn \arabic { 0 }
662 \__enumext_register_default_label_wd:Nn \Alph   { M }
663 \__enumext_register_default_label_wd:Nn \alph   { m }
664 \__enumext_register_default_label_wd:Nn \Roman  { VIII }
665 \__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.

```
666 \cs_new_protected:Nn \__enumext_label_width_by_box:Nn
667   {
668     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
669     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
670   }
671 \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.

```
672 \cs_new_protected:Nn \__enumext_label_style:Nnn
673   {
674     \tl_clear_new:N #1
675     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
676     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
677     \tl_map_inline:Nn \g__enumext_counter_styles_tl
678       {
```

```
      \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
      \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
        { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
    }
  \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
    { \tl_use:N \g__enumext_widest_label_tl }
  \tl_set_eq:cN { the #2 } #1
 }
\cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

13.13 The command `\AddEnumextCounter`

The `label` key replaces every style registered in `\g__enumext_counter_styles_tl` followed by a star with that style applied to the counter of the level. The command `\AddEnumextCounter` registers one more style with `\__enumext_register_default_label_wd:Nn`, so that a style defined outside the package, such as `\fnsymbol`, can be written as `\fnsymbol*` in the key.

counter-not-cs
counter-registered

Messages used by `\AddEnumextCounter` command.

```
\msg_new:nnnn { enumext } { counter-not-cs }
  {
    The~first~argument~'#1'~is~not~a~control~sequence~\msg_line_context:.
  }
  {
    The~first~argument~must~be~a~counter~style~such~as~
    '\c_backslash_str fnsymbol'.
  }
\msg_new:nnnn { enumext } { counter-registered }
  {
    The~counter~style~'#1'~is~already~registered~\msg_line_context:.
  }
  {
    A~counter~style~can~only~be~registered~once.~The~package~already~
    registers~'\c_backslash_str arabic',~'\c_backslash_str Alph',~
    '\c_backslash_str alph',~'\c_backslash_str Roman'~and~
    '\c_backslash_str roman'.
  }
```

(*End of definition for* `counter-not-cs` *and* `counter-registered`.)

__enumext_add_counter:nn

Checks that '`#1`' is a single control sequence that is not registered yet, and then registers it with the example '`#2`' used to calculate the default `labelwidth`.

```
\cs_new_protected:Nn \__enumext_add_counter:nn
  {
    \bool_lazy_and:nnTF
      { \tl_if_single_p:n {#1} }
      { \token_if_cs_p:N #1 }
      {
        \tl_if_exist:cTF { c__enumext_widest_ \cs_to_str:N #1 _tl }
          { \msg_error:nne { enumext } { counter-registered } { \token_to_str:N #1 } }
          { \__enumext_register_default_label_wd:Nn #1 {#2} }
      }
      { \msg_error:nne { enumext } { counter-not-cs } { \tl_to_str:n {#1} } }
  }
```

(*End of definition for* `\__enumext_add_counter:nn`.)

\AddEnumextCounter

Defines the user command `\AddEnumextCounter`. It accepts the counter style in '`#1`' and the example used to measure the label in '`#2`'.

```
\NewDocumentCommand \AddEnumextCounter { m m }
  {
    \__enumext_add_counter:nn {#1} {#2}
  }
```

(*End of definition for* `\AddEnumextCounter`. *This function is documented on page 9.*)

13.14 Setting keys associated with label

When *tagged* PDF is active \makelabel is redefined using \makebox to work correctly (§13.39). From the user side it is convenient to have a key that allows using this redefinition with \makebox without having \IfDocumentMetadataTF active.

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

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext /  #1 }
      {
        mode-box .bool_set:N = \l__enumext_mode_box_bool,
        mode-box .initial:n = false,
        mode-box .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

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

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        font        .tl_set:c = { l__enumext_label_font_style_ #2 _tl },
        font        .value_required:n = true,
        labelsep    .dim_set:c = { l__enumext_labelsep_ #2 _dim },
        labelsep    .initial:n = {0.3333em},
        labelsep    .value_required:n = true,
        labelwidth  .dim_set:c = { l__enumext_labelwidth_ #2 _dim },
        labelwidth  .value_required:n = true,
        wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_ #2 :n } ##1,
        wrap-label  .initial:n = {##1},
        wrap-label  .value_required:n = true,
        wrap-label* .code:n =
          {
            \bool_set_true:c { l__enumext_wrap_label_opt_ #2 _bool }
            \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
          },
        wrap-label* .value_required:n = true,
      }
  }
\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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        align .choices:nn =
          { left , right , center }
          {
            \int_case:nn { \l_keys_choice_int }
              {
                {1}
                  {
                    \tl_clear:c { l__enumext_label_fill_left_ #2 _tl  }
                    \tl_set:cn  { l__enumext_label_fill_right_ #2 _tl } { \hfill }
                    \str_set:cn { l__enumext_align_label_pos_ #2 _str } { l }
                  }
                {2}
                  {
                    \tl_set:cn  { l__enumext_label_fill_left_ #2 _tl  } { \hfill }
                    \tl_clear:c { l__enumext_label_fill_right_ #2 _tl }
                    \str_set:cn { l__enumext_align_label_pos_ #2 _str } { r }
```

```
                  }
                {3}
                  {
                    \tl_set:cn { l__enumext_label_fill_left_ #2 _tl  } { \hfill }
                    \tl_set:cn { l__enumext_label_fill_right_ #2 _tl } { \hfill }
                    \str_set:cn { l__enumext_align_label_pos_ #2 _str } { c }
                  }
              }
          },
        align .initial:n = left,
        align .value_required:n = true,
        align / unknown .code:n =
          \msg_error:nneee { enumext } { unknown-choice }
            { align } { left,~right,~ center } { \exp_not:n {##1} },
      }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
  }
  { \__enumext_tmp:nn #1 }

\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        align .choices:nn =
          { left , right , center }
          {
            \int_case:nn { \l_keys_choice_int }
              {
                {1} { \str_set:cn { l__enumext_align_label_ #2 _str } { l } }
                {2} { \str_set:cn { l__enumext_align_label_ #2 _str } { r } }
                {3} { \str_set:cn { l__enumext_align_label_ #2 _str } { c } }
              }
          },
        align .initial:n = left,
        align .value_required:n = true,
        align / unknown .code:n =
          \msg_error:nneee { enumext } { unknown-choice }
            { align } { left,~right,~ center } { \exp_not:n {##1} },
      }
  }
\clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* `align`.)

13.15 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.

13.15.1 Definition of variables

`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_the_count_tl`
`\l__enumext_renew_counter_X_tl`
`\l__enumext_the_counter_X_tl`

Internal variables used by `ref` key (§13.15).

```
\tl_new:N \l__enumext_ref_key_arg_tl
\tl_new:N \l__enumext_ref_the_count_tl
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \tl_new:c  { l__enumext_renew_counter_ #1 _tl }
    \tl_new:c  { l__enumext_the_counter_ #1 _tl   }
    \tl_set:ce { l__enumext_the_counter_ #1 _tl   } { \exp_not:c { theenumX #1 } }
  }
\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.*)

13.15.2 Definition of error messages

`key-ref-empty`

Messages used by `ref` key.

```
\msg_new:nnnn { enumext } { key-ref-empty }
```

©2024–2026 by Pablo González L

```
  {
    Key~'ref'~needs~a~value~in~'#1'~\msg_line_context:.
  }
  {
    Give~a~value~to~the~key~'ref'~or~remove~it.
  }
```

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

13.15.3 Define and set `label` and `ref` keys for `enumext` environment

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
\cs_set_protected:Nn \__enumext_tmp:nnn
  {
    \tl_const:cn { c__enumext_default_label_ #2 _tl } {#3}
    \keys_define:nn { enumext / #1 }
      {
        label .code:n =
          {
            \__enumext_label_style:cvn { l__enumext_label_ #2 _tl }
              { l__enumext_counter_ #2 _tl } {##1}
            \dim_set_eq:cN  { l__enumext_labelwidth_ #2 _dim }
              \l__enumext_current_widest_dim
          },
        label .initial:n = #3,
        label .value_required:n = true,
        ref   .code:n = { \__enumext_standard_ref:nn {#2} {##1} },
        ref   .value_required:n = true,
      }
  }
\__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
\__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
\__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
\__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`.

```
\cs_new_protected:Nn \__enumext_standard_ref:nn
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#2}
    \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
      {
        \msg_error:nnn { enumext } { key-ref-empty } { enumext }
      }
      {
        \tl_set_eq:Nc \l__enumext_ref_the_count_tl { l__enumext_the_counter_ #1 _tl }
        \tl_set:ce { l__enumext_renew_counter_ #1 _tl }
          {
            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
              { \exp_not:V \l__enumext_ref_key_arg_tl }
          }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_standard_ref:
  {
    \tl_if_empty:cF { l__enumext_renew_counter_ \__enumext_level: _tl }
      {
        \tl_use:c { l__enumext_renew_counter_ \__enumext_level: _tl }
      }
  }
```

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

©2024–2026 by Pablo González L

13.15.4 Define and set `label` and `ref` keys for `enumext*` and `keyans*` environments

`label` `ref` `\l__enumext_label_vii_tl` `\l__enumext_label_viii_tl`

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

```
\cs_set_protected:Nn \__enumext_tmp:nnn
  {
    \tl_const:cn { c__enumext_default_label_ #2 _tl } {#3}
    \keys_define:nn { enumext / #1 }
      {
        label .code:n =
          {
            \__enumext_label_style:cvn { l__enumext_label_ #2 _tl }
              { l__enumext_counter_ #2 _tl } {##1}
            \dim_set_eq:cN  { l__enumext_labelwidth_ #2 _dim }
                \l__enumext_current_widest_dim
          },
        label .initial:n = #3,
        label .value_required:n = true,
        ref   .code:n = { \__enumext_starred_ref:n {##1} },
        ref   .value_required:n = true,
      }
  }
\__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
\__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`.

```
\cs_new_protected:Nn \__enumext_starred_ref:n
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
          {
            \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
          }
          {
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
            \tl_set:Ne \l__enumext_renew_counter_vii_tl
              {
                \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
                  { \exp_not:V \l__enumext_ref_key_arg_tl }
              }
          }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
          {
            \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
          }
          {
            \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
            \tl_set:Ne \l__enumext_renew_counter_viii_tl
              {
                \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl }
                  { \exp_not:V \l__enumext_ref_key_arg_tl }
              }
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_starred_ref:
  {
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
```

```
            {
              \tl_use:N \l__enumext_renew_counter_vii_tl
            }
        }
      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
        {
          \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
            {
              \tl_use:N \l__enumext_renew_counter_viii_tl
            }
        }
    }
```

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

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

`label` `ref` `\l__enumext_label_v_tl` `\l__enumext_label_vi_tl`

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.

```
\keys_define:nn { enumext / keyans }
  {
    label .code:n =
      {
        \__enumext_label_style:cvn { l__enumext_label_v_tl }
          { l__enumext_counter_v_tl } {#1}
        \__enumext_label_style:cvn { l__enumext_label_vi_tl }
          { l__enumext_counter_vi_tl } {#1}
        \dim_set_eq:NN
          \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
      },
    label .initial:n = \Alph*),
    label .value_required:n = true,
    ref   .code:n = { \__enumext_keyans_ref:n {#1} },
    ref   .value_required:n = true,
  }
```

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

`\__enumext_keyans_ref:n` `\__enumext_keyans_ref:`

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

```
\cs_new_protected:Nn \__enumext_keyans_ref:n
  {
    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
    \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
      {
        \msg_error:nnn { enumext } { key-ref-empty } { keyans }
      }
      {
        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
        \tl_put_right:Ne \l__enumext_renew_counter_v_tl
          {
            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
              { \exp_not:V \l__enumext_ref_key_arg_tl }
          }
      }
  }
```

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

```
\cs_new_protected:Nn \__enumext_keyans_ref:
  {
    \tl_if_empty:NF \l__enumext_renew_counter_v_tl
      {
        \tl_use:N \l__enumext_renew_counter_v_tl
      }
  }
```

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

©2024–2026 by Pablo González L

13.16 Setting start, start* and widest keys

__enumext_start_from:NNn
__enumext_start_from:ccn
__enumext_start_from:cce

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

#1 : \l__enumext_label_X_tl
#2 : \l__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 (§13.28). This is necessary to be able to "reset" the *start value* by means of the start or start* keys.

```
\cs_new_protected:Nn \__enumext_start_from:NNn
  {
    \bool_set_false:N \l__enumext_resume_count_bool
    \__enumext_if_is_int:nTF {#3}
      {
        \int_set:Nn #2 {#3}
      }
      {
        \regex_if_match:nVT { \c{Alph} | \c{alph} } #1
          { \int_set:Nn #2 { \int_from_alph:n {#3} } }
        \regex_if_match:nVT { \c{Roman} | \c{roman} } #1
          { \int_set:Nn #2  { \int_from_roman:n {#3} } }
      }
  }
\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 \Alph, \alph, \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.

```
\cs_new_protected:Nn \__enumext_widest_from:nNNn
  {
    \__enumext_if_is_int:nTF {#4}
      {
        \setcounter{enumX #1} {#4}
      }
      {
        \regex_if_match:nVT { \c{Alph} | \c{alph} } #2
          { \setcounter{enumX #1} { \int_from_alph:n {#4} } }
        \regex_if_match:nVT { \c{Roman} | \c{roman} } #2
          { \setcounter{enumX #1} { \int_from_roman:n {#4} } }
      }
    \__enumext_label_width_by_box:cv
      { l__enumext_labelwidth_ #1 _dim } { l__enumext_label_ #1 _tl }
  }
\cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

(*End of definition for* __enumext_widest_from:nNNn.)

start
start*
widest

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

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        start* .code:n =
          {
            \__enumext_start_from:ccn
              { l__enumext_label_ #2 _tl }
```

```
                { l__enumext_start_ #2 _int } {##1}
            },
          start* .value_required:n = true,
          start  .code:n =
            {
              \__enumext_start_from:cce
                { l__enumext_label_ #2 _tl }
                { l__enumext_start_ #2 _int } { \int_eval:n {##1} }
            },
          start  .initial:n = 1,
          start  .value_required:n = true,
          widest .code:n =
            {
              \__enumext_widest_from:nccn {#2}
                { l__enumext_label_ #2 _tl }
                { l__enumext_labelwidth_ #2 _dim } {##1}
            },
          widest .value_required:n = true,
        }
    }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.17 Setting keys for penalties

beginpenalty
midpenalty
endpenalty

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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        beginpenalty .int_set:c = { l__enumext_beginparpenalty_ #2 _int },
        beginpenalty .initial:n = -51,
        beginpenalty .value_required:n = true,
        midpenalty   .int_set:c = { l__enumext_itempenalty_ #2 _int },
        midpenalty   .initial:n = -51,
        midpenalty   .value_required:n = true,
        endpenalty   .int_set:c = { l__enumext_endparpenalty_ #2 _int },
        endpenalty   .initial:n = -51,
        endpenalty   .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.18 Setting keys for vertical spaces

topsep
partopsep
parsep
itemsep
noitemsep
nosep

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

```
\cs_set_protected:Nn \__enumext_tmp:nnnnnn
  {
    \keys_define:nn { enumext / #1 }
      {
        topsep    .skip_set:c = { l__enumext_topsep_ #2 _skip },
        topsep    .initial:n = {#3},
        topsep    .value_required:n = true,
        partopsep .skip_set:c = { l__enumext_partopsep_ #2 _skip },
        partopsep .initial:n = {#4},
        partopsep .value_required:n = true,
        parsep    .skip_set:c = { l__enumext_parsep_ #2 _skip },
        parsep    .initial:n = {#5},
        parsep    .value_required:n = true,
        itemsep   .skip_set:c = { l__enumext_itemsep_ #2 _skip },
        itemsep   .initial:n = {#6},
        itemsep   .value_required:n = true,
        noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
        noitemsep .value_forbidden:n = true,
```

©2024–2026 by Pablo González L

```
      nosep     .meta:n =
        { itemsep = 0pt, parsep = 0pt, topsep = 0pt, partopsep = 0pt },
      nosep     .value_forbidden:n = true,
    }
}
```

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

```
\__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
  { 4.0pt plus 2.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
  { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
  { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
  { 4.0pt plus 2.0pt minus 1.0pt }
\__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
  { 2.0pt plus 1.0pt minus 1.0pt }
```

(*End of definition for* `topsep` *and others.*)

13.19 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.

```
\keys_define:nn { enumext / level-1 }
  {
    base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
    base-fix .initial:n = false,
    base-fix .value_forbidden:n = true,
  }
```

(*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 (§13.44) will be responsible for applying the *baseline correction* and adjusting the ⟨*keys*⟩ for the `enumext` environment and the `\printkeyans` with *starred argument* '`*`' (§13.52).

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 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.

```
\cs_new_protected:Nn \__enumext_nested_base_line_fix:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_starred_first_bool }
        { \bool_if_p:N \l__enumext_base_line_fix_bool }
        { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
      }
```

```
      {
        \mode_leave_vertical:
        \vspace
          {
            -\dim_eval:n
              {
                \baselineskip
                + \int_compare:nNnTF { \l__enumext_columns_i_int } > { 1 }
                  { 0pt }
                  { \l__enumext_parsep_vii_skip }
              }
          }
        \keys_set:nn { enumext / level-1 }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
```

When the \printkeyans command is run with the *starred argument* '*' the variable is true (\l__enumext_print_keyans_star_bool), and 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.

```
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_starred_first_bool }
      { \bool_if_p:N \l__enumext_print_keyans_star_bool }
      {
        \mode_leave_vertical:
        \skip_vertical:n { -\baselineskip }
        \skip_vertical:N \c_zero_skip
        \keys_set:nn { enumext / level-1 }
          {
            topsep = 0pt, above = 0pt, above* = 0pt,
          }
      }
    \bool_set_false:N \l__enumext_base_line_fix_bool
  }
```

(*End of definition for* __enumext_nested_base_line_fix:.)

13.20 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        itemindent    .dim_set:c = { l__enumext_fake_item_indent_ #2 _dim },
        itemindent    .value_required:n = true,
        rightmargin   .dim_set:c = { l__enumext_rightmargin_ #2 _dim },
        rightmargin   .value_required:n = true,
        listparindent .dim_set:c = { l__enumext_listparindent_ #2 _dim },
        listparindent .value_required:n = true,
        list-offset   .dim_set:c = { l__enumext_listoffset_ #2 _dim },
        list-offset   .value_required:n = true,
        list-indent   .code:n =
          {
            \bool_set_true:c { l__enumext_leftmargin_tmp_ #2 _bool }
            \dim_set:cn { l__enumext_leftmargin_tmp_ #2 _dim } {##1}
          },
        list-indent   .value_required:n = true,
      }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
  }
  { \__enumext_tmp:nn #1 }
```

©2024–2026 by Pablo González L

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

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        itemindent    .dim_set:c = { l__enumext_fake_item_indent_ #2 _dim },
        itemindent    .value_required:n = true,
        rightmargin   .dim_set:c = { l__enumext_rightmargin_ #2 _dim },
        rightmargin   .value_required:n = true,
        listparindent .dim_set:c = { l__enumext_listparindent_ #2 _dim },
        listparindent .value_required:n = true,
        list-offset   .dim_set:c = { l__enumext_listoffset_ #2 _dim },
        list-offset   .value_required:n = true,
        list-indent   .meta:n = { list-offset = ##1 },
        list-indent   .value_required:n = true,
      }
  }
\clist_map_inline:nn
  {
    {enumext*}{vii}, {keyans*}{viii}
  }
  { \__enumext_tmp:nn #1 }
```

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

13.20.1 Functions for setting the fake `itemindent`

`\__enumext_fake_item_indent:`
`\__enumext_keyans_fake_item_indent:`
`\__enumext_fake_item_indent_vii:`
`\__enumext_fake_item_indent_viii:`

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 0pt. 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.

```
\cs_set_protected:Nn \__enumext_fake_item_indent:
  {
    \dim_compare:nNnT
      { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
      >
      { \c_zero_dim }
      {
        \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:n { \skip_horizontal:n }
              { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_v_tl
          {
            \exp_not:N \mode_leave_vertical:
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_fake_item_indent_vii:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
          {
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
```

```
            \exp_not:N \ignorespaces
          }
      }
  }
\cs_set_protected:Nn \__enumext_fake_item_indent_viii:
  {
    \dim_compare:nNnT
      { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
      {
        \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
          {
            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
            \exp_not:N \ignorespaces
          }
      }
  }
```

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

13.21 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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        show-length .bool_set:c = { l__enumext_show_length_ #2 _bool },
        show-length .initial:n = false,
        show-length .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.22 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 (§13.49).

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        item-break .bool_set:c = { l__enumext_item_break_ #2 _bool },
        item-break .initial:n = false,
        item-break .value_required:n = true,
      }
  }
\clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.23 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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        before  .tl_set:c = { l__enumext_before_no_starred_key_ #2 _tl },
        before  .value_required:n = true,
        before* .tl_set:c = { l__enumext_before_starred_key_ #2 _tl },
        before* .value_required:n = true,
        after   .tl_set:c = { l__enumext_after_stop_list_ #2 _tl },
        after   .value_required:n = true,
        first   .tl_set:c = { l__enumext_after_list_args_ #2 _tl },
        first   .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

©2024–2026 by Pablo González L

13.23.1 Functions for `before`, `after` and `first` keys in `enumext`

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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⟩}.

```
\cs_new_protected:Nn \__enumext_before_args_exec:
  {
    \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
  }
```

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⟩}}

```
\cs_new_protected:Nn \__enumext_before_keys_exec:
  {
    \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
  }
```

The function __enumext_after_stop_list: executes the {⟨code⟩} set by the after key *"after"* the enumext environment has finished: \endlist{⟨code⟩}.

```
\cs_new_protected:Nn \__enumext_after_stop_list:
  {
    \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_after_args_exec:
  {
    \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
  }
```

(*End of definition for* __enumext_before_args_exec: *and others.*)

13.23.2 Functions for `before`, `after` and `first` keys in `keyans`

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:

Same implementation as the one used in the enumext environment.

```
\cs_new_protected:Nn \__enumext_before_args_exec_v:
  {
    \tl_use:N \l__enumext_before_starred_key_v_tl
  }
\cs_new_protected:Nn \__enumext_before_keys_exec_v:
  {
    \tl_use:N \l__enumext_before_no_starred_key_v_tl
  }
\cs_new_protected:Nn \__enumext_after_stop_list_v:
  {
    \tl_use:N \l__enumext_after_stop_list_v_tl
  }
\cs_new_protected:Nn \__enumext_after_args_exec_v:
  {
    \tl_use:N \l__enumext_after_list_args_v_tl
  }
```

(*End of definition for* __enumext_before_args_exec_v: *and others.*)

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

__enumext_before_args_exec_vii:
__enumext_before_args_exec_viii:
__enumext_before_keys_exec_vii:
__enumext_before_keys_exec_viii:
__enumext_after_stop_list_vii:
__enumext_after_stop_list_viii:

Same implementation as the one used in the enumext environment, except that the functions __enumext_after_args_exec_vii: and __enumext_after_args_exec_viii: do not exist. Instead, 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.

```
\cs_new_protected:Nn \__enumext_before_args_exec_vii:
  {
    \tl_use:N \l__enumext_before_starred_key_vii_tl
  }
\cs_new_protected:Nn \__enumext_before_args_exec_viii:
  {
```

```
    \tl_use:N \l__enumext_before_starred_key_viii_tl
  }
\cs_new_protected:Nn \__enumext_before_keys_exec_vii:
  {
    \tl_use:N \l__enumext_before_no_starred_key_vii_tl
  }
\cs_new_protected:Nn \__enumext_before_keys_exec_viii:
  {
    \tl_use:N \l__enumext_before_no_starred_key_viii_tl
  }
\cs_new_protected:Nn \__enumext_after_stop_list_vii:
  {
    \tl_use:N \l__enumext_after_stop_list_vii_tl
  }
\cs_new_protected:Nn \__enumext_after_stop_list_viii:
  {
    \tl_use:N \l__enumext_after_stop_list_viii_tl
  }
```

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

13.24 Setting keys for `multicols` and `minipage`

mini-env
mini-sep
columns-sep
columns
mini-right
mini-right*

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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        mini-env    .dim_set:c = { l__enumext_minipage_right_ #2 _dim },
        mini-env    .value_required:n = true,
        mini-sep    .dim_set:c = { l__enumext_minipage_hsep_ #2 _dim },
        mini-sep    .initial:n = 0.3333em,
        mini-sep    .value_required:n = true,
        columns-sep .dim_set:c = { l__enumext_columns_sep_ #2 _dim },
        columns-sep .value_required:n = true,
        columns     .int_set:c = { l__enumext_columns_ #2 _int },
        columns     .initial:n = 1,
        columns     .value_required:n = true,
      }
  }
\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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        mini-right  .tl_gset:c = { g__enumext_miniright_code_ #2 _tl },
        mini-right  .value_required:n = true,
        mini-right* .code:n =
          {
            \bool_gset_true:c { g__enumext_minipage_center_ #2 _bool }
            \keys_set:nn { enumext / #1 } { mini-right = {##1} }
          },
        mini-right* .value_required:n = true,
      }
  }
\clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.25 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.



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

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.

13.25.1 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 TEX is in ⟨*horizontal mode*⟩, then the settings for the ⟨*vertical mode*⟩, in which `\partopsep` comes into play, are made.

Set `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` equal to the value of `\topsep` in the *current level*.

```
\cs_new_protected:Nn \__enumext_multi_set_vskip:
  {
    \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \__enumext_add_pre_parsep:
  }
```

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

`\__enumext_add_pre_parsep:`

Since `\parsep` in the previous level affects the *vertical spaces*, the function `\__enumext_add_pre_parsep:` detects its value and *"adjusts"* `\l__enumext_multicols_above_X_skip`.

```
\cs_new_protected:Nn \__enumext_add_pre_parsep:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 2 }
          {
            \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
              {
                \skip_add:Nn \l__enumext_multicols_above_ii_skip
                  {
                    \l__enumext_parsep_i_skip
                  }
              }
          }
        { 3 }
          {
            \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
              {
                \skip_add:Nn \l__enumext_multicols_above_iii_skip
                  {
                    \l__enumext_parsep_ii_skip
                  }
              }
          }
        { 4 }
```

```
            {
              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
                {
                  \skip_add:Nn \l__enumext_multicols_above_iv_skip
                    {
                      \l__enumext_parsep_iii_skip
                    }
                }
            }
        }
    }
```

(*End of definition for* `\__enumext_add_pre_parsep:`.)

`\__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*⟩.

```
\cs_new_protected:Nn \__enumext_multi_addvspace:
  {
    \__enumext_multi_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
        \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
      }
    \par\nopagebreak
    \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
  }
```

(*End of definition for* `\__enumext_multi_addvspace:`.)

13.25.2 Adjustment of vertical spaces for `multicols` in `keyans`

`\__enumext_keyans_multi_set_vskip:` `\__enumext_keyans_multi_addvspace:` 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"*.

```
\cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
  {
    \skip_set:Nn \l__enumext_multicols_above_v_skip
      {
        \l__enumext_topsep_v_skip
      }
    \skip_set:Nn \l__enumext_multicols_below_v_skip
      {
        \l__enumext_topsep_v_skip
      }
    \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
      {
        \skip_add:Nn \l__enumext_multicols_above_v_skip { \l__enumext_parsep_i_skip }
      }
  }
\cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
  {
    \__enumext_keyans_multi_set_vskip:
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_multicols_above_v_skip
          {
            \skip_use:N \l__enumext_partopsep_v_skip
          }
        \skip_add:Nn \l__enumext_multicols_below_v_skip
          {
            \skip_use:N \l__enumext_partopsep_v_skip
```

```
            }
        }
      \par\nopagebreak
      \addvspace{ \l__enumext_multicols_above_v_skip }
    }
```

(*End of definition for* `\__enumext_keyans_multi_set_vskip:` *and* `\__enumext_keyans_multi_addvspace:`.)

13.26 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.



Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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=0pt` and to this is added the `\partopsep` parameter that comes into action according to whether TEX 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 TEX 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.

13.26.1 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` (§13.26). The space applied *"above"* the `\__enumext_mini_page` environment on the *"right side"* and *"below"* the `multicols` environment is held by the skip `\l__enumext_minipage_right_skip`. 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"*, and the boolean `\g__enumext_mini_end_bool` indicates that both are pending to be applied.

```
\skip_new:N \l__enumext_minipage_right_skip
\dim_new:N  \l__enumext_mini_prev_dim
\dim_new:N  \l__enumext_mini_gap_dim
\box_new:N  \l__enumext_mini_last_box
\dim_new:N  \g__enumext_mini_dp_dim
\skip_new:N \g__enumext_mini_end_skip
\bool_new:N \g__enumext_mini_end_bool
```

(*End of definition for* `\l__enumext_minipage_right_skip` *and others.*)

13.26.2 Definition of error messages

`wrong-miniright-place`
`wrong-miniright-use`
`wrong-miniright-starred`
`many-miniright-used`

Messages used by `\miniright` command.

```
\msg_new:nnnn { enumext } { wrong-miniright-place }
  {
    Wrong~place~for~'\token_to_str:N \miniright'~\msg_line_context:.
  }
  {
    Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
  }
\msg_new:nnnn { enumext } { wrong-miniright-use }
  {
    Wrong~use~for~'\token_to_str:N \miniright'~\msg_line_context:.
  }
```

```
    {
      '\token_to_str:N \miniright'~needs~a~key~'mini-env'.
    }
  \msg_new:nnnn { enumext } { wrong-miniright-starred }
    {
      Can't~use~'\token_to_str:N \miniright'~in~starred~environments~
      \msg_line_context:.
    }
    {
      In~'enumext*'~and~'keyans*'~use~the~keys~'mini-right'~or~
      'mini-right*'~instead.
    }
  \msg_new:nnnn { enumext } { many-miniright-used }
    {
      Can't~use~'\token_to_str:N \miniright'~more~than~once~
      \msg_line_context:.
    }
    {
      '\token_to_str:N \miniright'~can~be~used~only~once~in~each~
      environment.
    }
```

(*End of definition for* `wrong-miniright-place` *and others.*)

13.26.3 Computation of the vertical spaces for `minipage`

`\__enumext_mini_baselineskip:`

The function `\__enumext_mini_baselineskip:` adds with `\addvspace` the space that 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.

```
\cs_new_protected:Nn \__enumext_mini_baselineskip:
  {
    \dim_set:Nn \l__enumext_mini_prev_dim { \prevdepth }
    \nointerlineskip
    \dim_compare:nNnTF { \l__enumext_mini_prev_dim } < { -999pt }
      { \dim_zero:N \l__enumext_mini_gap_dim }
      {
        \dim_set:Nn \l__enumext_mini_gap_dim
          { \baselineskip - \l__enumext_mini_prev_dim - \fontcharht\font`l }
        \dim_compare:nNnT { \l__enumext_mini_gap_dim } < { \lineskip }
          { \dim_set_eq:NN \l__enumext_mini_gap_dim \lineskip }
      }
    \addvspace{ \l__enumext_mini_gap_dim }
  }
```

(*End of definition for* `\__enumext_mini_baselineskip:`.)

`\__enumext_mini_capture_end:`
`\__enumext_mini_capture_multicols:`
`\__enumext_mini_apply_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"*.

```
\cs_new_protected:Nn \__enumext_mini_capture_end:
  {
    \dim_gset:Nn \g__enumext_mini_dp_dim { \prevdepth }
    \skip_gzero:N \g__enumext_mini_end_skip
    \int_compare:nNnT { \lastnodetype } = { 11 }
      {
        \skip_gset:Nn \g__enumext_mini_end_skip { \lastskip }
        \unskip
      }
    \bool_gset_true:N \g__enumext_mini_end_bool
  }
\cs_new_protected:Nn \__enumext_mini_capture_multicols:
  {
    \int_compare:nNnTF { \lastnodetype } = { 1 }
```

```
      {
        \box_set_to_last:N \l__enumext_mini_last_box
        \dim_gset:Nn \g__enumext_mini_dp_dim { \box_dp:N \l__enumext_mini_last_box }
        \nointerlineskip
        \box_use_drop:N \l__enumext_mini_last_box
      }
      { \dim_gset:Nn \g__enumext_mini_dp_dim { \prevdepth } }
    \skip_gset_eq:NN \g__enumext_mini_end_skip \l__enumext_minipage_right_skip
    \bool_gset_true:N \g__enumext_mini_end_bool
  }
\cs_new_protected:Nn \__enumext_mini_apply_end:
  {
    \bool_if:NT \g__enumext_mini_end_bool
      {
        \par
        \dim_compare:nNnF { \g__enumext_mini_dp_dim } < { -999pt }
          { \dim_set_eq:NN \prevdepth \g__enumext_mini_dp_dim }
        \addpenalty { \@endparpenalty }
        \addvspace { \g__enumext_mini_end_skip }
        \bool_gset_false:N \g__enumext_mini_end_bool
      }
  }
```

(*End of definition for* `\__enumext_mini_capture_end:`, `\__enumext_mini_capture_multicols:`, *and* `\__enumext_mini_apply_end:`.)

13.26.4 Adjustment of vertical spaces for `minipage` in `enumext`

`\__enumext_minipage_set_skip:`
`\__enumext_minipage_add_space:`

In the `enumext` environment, 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.

First it sets the value of `\l__enumext_minipage_right_skip` equal to `\topsep`, then checks if TEX is in ⟨*vertical mode*⟩ and, if so, adds `\partopsep`.

```
\cs_new_protected:Nn \__enumext_minipage_set_skip:
  {
    \skip_set:Nn \l__enumext_minipage_right_skip
      {
        \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
      }
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_right_skip
          {
            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
          }
      }
```

Sets the value of the variable `\l__enumext_multicols_above_X_skip` equal to the value of the variable `\l__enumext_minipage_right_skip`.

```
    \skip_set_eq:cN
      { 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`.

```
    \int_compare:nNnT
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \skip_zero:N \topskip
        \skip_set_eq:Nc \multicolsep
          { l__enumext_multicols_above_ \__enumext_level: _skip }
      }
  }
```

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 ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. First the last ⟨*skip*⟩ or ⟨*kern*⟩ is removed; in ⟨*vertical mode*⟩ only `\nopagebreak` is added, while in ⟨*horizontal mode*⟩ 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:.

```
\cs_new_protected:Nn \__enumext_minipage_add_space:
  {
    \__enumext_minipage_set_skip:
    \__enumext_unskip_unkern:
    \mode_if_vertical:TF
      {
        \nopagebreak
      }
      {
        \par\nopagebreak
        \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
      }
    \__enumext_mini_baselineskip:
  }
```

(*End of definition for* __enumext_minipage_set_skip: *and* __enumext_minipage_add_space:.)

13.26.5 Adjustment of vertical spaces for minipage in keyans

__enumext_keyans_minipage_set_skip:
__enumext_keyans_minipage_add_space:

The function __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.

```
\cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
  {
    \skip_zero:N \l__enumext_minipage_right_skip
    \skip_set:Nn \l__enumext_minipage_right_skip
      {
        \l__enumext_topsep_v_skip
      }
    \mode_if_vertical:T
      {
        \skip_add:Nn \l__enumext_minipage_right_skip
          {
            \l__enumext_partopsep_v_skip
          }
      }
    \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
        \skip_zero:N \topskip
        \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
      }
  }
\cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
  {
    \__enumext_keyans_minipage_set_skip:
    \__enumext_unskip_unkern:
    \mode_if_vertical:TF
      {
        \nopagebreak
      }
      {
        \par\nopagebreak
        \skip_zero:N \l__enumext_partopsep_v_skip
      }
    \__enumext_mini_baselineskip:
  }
```

(*End of definition for* __enumext_keyans_minipage_set_skip: *and* __enumext_keyans_minipage_add_space:.)

13.26.6 Adjustment of vertical spaces for minipage in enumext* and keyans*

__enumext_mini_set_vskip_vii:
__enumext_mini_set_vskip_viii:

The functions __enumext_mini_set_vskip_vii: and __enumext_mini_set_vskip_viii: will take care of determining the space applied *"above"* the __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"*.

```
\cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
  {
```

```
    \skip_gset:Nn \g__enumext_minipage_right_skip { \l__enumext_topsep_vii_skip }
  }
\cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
  {
    \skip_gset:Nn \g__enumext_minipage_right_skip { \l__enumext_topsep_viii_skip }
  }
```

(*End of definition for* `\__enumext_mini_set_vskip_vii:` *and* `\__enumext_mini_set_vskip_viii:`.)

`\__enumext_mini_addvspace_vii:`
`\__enumext_mini_addvspace_viii:`

The functions `\__enumext_mini_addvspace_vii:` and `\__enumext_mini_addvspace_viii:` will apply the vertical space *"only above"* the `\__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 TEX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩, since `\partopsep` is equal to `0pt` in both environments.

```
\cs_new_protected:Nn \__enumext_mini_addvspace_vii:
  {
    \__enumext_mini_set_vskip_vii:
    \par\nopagebreak
    \__enumext_mini_baselineskip:
  }
\cs_new_protected:Nn \__enumext_mini_addvspace_viii:
  {
    \__enumext_mini_set_vskip_viii:
    \par\nopagebreak
    \__enumext_mini_baselineskip:
  }
```

(*End of definition for* `\__enumext_mini_addvspace_vii:` *and* `\__enumext_mini_addvspace_viii:`.)

13.26.7 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 LATEX 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.

```
\NewDocumentCommand \miniright { s }
  {
    % outside
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { \l__enumext_keyans_pic_level_int } = { 1 } }
      {
        \bool_lazy_and_p:nn
          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      }
      {
        \msg_error:nn { enumext } { wrong-miniright-place }
      }
      {
        % starred env
        \bool_lazy_and:nnTF
          { \bool_if_p:N \g__enumext_starred_bool }
          { \bool_not_p:n { \l__enumext_standard_bool } }
          {
            \msg_error:nn { enumext } { wrong-miniright-starred }
          }
          {
            % exec
            \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
              {
                \__enumext_keyans_mini_right_cmd:n {#1}
              }
```

```
                { \__enumext_mini_right_cmd:n {#1} }
            }
        }
    }
```

(*End of definition for* `\minirright`. *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 `\l__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"*.

```
\cs_new_protected:Nn \__enumext_mini_right_cmd:n
  {
    \dim_compare:nNnTF
      { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
      > { \c_zero_dim }
      {
        \__enumext_multicols_stop:
        \int_compare:nNnT
          { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
          {
            \par
            \__enumext_mini_capture_end:
          }
        \end__enumext_mini_page
        \hfill
        \__enumext_mini_page
          { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
          \par\nointerlineskip
          \addvspace { \l__enumext_minipage_right_skip }
          \bool_if:nF {#1}
            {
              \centering
            }
          \int_gzero:N \g__enumext_minipage_stat_int
      }
      { \msg_error:nn { enumext } { wrong-miniright-use } }
    % paranoia
    \RenewDocumentCommand \miniright { s }
      {
        \msg_error:nn { enumext } { many-miniright-used }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_keyans_mini_right_cmd:n
  {
    \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
      {
        \__enumext_keyans_multicols_stop:
        \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
          {
            \par
            \__enumext_mini_capture_end:
          }
        \end__enumext_mini_page
        \hfill
        \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
          \par\nointerlineskip
          \addvspace { \l__enumext_minipage_right_skip }
          \bool_if:nF {#1}
```

```
            {
              \centering
            }
          \int_gzero:N \g__enumext_minipage_stat_int
      }
      { \msg_error:nn { enumext } { wrong-miniright-use } }
    % paranoia
    \RenewDocumentCommand \miniright { s }
      {
        \msg_error:nn { enumext } { many-miniright-used }
      }
  }
```

(*End of definition for* `\__enumext_keyans_mini_right_cmd:n`.)

13.27 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.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_vspace_a_star_ #1 _bool }
    \bool_new:c { l__enumext_vspace_b_star_ #1 _bool }
  }
\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`.)

`above` `above*` `below` `below*`

Define `above`, `above*`, `below` and `below*` keys for `enumext` and `keyans` environments.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        above  .skip_set:c = { l__enumext_vspace_above_ #2 _skip },
        above  .value_required:n = true,
        above* .code:n =
          {
            \bool_set_true:c { l__enumext_vspace_a_star_ #2 _bool }
              \keys_set:nn { enumext / #1 } { above = {##1} }
          },
        above* .value_required:n = true,
        below  .skip_set:c = { l__enumext_vspace_below_ #2 _skip },
        below  .value_required:n = true,
        below* .code:n =
          {
            \bool_set_true:c { l__enumext_vspace_b_star_ #2 _bool }
              \keys_set:nn { enumext / #1 } { below = {##1} }
          },
        below* .value_required:n = true,
      }
  }
\clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

(*End of definition for* `above` *and others.*)

13.27.1 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.

```
\cs_new_protected:Nn \__enumext_vspace_above:
  {
    \skip_if_eq:nnF
      { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
      { \c_zero_skip }
```

```
      {
        \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
          {
            \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
          }
          {
            \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
          }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_vspace_below:
  {
    \skip_if_eq:nnF
      { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
      { \c_zero_skip }
      {
        \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
          {
            \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
          }
          {
            \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
          }
      }
  }
```

(*End of definition for* `\__enumext_vspace_below:`.)

13.27.2 Functions for `above` and `below` keys in `keyans`

`\__enumext_vspace_above_v:` The function `\__enumext_vspace_above_v:` applies the *vertical space above* the `keyans` environment set by the `above` and `above*` keys.

```
\cs_new_protected:Nn \__enumext_vspace_above_v:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_v_bool
          {
            \vspace*{ \l__enumext_vspace_above_v_skip }
          }
          { \vspace { \l__enumext_vspace_above_v_skip } }
      }
  }
```

(*End of definition for* `\__enumext_vspace_above_v:`.)

`\__enumext_vspace_below_v:` The function `\__enumext_vspace_below_v:` applies the *vertical space below* the `keyans` environment set by the `below*` and `below` keys.

```
\cs_new_protected:Nn \__enumext_vspace_below_v:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_v_bool
          {
            \vspace*{ \l__enumext_vspace_below_v_skip }
          }
          { \vspace { \l__enumext_vspace_below_v_skip } }
      }
  }
```

(*End of definition for* `\__enumext_vspace_below_v:`.)

©2024–2026 by Pablo González L

13.27.3 Functions for above and below keys in enumext* keyans*

__enumext_vspace_above_vii:
__enumext_vspace_above_viii:

The functions __enumext_vspace_above_vii: and __enumext_vspace_above_viii: apply the *vertical space above* the enumext* and keyans* environments set by the above and above* keys.

```
\cs_new_protected:Nn \__enumext_vspace_above_vii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
          {
            \vspace*{ \l__enumext_vspace_above_vii_skip }
          }
          { \vspace { \l__enumext_vspace_above_vii_skip } }
      }
  }
\cs_new_protected:Nn \__enumext_vspace_above_viii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
          {
            \vspace*{ \l__enumext_vspace_above_viii_skip }
          }
          { \vspace { \l__enumext_vspace_above_viii_skip } }
      }
  }
```

(*End of definition for* __enumext_vspace_above_vii: *and* __enumext_vspace_above_viii:.)

__enumext_vspace_below_vii:
__enumext_vspace_below_viii:

The functions __enumext_vspace_below_vii: and __enumext_vspace_below_viii: apply the *vertical space below* the enumext* and keyans* environments set by the below* and below keys.

```
\cs_new_protected:Nn \__enumext_vspace_below_vii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
          {
            \vspace*{ \l__enumext_vspace_below_vii_skip }
          }
          { \vspace { \l__enumext_vspace_below_vii_skip } }
      }
  }
\cs_new_protected:Nn \__enumext_vspace_below_viii:
  {
    \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
      {
        \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
          {
            \vspace*{ \l__enumext_vspace_below_viii_skip }
          }
          { \vspace { \l__enumext_vspace_below_viii_skip } }
      }
  }
```

(*End of definition for* __enumext_vspace_below_vii: *and* __enumext_vspace_below_viii:.)

13.28 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*⟩.

13.28.1 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 (§13.28).

```
\tl_new:N \l__enumext_series_name_tl
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_resume_series_ #1 _bool   }
    \bool_new:c { l__enumext_resume_star_key_ #1 _bool }
    \tl_new:c   { g__enumext_resume_last_keys_ #1 _tl  }
  }
\clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

©2024–2026 by Pablo González L

(*End of definition for* `\l__enumext_series_name_tl` *and others.*)

13.28.2 Definition of error messages

unknown-series-starred
unknown-series-standard

Messages used by `resume` key.

```
\msg_new:nnnn { enumext } { unknown-series-starred }
  {
    The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
    enumext*~environment~\msg_line_context:.
  }
  {
    Define~the~series~with~the~key~'series'~in~a~previous~
    'enumext*'~environment~and~check~the~name.
  }
\msg_new:nnnn { enumext } { unknown-series-standard }
  {
    The~series~'#1'~for~the~resume~key~does~not~exist~at~level~
    \int_use:N \l__enumext_level_int
    \c_space_tl of~enumext~environment~\msg_line_context:.
  }
  {
    Define~the~series~with~the~key~'series'~at~that~level~in~
    a~previous~'enumext'~environment~and~check~the~name.
  }
```

(*End of definition for* `unknown-series-starred` *and* `unknown-series-standard`.)

13.28.3 Definition of keys for `series`, `resume` and `resume*`

series
resume
resume*

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 `\l__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*`.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        series  .str_set:N = \l__enumext_series_name_str,
        series  .value_required:n = true,
        resume  .code:n =
          {
            \bool_if:NF \l__enumext_print_keyans_cmd_bool
              {
                \tl_set:Nn \l__enumext_series_name_tl {##1}
                \tl_if_empty:NTF \l__enumext_series_name_tl
                  {
                    \bool_set_true:c { l__enumext_resume_count_ #2 _bool }
                    \bool_set_eq:Nc
                      \l__enumext_resume_count_bool
                      { l__enumext_resume_count_ #2 _bool }
                  }
                  {
                    \__enumext_resume:n {##1}
                  }
              }
          },
        resume* .code:n =
          {
            \bool_if:NF \l__enumext_print_keyans_cmd_bool
              {
                \bool_set_true:c { l__enumext_resume_count_ #2 _bool }
                \bool_set_eq:Nc
                  \l__enumext_resume_count_bool { l__enumext_resume_count_ #2 _bool }
                \bool_set_true:c { l__enumext_resume_star_key_ #2 _bool }
                \__enumext_resume_star:
              }
          },
```

©2024–2026 by Pablo González L

```
      resume* .value_forbidden:n = true,
    }
  }
\clist_map_inline:nn
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},{enumext*}{vii},
  }
  { \__enumext_tmp:nn #1 }
```

(*End of definition for* `series`, `resume`, *and* `resume*`.)

13.28.4 Internal function to save counter and integer values

__enumext_standard_save_counter:
__enumext_standard_save_counter_aux:
__enumext_starred_save_counter:
__enumext_starred_save_counter_aux:

The `\__enumext_standard_save_counter:` and `\__enumext_starred_save_counter:` functions save the last counter value as follows. If the key `series`={⟨*series name*⟩} has been passed, the value goes to `\g__enumext_series_#1_X_int` with that name. If none of the keys `series`, `resume`={⟨*series name*⟩} and `save-ans` has been passed, which is the case of the key `resume` *without value*, it goes to `\c@ __enumext_resume_X_int`. If the key `resume`={⟨*series name*⟩} has been passed, it goes to `\g__enumext_series_#1_X_int` with that name. In the *"first level"* it also goes 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 (§13.44) and the `enumext*` environment definition (§13.49).

```
\cs_new_protected:Nn \__enumext_standard_save_counter:
  {
    \bool_if:NTF \g__enumext_standard_bool
      {
        \__enumext_standard_save_counter_aux:
        \int_compare:nNnT { \l__enumext_level_int } = { 1 }
          {
            \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
              {
                \int_gset_eq:cN
                  { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
              }
          }
      }
      {
        \__enumext_standard_save_counter_aux:
      }
  }
\cs_new_protected:Nn \__enumext_standard_save_counter_aux:
  {
    \str_if_empty:NF \l__enumext_series_name_str
      {
        \int_gset_eq:cc
          { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
          { c@enumX \__enumext_level: }
      }
    \tl_if_empty:NTF \l__enumext_series_name_tl
      {
        \str_if_empty:NT \l__enumext_series_name_str
          {
            \tl_if_empty:NT \l__enumext_store_name_tl
              {
                \int_gset_eq:cc
                  { c@ __enumext_resume_ \__enumext_level: _int }
                  { c@enumX \__enumext_level: }
              }
          }
      }
      {
        \int_if_exist:cT
          { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
          {
            \int_gset_eq:cc
              { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
              { c@enumX \__enumext_level: }
```

```
            }
        }
    }
\cs_new_protected:Nn \__enumext_starred_save_counter:
    {
      \bool_if:NTF \g__enumext_starred_bool
        {
          \__enumext_starred_save_counter_aux:
          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
            {
              \int_gset_eq:cN
                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
            }
        }
        {
          \__enumext_starred_save_counter_aux:
        }
    }
\cs_new_protected:Nn \__enumext_starred_save_counter_aux:
    {
      \str_if_empty:NF \l__enumext_series_name_str
        {
          \int_gset_eq:cN
            { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
        }
      \tl_if_empty:NTF \l__enumext_series_name_tl
        {
          \str_if_empty:NT \l__enumext_series_name_str
            {
              \tl_if_empty:NT \l__enumext_store_name_tl
                {
                  \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
                }
            }
        }
        {
          \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
            {
              \int_gset_eq:cN
                { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
            }
        }
    }
```

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

13.28.5 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.

```
\cs_new_protected:Nn \__enumext_resume_counter:
    {
      \cs_set:Nn \__enumext_tmp:n
        {
          \bool_if:cT { l__enumext_resume_count_ \int_to_roman:n {##1} _bool }
            {
              \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
                {
                  \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
                }
            }
        }
      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
        {
          \bool_lazy_and:nnTF
            { \bool_if_p:N \l__enumext_standard_first_bool }
            { \bool_if_p:N \l__enumext_store_active_bool }
            {
              \int_set:Nn \l__enumext_start_i_int
                {
```

```
                \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
              }
          }
          {
            \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \bool_lazy_and:nnTF
          { \bool_if_p:N \l__enumext_starred_first_bool }
          { \bool_if_p:N \l__enumext_store_active_bool }
          {
            \int_set:Nn \l__enumext_start_vii_int
              {
                \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
              }
          }
          {
            \bool_if:NT \l__enumext_resume_count_vii_bool
              {
                \int_set:Nn \l__enumext_start_vii_int
                  {
                    \int_use:c { c@ __enumext_resume_vii_int } + 1
                  }
              }
          }
      }
  }
```

(*End of definition for* `\__enumext_resume_counter:`.)

13.28.6 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

```
\cs_new:Nn \__enumext_filter_series:n
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_series_key:n
          \__enumext_filter_series_pair:nn {#1}
      }
  }
```

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.

```
\cs_new:Nn \__enumext_filter_series_key:n
  {
    \str_case:nnF {#1}
      {
        { resume } { } { resume* } { } { reset } { } { reset* } { }
          { base-fix } { }
      }
      { , { \exp_not:n {#1} } }
  }
```

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.

```
\cs_new:Nn \__enumext_filter_series_pair:nn
  {
    \str_case:nnF {#1}
      {
        { series } { } { resume } { } { start } { }
        { start* } { }  { save-ans } { } { save-key } { }
      }
      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
  }
```

(*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*⟩.

```
\cs_new_protected:Nn \__enumext_save_last_keys:n
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \bool_if:cF { l__enumext_resume_series_ \__enumext_level: _bool }
          {
            \bool_if:cF { l__enumext_resume_star_key_ \__enumext_level: _bool }
              {
                \tl_gclear:c { g__enumext_resume_last_keys_ \__enumext_level: _tl }
                \tl_gset:ce
                  { g__enumext_resume_last_keys_ \__enumext_level: _tl }
                  { \__enumext_filter_series:n {#1} }
              }
          }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \bool_if:NF \l__enumext_resume_series_vii_bool
          {
            \tl_gclear:N \g__enumext_resume_last_keys_vii_tl
            \tl_gset:Ne \g__enumext_resume_last_keys_vii_tl
              { \__enumext_filter_series:n {#1} }
          }
      }
  }
```

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*` (§13.16).

```
\cs_new_protected:Nn \__enumext_resume_last_counter:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_resume_count_bool }
      { \tl_if_empty_p:N \l__enumext_series_name_tl }
      {
        \__enumext_resume_counter:
      }
  }
```

(*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 (§13.44) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.49).

```
\cs_new_protected:Nn \__enumext_parse_series:n
  {
    \str_if_empty:NTF \l__enumext_series_name_str
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
```

```
            {
              \__enumext_save_last_keys:n {#1}
              \__enumext_resume_last_counter:
            }
          \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
            {
              \__enumext_save_last_keys:n {#1}
              \__enumext_resume_last_counter:
            }
        }
        {
          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
            {
              \tl_gclear_new:c
                { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
              \tl_gset:ce
                { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
                { \__enumext_filter_series:n {#1} }
              \int_if_exist:cF
                { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
                {
                  \int_new:c
                { g__enumext_series_ \l__enumext_series_name_str
                  _ \__enumext_level: _int }
                }
            }
          \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
            {
              \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
              \tl_gset:ce
                { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
                { \__enumext_filter_series:n {#1} }
              \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
                {
                  \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
                }
            }
        }
    }
```

(*End of definition for* `\__enumext_parse_series:n`.)

13.28.7 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.

```
\cs_new_protected:Nn \__enumext_resume:n
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
          {
            \__enumext_resume_series:n {#1}
            \keys_set:ev { enumext /
              level-\int_use:N \l__enumext_level_int }
              { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
          }
          {
            \msg_error:nnn { enumext } { unknown-series-standard } {#1}
          }
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
          {
            \__enumext_resume_series:n {#1}
            \keys_set:nv { enumext / enumext* }
              { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
          }
```

```
2203            {
2204              \msg_error:nnn { enumext } { unknown-series-starred } {#1}
2205            }
2206        }
2207    }
```

(*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:`.

```
2208 \cs_new_protected:Nn \__enumext_resume_series:n
2209   {
2210     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2211       {
2212         \bool_set_true:c { l__enumext_resume_series_ \__enumext_level: _bool }
2213         \tl_clear:N \l__enumext_series_name_tl
2214         \tl_set:Nn \l__enumext_series_name_tl {#1}
2215         \__enumext_resume_integer_series:
2216       }
2217     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2218       {
2219         \bool_set_true:N \l__enumext_resume_series_vii_bool
2220         \tl_clear:N \l__enumext_series_name_tl
2221         \tl_set:Nn \l__enumext_series_name_tl {#1}
2222         \__enumext_resume_integer_series:
2223       }
2224   }
```

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.

```
2225 \cs_new_protected:Nn \__enumext_resume_integer_series:
2226   {
2227     \cs_set:Nn \__enumext_tmp:n
2228       {
2229         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _
2230           \int_to_roman:n {##1} _int }
2231           {
2232             \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2233               {
2234                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _
2235                   \int_to_roman:n {##1} _int } + 1
2236               }
2237           }
2238       }
2239     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2240       {
2241         \bool_lazy_and:nnTF
2242           { \bool_if_p:N \l__enumext_standard_first_bool }
2243           { \bool_if_p:N \l__enumext_store_active_bool }
2244           {
2245             \int_set:Nn \l__enumext_start_i_int
2246               {
2247                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2248               }
2249           }
2250           {
2251             \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
2252           }
2253       }
2254     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2255       {
2256         \bool_lazy_and:nnTF
2257           { \bool_if_p:N \l__enumext_starred_first_bool }
2258           { \bool_if_p:N \l__enumext_store_active_bool }
2259           {
2260             \int_set:Nn \l__enumext_start_vii_int
2261               {
```

```
                \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
              }
          }
          {
            \int_set:Nn \l__enumext_start_vii_int
              {
                \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
              }
          }
      }
  }
```

(*End of definition for* `\__enumext_resume_series:n` *and* `\__enumext_resume_integer_series:`.)

13.28.8 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.

```
\cs_new_protected:Nn \__enumext_resume_star:
  {
    \cs_set:Nn \__enumext_tmp:n
      {
        \tl_if_empty:cTF { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
          {
            \__enumext_resume_counter:
          }
          {
            \__enumext_resume_counter:
            \keys_set:ev
              { enumext / level-\int_use:N \l__enumext_level_int }
              { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
      }
    \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
      {
        \tl_if_empty:NTF \g__enumext_resume_last_keys_vii_tl
          {
            \__enumext_resume_counter:
          }
          {
            \__enumext_resume_counter:
            \keys_set:nV { enumext / enumext* } \g__enumext_resume_last_keys_vii_tl
          }
      }
  }
```

(*End of definition for* `\__enumext_resume_star:`.)

13.29 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.

```
\msg_new:nnnn { enumext } { out-of-range }
  {
    The~number~must~be~exactly~1,~2,~3~or~4.
  }
  {
    Received:~'#1'.
  }
```

(*End of definition for* `out-of-range`.)

`\__enumext_standard_reset:nn`
`\__enumext_starred_reset:n`
`\__enumext_reset_count_resume:nn`
`\__enumext_reset_count_resume:en`
`\__enumext_reset_count_resume_all:n`
`\__enumext_reset_count_resume_levels:n`

Internal functions used by the `\resetenumext` command.

```
\cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
  {
    \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } %{#2}
  }
\cs_new_protected:Nn \__enumext_starred_reset:n
  {
    \__enumext_reset_count_resume:nn { vii } {#1}
  }
\cs_new_protected:Nn \__enumext_reset_count_resume:nn
  {
    \counterwithin*{enumX #1}{#2}
    \counterwithin*{__enumext_resume_ #1 _int}{#2}
  }
\cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
\cs_new_protected:Nn \__enumext_reset_count_resume_all:n
  {
    \clist_map_inline:nn { i,ii,iii,iv,vii }
      {
        \__enumext_reset_count_resume:nn {##1} {#1}
      }
  }
\cs_new_protected:Nn \__enumext_reset_count_resume_levels:n
  {
    \clist_map_inline:nn { i,ii,iii,iv }
      {
        \__enumext_reset_count_resume:nn {##1} {#1}
      }
  }
```

(*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 {⟨*some counter*⟩}. 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 {⟨*some counter*⟩} in '`#3`'.

```
\NewDocumentCommand \resetenumext { s o m }
  {
    \bool_if:nTF {#1}
      {
        \__enumext_reset_count_resume_all:n {#3}
      }
      {
        \tl_if_novalue:nTF {#2}
          {
            \__enumext_reset_count_resume_levels:n {#3}
          }
          {
            \str_if_eq:nnTF {#2} { * }
              { \__enumext_starred_reset:n {#3} }
              {
                \bool_lazy_and:nnTF
                  { \int_compare_p:nNn {#2} > { 0 } }
                  { \int_compare_p:nNn {#2} < { 5 } }
                  { \__enumext_standard_reset:nn {#2} {#3} }
                  {
                    \msg_error:nne { enumext } { out-of-range }
                      { \int_eval:n {#2} }
                  }
              }
          }
      }
  }
```

(*End of definition for* `\resetenumext`. *This function is documented on page 13.*)

13.30 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 ⟨*keys*⟩ that adjust the internal variables if necessary.

reset
reset*

Defines the keys reset and reset* for the *"all levels"* of enumext and enumext*.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / #1 }
      {
        reset  .code:n = { \__enumext_standard_reset_key: },
        reset  .value_forbidden:n = true,
        reset* .code:n = { \__enumext_standard_reset_key_star: },
        reset* .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
\keys_define:nn { enumext / enumext* }
  {
    reset  .code:n = { \__enumext_starred_reset_key: },
    reset  .value_forbidden:n = true,
    reset* .code:n = { \__enumext_starred_reset_key: },
    reset* .value_forbidden:n = true,
  }
```

(*End of definition for* reset *and* reset*.*)

13.30.1 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.

```
\cs_new_protected:Nn \__enumext_standard_reset_key:
  {
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int }
          _int }
          {
            \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int }
              _int }
          }
      }
  }
```

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*.

```
\cs_new_protected:Nn \__enumext_standard_reset_key_star:
  {
    \cs_set:Nn \__enumext_tmp:n
      {
        \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
          {
            \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
          }
      }
    \int_compare:nNnT { \l__enumext_level_int } > { 0 }
      {
        \int_step_function:nnN { \l__enumext_level_int } { 4 } \__enumext_tmp:n
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_starred_reset_key:
  {
    \int_gzero:c { c@ __enumext_resume_vii_int }
  }
```

(*End of definition for* __enumext_standard_reset_key:*,* __enumext_standard_reset_key_star:*, and* __enumext_starred_reset_key:*.*)

13.31 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.

13.31.1 Definition of error messages

save-ans-empty

Message used by save-ans key.

```
\msg_new:nnnn { enumext } { save-ans-empty }
  {
    Key~'save-ans'~needs~a~value~in~'#1'~\msg_line_context:.
  }
  {
    Give~a~name~to~the~storage,~for~example~'save-ans=name'.
  }
```

(*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.

```
\msg_new:nnn { enumext } { missing-starred }
  {
    Missing~'\c_backslash_str #1*'~#2.
  }
\msg_new:nnn { enumext } { many-starred }
  {
    Many~'\c_backslash_str #1*'~#2.
  }
```

(*End of definition for* missing-starred *and* many-starred.)

13.31.2 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.

```
\msg_new:nnn { enumext } { save-ans-log }
  {
    Start~'#1'~with~'save-ans=#2'~\msg_line_context:.
  }
\msg_new:nnn { enumext } { save-ans-log-hook }
  {
    Stop~'#1'~with~'save-ans=#2'~\msg_line_context:.
  }
```

(*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.

```
\msg_new:nnn { enumext } { store-prop }
  {
    Creating~\c_backslash_str g__enumext_ #1 _prop~\msg_line_context:.
  }
\msg_new:nnn { enumext } { store-seq }
  {
    Creating~\c_backslash_str g__enumext_ #1 _seq~\msg_line_context:.
  }
\msg_new:nnn { enumext } { store-int }
  {
    Creating~\c_backslash_str g__enumext_resume_ #1 _int~\msg_line_context:.
  }
```

(*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.

```
\msg_new:nnn { enumext } { items-same-answer }
  {
    Answers~in~'#1'~(#2):~OK,~items~=~answers.\\
    Started~#3~and~closed~\msg_line_context:.
  }
\msg_new:nnn { enumext } { item-greater-answer }
  {
    Answers~in~'#1'~(#2):~NOT~OK,~items~>~answers.\\
    Started~#3~and~closed~\msg_line_context:.
  }
\msg_new:nnn { enumext } { item-less-answer }
```

©2024–2026 by Pablo González L

```
  {
    Answers~in~'#1'~(#2):~NOT~OK,~items~<~answers.\\
    Started~#3~and~closed~\msg_line_context:.
  }
```

(*End of definition for* `items-same-answer`, `item-greater-answer`, *and* `item-less-answer`.)

13.31.3 Setting save-ans key

save-ans

Defines the key save-ans only for the *"first level"* of enumext and enumext*.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / #1 }
      {
        save-ans .code:n = { \__enumext_storing_set:n {##1} },
        save-ans .value_required:n = true,
      }
  }
\clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `save-ans`.)

13.31.4 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:.

```
\cs_new_protected:Nn \__enumext_start_save_ans_msg:
  {
    \msg_info:nnVV { enumext } { save-ans-log }
      \g__enumext_envir_name_tl \l__enumext_store_name_tl
  }
\cs_new_protected:Nn \__enumext_stop_save_ans_msg:
  {
    \msg_info:nnVV { enumext } { save-ans-log-hook }
      \g__enumext_envir_name_tl \g__enumext_store_name_tl
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_storing_set:n
  {
    \tl_set:Ne \l__enumext_store_name_tl {#1}
    \tl_if_empty:NTF \l__enumext_store_name_tl
      {
        \bool_lazy_or:nnT
          { \l__enumext_standard_first_bool } { \l__enumext_starred_first_bool }
          {
            \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
          }
      }
      {
        \bool_lazy_or:nnT
          { \l__enumext_standard_first_bool } { \l__enumext_starred_first_bool }
          {
            \__enumext_start_save_ans_msg:
            \__enumext_storing_exec:
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_storing_exec:
  {
    \bool_set_true:N \l__enumext_store_active_bool
    \bool_set_true:N \l__enumext_check_answers_bool
    \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*.

```
    \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
      {
        \msg_info:nnV { enumext } { store-prop } \l__enumext_store_name_tl
        \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
      }
    \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
      {
        \msg_info:nnV { enumext } { store-seq } \l__enumext_store_name_tl
        \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
      }
    \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
      {
        \msg_info:nnV { enumext } { store-int } \l__enumext_store_name_tl
        \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
      }
  }
```

(*End of definition for* __enumext_storing_set:n *and* __enumext_storing_exec:.)

13.31.5 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 \g__enumext_item_number_int, associated to the sum of each \item and \item* in the environment, 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.

13.31.6 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.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / #1 }
      {
        check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
        check-ans .initial:n = false,
        check-ans .value_required:n = true,
        no-store  .code:n =
          {
            \bool_set_false:N \l__enumext_check_answers_bool
            \bool_set_false:N \l__enumext_check_ans_key_bool
          },
        no-store  .value_forbidden:n = true,
      }
  }
\clist_map_inline:nn
  {
    level-1, level-2, level-3, level-4, enumext*
  }
  { \__enumext_tmp:n {#1} }
```

(*End of definition for* check-ans *and* no-store.)

13.31.7 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

The function __enumext_check_ans_active: first checks if the key save-ans is active, that is, if the variable \l__enumext_store_name_tl is not empty. If so, it checks the state of the variable \l__enumext_check_answers_bool, handled by the key no-store. Only if it is true, i.e. the key no-store is not active, it executes the function __enumext_check_ans_level:.

```
\cs_new_protected:Nn \__enumext_check_ans_active:
  {
    \tl_if_empty:NF \l__enumext_store_name_tl
      {
        \bool_if:NT \l__enumext_check_answers_bool
          {
            \__enumext_check_ans_level:
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_check_ans_level:
  {
    \int_case:nn { \l__enumext_level_int }
      {
        { 1 }
          {
            \bool_lazy_all:nT
              {
                { \bool_if_p:N \g__enumext_starred_bool }
                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
              }
              {
                \int_gdecr:N \g__enumext_item_number_int
                \bool_set_false:N \l__enumext_item_number_bool
              }
          }
        { 2 }
          {
            \int_gdecr:N \g__enumext_item_number_int
            \bool_set_false:N \l__enumext_item_number_bool
          }
        { 3 }
          {
```

```
2577              \int_gdecr:N \g__enumext_item_number_int
2578              \bool_set_false:N \l__enumext_item_number_bool
2579            }
2580          { 4 }
2581            {
2582              \int_gdecr:N \g__enumext_item_number_int
2583              \bool_set_false:N \l__enumext_item_number_bool
2584            }
2585        }
```

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.

```
2586      \int_case:nn { \l__enumext_level_h_int }
2587        {
2588          { 1 }
2589            {
2590              \bool_lazy_all:nT
2591                {
2592                  { \bool_if_p:N \g__enumext_standard_bool }
2593                  { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2594                }
2595                {
2596                  \int_gdecr:N \g__enumext_item_number_int
2597                  \bool_set_false:N \l__enumext_item_number_bool
2598                }
2599            }
2600        }
2601    }
```

(*End of definition for* `\__enumext_check_ans_active:` *and* `\__enumext_check_ans_level:`.)

`\__enumext_check_ans_key_hook:`

Only if the key check-ans is active, 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.

```
2602 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2603   {
2604     \bool_lazy_and:nnT
2605       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2606       { \bool_if_p:N \g__enumext_standard_bool }
2607       {
2608         \bool_gset_true:N \g__enumext_check_ans_key_bool
2609       }
2610     \bool_lazy_and:nnT
2611       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2612       { \bool_if_p:N \g__enumext_starred_bool }
2613       {
2614         \bool_gset_true:N \g__enumext_check_ans_key_bool
2615       }
2616   }
```

(*End of definition for* `\__enumext_check_ans_key_hook:`.)

`\__enumext_item_answer_diff:`

The function __enumext_item_answer_diff: will be passed to __enumext_execute_after_env: and sets the value of the integer \g__enumext_item_answer_diff_int. This value is used by the function __enumext_check_ans_show: for the key save-ans and by __enumext_check_ans_log: for the internal *"check answer"* mechanism.

```
2617 \cs_new_protected:Nn \__enumext_item_answer_diff:
2618   {
2619     \int_gset:Nn \g__enumext_item_answer_diff_int
2620       {
2621         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2622       }
2623   }
```

(*End of definition for* `\__enumext_item_answer_diff:`.)

`\__enumext_check_ans_show:`
`\__enumext_check_ans_msg_less:`
`\__enumext_check_ans_msg_same_ok:`
`\__enumext_check_ans_msg_greater:`

The function __enumext_check_ans_show: will be executed within __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 __enumext_item_answer_diff:.

```
\cs_new_protected:Nn \__enumext_check_ans_show:
  {
    \int_case:nn { \g__enumext_item_answer_diff_int }
      {
        { -1 }{ \__enumext_check_ans_msg_less:    }
        {  0 }{ \__enumext_check_ans_msg_same_ok: }
        {  1 }{ \__enumext_check_ans_msg_greater: }
      }
  }
\cs_new_protected:Nn \__enumext_check_ans_msg_less:
  {
    \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
  {
    \msg_info:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_msg_greater:
  {
    \msg_warning:nneee { enumext } { item-greater-answer }
      { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
```

(*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 __enumext_execute_after_env: when the key check-ans is not active, that is, when \g__enumext_check_ans_key_bool is false, and will write in the log the appropriate message according to the value of \g__enumext_item_answer_diff_int set by __enumext_item_answer_diff:.

```
\cs_new_protected:Nn \__enumext_check_ans_log:
  {
    \int_case:nn { \g__enumext_item_answer_diff_int }
      {
        { -1 }{ \__enumext_check_ans_log_msg_less:    }
        {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
        {  1 }{ \__enumext_check_ans_log_msg_greater: }
      }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
  {
    \msg_info:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
  {
    \msg_info:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
\cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
  {
    \msg_info:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
      { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
  }
```

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

13.31.8 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.

```
\cs_new_protected:Nn \__enumext_check_starred_cmd:n
  {
    \int_compare:nNnT
      { \g__enumext_check_starred_cmd_int } = { 0 }
```

```
      {
        \msg_warning:nnnV
          { enumext } { missing-starred }{#1} \l__enumext_check_start_line_env_tl
      }
    \int_compare:nNnT
      { \g__enumext_check_starred_cmd_int } > { 1 }
      {
        \msg_warning:nnnV
          { enumext } { many-starred }{#1} \l__enumext_check_start_line_env_tl
      }
    \int_gzero:N \g__enumext_check_starred_cmd_int
    \tl_clear:N \l__enumext_check_start_line_env_tl
  }
```

(*End of definition for* `\__enumext_check_starred_cmd:n`.)

13.32 Keys and functions associated with storage

13.32.1 Definition of variables

`\l__enumext_store_save_key_X_tl`
`\l__enumext_store_save_key_X_bool`

Internal variables used by `save-key` key (§13.32.4) and the *"storing structure"*.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \tl_new:c   { l__enumext_store_save_key_ #1 _tl   }
    \bool_new:c { l__enumext_store_save_key_ #1 _bool }
  }
\clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\l__enumext_store_save_key_X_tl` *and* `\l__enumext_store_save_key_X_bool`.)

13.32.2 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.

`mark-ans*`
`mark-pos*`
`mark-sep*`
`wrap-ans*`
`wrap-opt`
`save-sep`
`show-ans`
`show-pos`

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.

```
\cs_set_protected:Nn \__enumext_tmp:nn
  {
    \keys_define:nn { enumext / #1 }
      {
        mark-ans* .tl_set:c = { l__enumext_mark_answer_sym_ #2 _tl },
        mark-ans* .initial:n = \textasteriskcentered,
        mark-ans* .value_required:n = true,
        mark-pos* .choices:nn =
          { left , right , center }
          {
            \int_case:nn { \l_keys_choice_int }
              {
                {1} { \str_set:cn { l__enumext_mark_position_ #2 _str } { l } }
                {2} { \str_set:cn { l__enumext_mark_position_ #2 _str } { r } }
                {3} { \str_set:cn { l__enumext_mark_position_ #2 _str } { c } }
              }
          },
        mark-pos* .initial:n = right,
        mark-pos* .value_required:n = true,
        mark-pos* / unknown .code:n =
          \msg_error:nneee { enumext } { unknown-choice }
            { mark-pos* } { left,~right,~center } { \exp_not:n {##1} },
        mark-sep* .dim_set:c = { l__enumext_mark_sym_sep_ #2 _dim },
        mark-sep* .value_required:n = true,
        wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_ #2 :n } ##1,
        wrap-ans* .value_required:n = true,
        wrap-opt  .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_ #2 :n } ##1,
        wrap-opt  .initial:n = [{##1}],
        wrap-opt  .value_required:n = true,
        save-sep  .tl_set:c = { l__enumext_store_keyans_item_opt_sep_ #2 _tl },
        save-sep  .initial:n = {,~},
        save-sep  .value_required:n = true,
        show-ans  .bool_set:N = \l__enumext_show_answer_bool,
        show-ans  .initial:n = false,
```

```
      show-ans  .value_required:n = true,
      show-pos  .bool_set:N = \l__enumext_show_position_bool,
      show-pos  .initial:n = false,
      show-pos  .value_required:n = true,
    }
  }
\clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* `mark-ans*` *and others.*)

`mark-ref` `save-ref` `show-ans` `show-pos` `mark-ans` `mark-pos` `mark-sep` `wrap-ans` `mark-ans*` `mark-pos*` `mark-sep*` `wrap-ans*` `wrap-opt` `save-sep`

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*`.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / #1 }
      {
        mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
        mark-ref .initial:n = \textreferencemark,
        mark-ref .value_required:n = true,
        save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
        save-ref .initial:n = false,
        save-ref .value_required:n = true,
        show-ans .bool_set:N = \l__enumext_show_answer_bool,
        show-ans .initial:n = false,
        show-ans .value_required:n = true,
        show-pos .bool_set:N = \l__enumext_show_position_bool,
        show-pos .initial:n = false,
        show-pos .value_required:n = true,
        mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
        mark-ans .initial:n = \textasteriskcentered,
        mark-ans .value_required:n = true,
        mark-sep .dim_set:N = \l__enumext_mark_sym_sep_dim,
        mark-sep .value_required:n = true,
        mark-pos .choices:nn =
          { left , right , center }
          {
            \int_case:nn { \l_keys_choice_int }
              {
                {1} { \str_set:Nn \l__enumext_mark_position_str { l } }
                {2} { \str_set:Nn \l__enumext_mark_position_str { r } }
                {3} { \str_set:Nn \l__enumext_mark_position_str { c } }
              }
          },
        mark-pos .initial:n = right,
        mark-pos .value_required:n = true,
        mark-pos / unknown .code:n =
          \msg_error:nneee { enumext } { unknown-choice }
            { mark-pos } { left,~right,~center } { \exp_not:n {##1} },

        wrap-ans  .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
        wrap-ans  .initial:n =
                    {
                      \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep
                        -2\fboxrule}}{##1}}
                    },
        wrap-ans  .value_required:n = true,
        mark-ans* .code:n =
          {
            \keys_set:nn { enumext / keyans  } { mark-ans* = {##1} }
            \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
          },
        mark-ans* .value_required:n = true,
        mark-pos* .code:n =
          {
            \keys_set:nn { enumext / keyans  } { mark-pos* = {##1} }
            \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
          },
        mark-pos* .value_required:n = true,
        mark-sep* .code:n =
```

```
2794            {
2795              \keys_set:nn { enumext / keyans  } { mark-sep* = {##1} }
2796              \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2797            },
2798          mark-sep* .value_required:n = true,
2799          wrap-ans* .code:n =
2800            {
2801              \keys_set:nn { enumext / keyans  } { wrap-ans* = {##1} }
2802              \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2803            },
2804          wrap-ans* .value_required:n = true,
2805          wrap-opt  .code:n =
2806            {
2807              \keys_set:nn { enumext / keyans  } { wrap-opt = {##1} }
2808              \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2809            },
2810          wrap-opt  .value_required:n = true,
2811          save-sep  .code:n =
2812            {
2813              \keys_set:nn { enumext / keyans  } { save-sep = {##1} }
2814              \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2815            },
2816          save-sep  .value_required:n = true,
2817        }
2818    }
2819  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `mark-ref` *and others.*)

13.32.3 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 `\l__enumext_store_save_key_X_bool` is false, that is, the key `save-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered ⟨*keys*⟩.

```
2820  \cs_new_protected:Nn \__enumext_store_active_keys:n
2821    {
2822      \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2823        {
2824          \tl_clear:c { l__enumext_store_save_key_ \__enumext_level: _tl }
2825          \tl_set:ce
2826            { l__enumext_store_save_key_ \__enumext_level: _tl }
2827            { \__enumext_filter_save_key:n {#1} }
2828        }
2829    }
2830  \cs_new_protected:Nn \__enumext_store_active_keys_vii:n
2831    {
2832      \bool_if:NF \l__enumext_store_save_key_vii_bool
2833        {
2834          \tl_clear:N \l__enumext_store_save_key_vii_tl
2835          \tl_set:Ne \l__enumext_store_save_key_vii_tl
2836            { \__enumext_filter_save_key:n {#1} }
2837        }
2838    }
```

(*End of definition for* `\__enumext_store_active_keys:n` *and* `\__enumext_store_active_keys_vii:n`.)

13.32.4 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 `\l__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.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / enumext* }
      {
        save-key .code:n = { \__enumext_parse_save_key_vii:n {##1} },
        save-key .value_required:n = true,
      }
    \keys_define:nn { enumext / #1 }
      {
        save-key .code:n = { \__enumext_parse_save_key:n {##1} },
        save-key .value_required:n = true,
      }
  }
\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 \l__enumext_store_save_key_X_tl for enumext and enumext*.

```
\cs_new_protected:Nn \__enumext_parse_save_key:n
  {
    \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
    \tl_set:ce
      { l__enumext_store_save_key_ \__enumext_level: _tl }
      { \__enumext_filter_save_key:n {#1} }
  }
\cs_new_protected:Nn \__enumext_parse_save_key_vii:n
  {
    \bool_set_true:N \l__enumext_store_save_key_vii_bool
    \tl_clear:N \l__enumext_store_save_key_vii_tl
    \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
  }
```

(*End of definition for* __enumext_parse_save_key:n *and* __enumext_parse_save_key_vii:n.)

13.32.5 Internal functions to store optional arguments

__enumext_filter_save_key:n
__enumext_filter_save_key_key:n
__enumext_filter_save_key_pair:nn

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.

```
\cs_new:Nn \__enumext_filter_save_key:n
  {
    \use:e
      {
        \keyval_parse:NNn
          \__enumext_filter_save_key_key:n
          \__enumext_filter_save_key_pair:nn {#1}
      }
  }
```

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.

```
\cs_new:Nn \__enumext_filter_save_key_key:n
  {
    \str_case:nnF {#1}
      {
        { resume } { } { resume* } { } { reset } { } { reset* } { }
          { no-store } { } { base-fix } { }
      }
      { , { \exp_not:n {#1} } }
  }
```

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.

```
\cs_new:Nn \__enumext_filter_save_key_pair:nn
  {
```

```
    \str_case:nnF {#1}
      {
        { 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 } { } { mini-right* } { } { }
      }
      { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
  }
```

(*End of definition for* `\__enumext_filter_save_key:n`, `\__enumext_filter_save_key_key:n`, *and* `\__enumext_filter_save_key_pair:nn`.)

13.32.6 Function for storing content in prop list

`\__enumext_store_addto_prop:n`
`\__enumext_store_addto_prop:V`

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.

```
\cs_new_protected:Nn \__enumext_store_addto_prop:n
  {
    \tl_if_empty:NF \l__enumext_store_name_tl
      {
        \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
          {
            \int_eval:n
              { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
          }
          {#1}
      }
  }
\cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

(*End of definition for* `\__enumext_store_addto_prop:n`.)

13.32.7 Function for storing content in sequence

`\__enumext_store_addto_seq:n`
`\__enumext_store_addto_seq:v`
`\__enumext_store_addto_seq:V`

The function `\__enumext_store_addto_seq:n` stores the {⟨*content*⟩} 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 {⟨*content*⟩} 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*.

```
\cs_new_protected:Nn \__enumext_store_addto_seq:n
  {
    \tl_if_empty:NF \l__enumext_store_name_tl
      {
        \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } {#1}
      }
  }
\cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

(*End of definition for* `\__enumext_store_addto_seq:n`.)

13.32.8 Functions for storing structure in the sequence

`\__enumext_store_level_open:`
`\__enumext_store_level_close:`

Within the `enumext` environment, the *"storing structure"* is handled per level by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:`.

```
\cs_new_protected:Nn \__enumext_store_level_open:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
```

©2024–2026 by Pablo González L

```
      \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
        {
          \__enumext_store_addto_seq:n
            {
              \item \begin{enumext}
            }
        }
        {
          \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
            {
              \item \begin{enumext} [
            }
          \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
            {
              ]
            }
          \__enumext_store_addto_seq:v
            { l__enumext_store_save_key_ \__enumext_level: _tl }
        }
    }
  }
\cs_new_protected:Nn \__enumext_store_level_close:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \__enumext_store_addto_seq:n { \end{enumext} }
      }
  }
```

(*End of definition for* `\__enumext_store_level_open:` *and* `\__enumext_store_level_close:`.)

`\__enumext_store_level_open_vii:`
`\__enumext_store_level_close_vii:`

The functions `\__enumext_store_level_open_vii:` and `\__enumext_store_level_close_vii:` handle the *"storing structure"* and are executed in the `enumext*` environment.

```
\cs_new_protected:Nn \__enumext_store_level_open_vii:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
          {
            \__enumext_store_addto_seq:n
              {
                \item \begin{enumext*}
              }
          }
          {
            \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
              {
                \item \begin{enumext*}[
              }
            \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
              {
                ]
              }
            \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
          }
      }
  }
\cs_new_protected:Nn \__enumext_store_level_close_vii:
  {
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \__enumext_store_addto_seq:n { \end{enumext*} }
      }
  }
```

(*End of definition for* `\__enumext_store_level_open_vii:` *and* `\__enumext_store_level_close_vii:`.)

13.32.9 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 the content of `\l__enumext_mark_answer_sym_tl`, 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

\cs_new_protected:Nn \__enumext_print_keyans_box:NN
  {
    \mode_leave_vertical:
    \skip_horizontal:n { -\dim_use:N #2 }
    \hbox_overlap_left:n
      {
        \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
          {
            \tl_use:N \l__enumext_mark_answer_sym_tl
          }
      }
    \skip_horizontal:n { \dim_use:N #2 }
  }
\cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

(*End of definition for* `\__enumext_print_keyans_box:NN`.)

13.33 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 ⟨*labels*⟩, since double dots in references are not wanted, then the result is placed in the variable `\l__enumext_newlabel_arg_two_tl`.

```
\cs_new_protected:Nn \__enumext_store_internal_ref:
  {
    \cs_set_protected:Nn \__enumext_tmp:n
      {
        \tl_set_eq:cc { l__enumext_label_copy_ ##1 _tl } { l__enumext_label_ ##1 _tl }
        \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
        \tl_remove_once:cn { l__enumext_label_copy_ ##1 _tl } { . }
        \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
      }
    \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
    \cs_set:Nn \__enumext_tmp:n
      { . \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.

```
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \tl_use:N \l__enumext_label_copy_vii_tl }
      }
    \bool_lazy_all:nT
      {
        { \bool_not_p:n { \g__enumext_standard_bool } }
        { \bool_if_p:N \l__enumext_standard_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \tl_use:N \l__enumext_label_copy_vii_tl
            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
```

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.

```
    \bool_lazy_all:nT
```

```
      {
        { \bool_if_p:N \g__enumext_standard_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \tl_use:N \l__enumext_label_copy_i_tl
            \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
          }
      }
    \cs_set:Nn \__enumext_tmp:n
      { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \g__enumext_standard_bool }
        { \bool_if_p:N \l__enumext_starred_bool }
        { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
      }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          {
            \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
            \tl_use:N \l__enumext_label_copy_vii_tl
          }
      }
```

Now the variable `\l__enumext_newlabel_arg_one_tl` is set, which contains {⟨*store name* : *position*⟩}.

```
    \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
      {
        \l__enumext_store_name_tl \c_colon_str
        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
      }
```

Now the function `\__enumext_newlabel:nn` is executed and the result is saved in the variable `\l__enumext_write_aux_file_tl`, which is finally written in the `.aux` file.

```
    \tl_put_right:Ne \l__enumext_write_aux_file_tl
      {
        \__enumext_newlabel:nn
          { \exp_not:V \l__enumext_newlabel_arg_one_tl }
          { \l__enumext_newlabel_arg_two_tl }
      }
    \l__enumext_write_aux_file_tl
  }
```

(*End of definition for* `\__enumext_store_internal_ref:`.)

13.34 Common functions for \anskey and anskey* environment

13.34.1 Definition of variables

`\l__enumext_store_anskey_arg_tl`

The variable `\l__enumext_store_anskey_arg_tl` saves the *argument* of `\anskey` (§13.35).

```
\tl_new:N \l__enumext_store_anskey_arg_tl
```

(*End of definition for* `\l__enumext_store_anskey_arg_tl`.)

13.34.2 Definition of error messages

`anskey-unnumber-item`
`anskey-empty-arg`
`anskey-math-mode`

Messages used by `\anskey` and `anskey*`.

```
\msg_new:nnnn { enumext } { anskey-unnumber-item }
  {
    Can't~store~with~an~unnumbered~'\token_to_str:N \item'~\msg_line_context:.
  }
  {
    Store~the~content~with~a~numbered~'\token_to_str:N \item'.
  }
\msg_new:nnnn { enumext } { anskey-empty-arg }
  {
    Can't~store~empty~content~\msg_line_context:.
```

```
  }
  {
    Give~some~content~to~'\token_to_str:N \anskey'~or~remove~the~command.
  }
\msg_new:nnnn { enumext } { anskey-math-mode }
  {
    '#1'~can't~work~in~math~mode~\msg_line_context:.
  }
  {
    Leave~math~mode~before~using~'#1'.
  }
```

(*End of definition for* `anskey-unnumber-item`, `anskey-empty-arg`, *and* `anskey-math-mode`.)

13.34.3 Internal functions common to \anskey and anskey*

`\__enumext_store_anskey_arg:n`
`\__enumext_store_anskey_arg:e`

The internal function `\__enumext_store_anskey_arg:n` first passes the {⟨*argument*⟩} 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"* {⟨*argument*⟩}.

```
\cs_new_protected:Nn \__enumext_store_anskey_arg:n
  {
    \int_gincr:N \g__enumext_item_anskey_int
    \__enumext_store_addto_prop:n {#1}
    \bool_if:NT \l__enumext_store_ref_key_bool
      {
        \__enumext_store_internal_ref:
      }
    \__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.

```
    \tl_clear:N \l__enumext_store_anskey_arg_tl
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_store_columns_break_bool }
      { \bool_not_p:n { \l__enumext_starred_bool } }
      {
        \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
      }
    \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`.

```
    \bool_lazy_and:nnT
      { \bool_not_p:n { \l__enumext_starred_bool } }
      { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
      {
        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
          {
            ( \exp_not:V \l__enumext_store_item_join_int )
          }
      }
```

Then the keys `item-star`, `item-sym*` and `item-pos*` are reviewed and passed, along with the {⟨*argument*⟩} for `\anskey` or ⟨*body*⟩ for `anskey*`, to `\l__enumext_store_anskey_arg_tl`.

```
    \bool_if:NTF \l__enumext_store_item_star_bool
      {
        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
        \tl_if_empty:NF \l__enumext_store_item_symbol_tl
          {
            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
              {
                [ \exp_not:V \l__enumext_store_item_symbol_tl ]
              }
          }
        \dim_compare:nT
          {
```

```
3132              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
3133            }
3134            {
3135              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3136                {
3137                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
3138                }
3139            }
3140          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
3141        }
3142        {
3143          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
3144        }
```

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*.

```
3145      \bool_lazy_and:nnT
3146        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3147        { \bool_if_p:N \l__enumext_hyperref_bool }
3148        {
3149          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
3150            {
3151              \hfill \exp_not:N \hyperlink
3152                { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3153                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3154            }
3155        }
3156      \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
3157    }
3158  \cs_generate_variant:Nn \__enumext_store_anskey_arg:n { e }
```

(*End of definition for* __enumext_store_anskey_arg:n.)

__enumext_anskey_show_wrap_arg:n

The function __enumext_anskey_show_wrap_arg:n *"wraps"* the {⟨*argument*⟩} passed to \anskey and the ⟨*body*⟩ for anskey* when using the wrap-ans and mark-sep keys.

```
3159  \cs_new_protected:Nn \__enumext_anskey_show_wrap_arg:n
3160    {
3161      \par
3162      \bool_if:NTF \l__enumext_starred_bool
3163        {
3164          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3165            {
3166              \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
3167            }
3168          \__enumext_print_keyans_box:NN
3169            \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
3170        }
3171        {
3172          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3173            {
3174              \dim_set:Nn \l__enumext_mark_sym_sep_dim
3175                {
3176                  \dim_use:c {l__enumext_labelsep_ \__enumext_level: _dim }
3177                }
3178            }
3179          \__enumext_print_keyans_box:cc
3180            { l__enumext_labelwidth_ \__enumext_level: _dim }
3181            { l__enumext_mark_sym_sep_dim }
3182        }
3183      \__enumext_anskey_wrapper:n {#1}
3184    }
```

(*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 {⟨*content*⟩} stored in the *prop list* when using the show-pos key on the left margin next to the *"wraps"* {⟨*argument*⟩} passed to \anskey and the ⟨*body*⟩ in anskey* on the right side when using the show-ans key.

```
3185  \cs_new_protected:Nn \__enumext_anskey_show_wrap_left:n
3186    {
```

```
    \bool_if:NT \l__enumext_show_answer_bool
      {
        \__enumext_anskey_show_wrap_arg:n {#1}
      }
    \bool_if:NT \l__enumext_show_position_bool
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
            \exp_not:N \normalfont
            \exp_not:N \footnotesize [ \int_eval:n
              {
                \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
              }
              ]
            \group_end:
          }
        \__enumext_anskey_show_wrap_arg:n {#1}
      }
  }
```

(*End of definition for* `\__enumext_anskey_show_wrap_left:n`.)

13.35 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*⟩`}`.

13.35.1 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.

```
\int_new:N \l__enumext_anskey_level_int
```

(*End of definition for* `\l__enumext_anskey_level_int`.)

13.35.2 Definition of error messages for \anskey

`anskey-wrong-place`
`anskey-nested`
`anskey-cmd-key-unknown`
`anskey-cmd-key-value-unknown`

Messages used by `\anskey` command.

```
\msg_new:nnnn { enumext } { anskey-wrong-place }
  {
    Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.
  }
  {
    '\c_backslash_str #1'~works~in~the~environment~'#2'.
  }
\msg_new:nnnn { enumext } { anskey-nested }
  {
    The~command~'\token_to_str:N \anskey'~can't~be~nested~
    \msg_line_context:.
  }
  {
    Close~the~first~'\token_to_str:N \anskey'~before~using~another~one.
  }
\msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
  {
    The~key~'#1'~is~unknown~by~'\token_to_str:N \anskey'~and~is~being~
    ignored~\msg_line_context:.
  }
  {
    The~command~'\token_to_str:N \anskey'~does~not~have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
  {
    The~key~'#1=#2'~is~unknown~by~'\token_to_str:N \anskey'~and~is~being~
    ignored~\msg_line_context:.
  }
  {
    The~command~'\token_to_str:N \anskey'~does~not~have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

(*End of definition for* `anskey-wrong-place` *and others.*)

13.35.3 Definition of keys for \anskey

break-col
item-join
item-star
item-sym*
item-pos*
unknown

First, the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*` are added.

```
\keys_define:nn { enumext / anskey }
  {
    break-col .bool_set:N = \l__enumext_store_columns_break_bool,
    break-col .default:n = true,
    break-col .value_forbidden:n = true,
    item-join .int_set:N = \l__enumext_store_item_join_int,
    item-join .value_required:n = true,
    item-star .bool_set:N = \l__enumext_store_item_star_bool,
    item-star .default:n = true,
    item-star .value_forbidden:n = true,
    item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
    item-sym* .value_required:n = true,
    item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
    item-pos* .value_required:n = true,
    unknown   .code:n = { \__enumext_anskey_unknown:n {#1} },
  }
```

(*End of definition for* `break-col` *and others.*)

13.35.4 Internal functions for the command

__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn
__enumext_anskey_unknown:Vn

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`.

```
\cs_new_protected:Nn \__enumext_anskey_unknown:n
  {
    \__enumext_anskey_unknown:Vn \l_keys_key_str {#1}
  }
\cs_new_protected:Nn \__enumext_anskey_unknown:nn
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
      }
      {
        \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__enumext_anskey_unknown:nn { V }
```

(*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.

```
\prg_new_conditional:Npnn \__enumext_in_keyans: { p , T , F , TF }
  {
    \int_compare:nNnTF
      { \l__enumext_keyans_level_int + \l__enumext_keyans_level_h_int
        + \l__enumext_keyans_pic_level_int } > { 0 }
      { \prg_return_true: } { \prg_return_false: }
  }
```

(*End of definition for* `\__enumext_in_keyans:`.)

__enumext_anskey_safe_outer:
__enumext_anskey_safe_inner:

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.

```
\cs_new_protected:Nn \__enumext_anskey_safe_outer:
  {
    \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
      {
        \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
```

```
3287          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
3288        }
3289      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3290        {
3291          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
3292        }
3293      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3294        {
3295          \msg_error:nnnn { enumext }
3296            { command-wrong-place }{ anskey }{ keyanspic }
3297        }
3298    }
```

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.

```
3299 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3300   {
3301     \int_incr:N \l__enumext_anskey_level_int
3302     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
3303       {
3304         \msg_error:nn { enumext } { anskey-nested }
3305       }
3306     \bool_if:NF \l__enumext_item_number_bool
3307       {
3308         \msg_error:nn { enumext } { anskey-unnumber-item }
3309       }
3310     \mode_if_math:T
3311       {
3312         \msg_error:nne { enumext } { anskey-math-mode }
3313           { \token_to_str:N \anskey }
3314       }
3315   }
```

(*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.

```
3316 \NewDocumentCommand \anskey { o +m }
3317   {
3318     \__enumext_anskey_safe_outer:
3319     \group_begin:
3320       \bool_if:NT \l__enumext_check_answers_bool
3321         {
3322           \tl_if_novalue:nF {#1}
3323             {
3324               \keys_set:nn { enumext / anskey } {#1}
3325             }
3326           \tl_if_blank:nTF {#2}
3327             {
3328               \msg_error:nn { enumext } { anskey-empty-arg }
3329             }
3330             {
3331               \__enumext_anskey_safe_inner:
3332               \__enumext_store_anskey_arg:n {#2}
3333             }
3334         }
3335     \group_end:
3336   }
```

(*End of definition for* `\anskey`. *This function is documented on page* 16.)

13.36 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 LaTeX release 2025-06-01 implemented the new c-type argument in the ltcmd[11], with which the ⟨*body*⟩ of the environment can be recorded in *verbatim mode*, and \tl_retokenize:n (wrapper around the \scantokens) provided by LaTeX3 release 2025-07-08 does the work as the original implementation.

13.36.1 Definition of variables for anskey*

\l__enumext_store_anskey_env_tl
\l__enumext_write_anskey_env_bool
\l__enumext_write_anskey_env_file_name_tl
\l__enumext_write_anskey_env_file_iow

The variable \l__enumext_store_anskey_env_tl saves the ⟨*body*⟩ of the environment anskey* (§13.36). In the implementation of the anskey* environment, the keys write-env and overwrite use the variables \l__enumext_write_anskey_env_bool, \l__enumext_write_anskey_env_file_name_tl and \l__enumext_write_anskey_env_file_iow.

```
\tl_new:N   \l__enumext_store_anskey_env_tl
\bool_new:N \l__enumext_write_anskey_env_bool
\tl_new:N   \l__enumext_write_anskey_env_file_name_tl
\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 TeX 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.

```
\str_const:Ne \c__enumext_anskey_env_hidden_space_str
  { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol
    \c_percent_str }
```

(*End of definition for* \c__enumext_anskey_env_hidden_space_str.)

13.36.2 Definition of error messages for anskey*

anskey-env-error

Message used by reserved anskey* environment by enumext package.

```
\msg_new:nnnn { enumext } { anskey-env-error }
  {
    The~environment~'#1'~is~reserved~by~'enumext'~package,~
    it~is~already~defined~\msg_line_context:.
  }
  {
    The~environment~'#1'~is~defined~internally~
    for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.
  }
```

(*End of definition for* anskey-env-error.)

anskey-env-wrong
anskey-env-key-unknown
anskey-env-key-value-unknown

Messages used by anskey* environment.

```
\msg_new:nnnn { enumext } { anskey-env-wrong }
  {
    The~environment~anskey*~cannot~be~used~in~'#1'~\msg_line_context:.
  }
  {
    'anskey*'~can~only~be~used~with~the~key~'save-ans'~in~
    'enumext'~or~'enumext*'.
  }
\msg_new:nnnn { enumext } { anskey-env-key-unknown }
  {
    The~key~'#1'~is~unknown~by~environment~
    'anskey*'~and~is~being~ignored~\msg_line_context:.
  }
  {
    The~environment~'anskey*'~does~not~have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
  {
    The~key~'#1=#2'~is~unknown~by~environment~
    'anskey*'~and~is~being~ignored~\msg_line_context:.
  }
  {
    The~environment~'anskey*'~does~not~have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

(*End of definition for* anskey-env-wrong, anskey-env-key-unknown, *and* anskey-env-key-value-unknown.)

©2024–2026 by Pablo González L

13.36.3 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
writing-file
not-writing

Messages used when anskey* writes the external file.

```
\msg_new:nnn { enumext } { overwrite-file }
  {
    Overwriting~file~'#1'.
  }
\msg_new:nnn { enumext } { writing-file }
  {
    Writing~file~'#1'.
  }
\msg_new:nnn { enumext } { not-writing }
  {
    File~'#1'~already~exists.~Not~writing.
  }
```

(*End of definition for* `overwrite-file`, `writing-file`, *and* `not-writing`.)

13.36.4 Definition of keys for anskey*

break-col
item-join
item-star
item-sym*
item-pos*
force-eol
write-env
overwrite
unknown

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.

```
\keys_define:nn { enumext / anskey* }
  {
    break-col .bool_set:N = \l__enumext_store_columns_break_bool,
    break-col .default:n = true,
    break-col .value_forbidden:n = true,
    item-join .int_set:N = \l__enumext_store_item_join_int,
    item-join .value_required:n = true,
    item-star .bool_set:N = \l__enumext_store_item_star_bool,
    item-star .default:n = true,
    item-star .value_forbidden:n = true,
    item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
    item-sym* .value_required:n = true,
    item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
    item-pos* .value_required:n = true,
    force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
    force-eol .initial:n = false,
    force-eol .default:n = true,
    write-env .code:n =
      {
        \bool_set_true:N \l__enumext_write_anskey_env_bool
        \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
      },
    write-env .value_required:n = true,
    overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
    overwrite .initial:n = false,
    overwrite .default:n = true,
    unknown   .code:n = { \__enumext_anskey_env_unknown:n {#1} },
  }
```

(*End of definition for* `break-col` *and others.*)

13.36.5 Internal functions for the environment

__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn
__enumext_anskey_env_unknown:Vn

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.

```
\cs_new_protected:Nn \__enumext_anskey_env_unknown:n
  {
    \__enumext_anskey_env_unknown:Vn \l_keys_key_str {#1}
  }
\cs_new_protected:Nn \__enumext_anskey_env_unknown:nn
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
      }
      {
```

```
3430          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3431        }
3432    }
3433  \cs_generate_variant:Nn \__enumext_anskey_env_unknown:nn { V }
```

(*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.

```
3434  \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1
3435    { T, F, TF }
3436    {
3437      \bool_if:NTF \l__enumext_write_anskey_env_bool
3438        {
3439          \file_if_exist:nTF {#1}
3440            {
3441              \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
3442                {
3443                  \msg_warning:nne { enumext } { overwrite-file } {#1}
3444                  \prg_return_true:
3445                }
3446                {
3447                  \msg_warning:nne { enumext } { not-writing } {#1}
3448                  \prg_return_false:
3449                }
3450            }
3451            {
3452              \msg_warning:nne { enumext } { writing-file } {#1}
3453              \prg_return_true:
3454            }
3455        }
3456        { \prg_return_false: }
3457    }
```

The `\__enumext_anskey_env_file_write:nn` function is used by the `write-env` key in the `anskey*` environment to write the output file with the ⟨*body*⟩ of the environment.

```
3458  \cs_new_protected:Nn \__enumext_anskey_env_file_write:nn
3459    {
3460      \__enumext_anskey_env_file_if_writable:nT {#1}
3461        {
3462          \iow_open:Nn \l__enumext_write_anskey_env_file_iow {#1}
3463          \iow_now:Nn  \l__enumext_write_anskey_env_file_iow {#2}
3464          \iow_close:N \l__enumext_write_anskey_env_file_iow
3465        }
3466    }
3467  \cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }
```

(*End of definition for* `\__enumext_anskey_env_file_if_writable:n` *and others.*)

`\__enumext_anskey_env_safe_outer:`
`\__enumext_anskey_env_safe_inner:`
`\__enumext_store_anskey_env:n`

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.

```
3468  \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3469    {
3470      \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
3471        {
3472          \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3473        }
3474      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3475        {
3476          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
3477        }
3478      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3479        {
3480          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3481        }
3482      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3483        {
3484          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3485        }
3486    }
```

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.

```
\cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
  {
    \bool_if:NF \l__enumext_item_number_bool
      {
        \msg_error:nn { enumext } { anskey-unnumber-item }
      }
    \mode_if_math:T
      {
        \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
      }
  }
```

The __enumext_store_anskey_env:n function will first pass the c-type argument ⟨*body*⟩ to \l__enumext_store_anskey_env_tl and replace the macro \obeyedline with ^^J. Then it executes the write-env and overwrite keys, checks the state of \l__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 calls __enumext_store_anskey_arg:e to expand the function \tl_retokenize:n, which rescans \l__enumext_store_anskey_env_tl before processing it.

```
\cs_new_protected:Nn \__enumext_store_anskey_env:n
  {
    \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
    \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^J }
    \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline }
      { \iow_char:N \^^J }
    \__enumext_anskey_env_file_write:VV
      \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
    \bool_if:NF \l__enumext_anskey_env_force_eol_bool
      {
        \tl_put_right:Ne \l__enumext_store_anskey_env_tl
          {
            \c__enumext_anskey_env_hidden_space_str
          }
      }
    \__enumext_store_anskey_arg:e
      {
        \tl_retokenize:n { \l__enumext_store_anskey_env_tl }
      }
  }
```

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.

```
\NewDocumentEnvironment{anskey*}{ o c }
  {
    \__enumext_anskey_env_safe_outer:
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \tl_if_novalue:nF {#1}
          {
            \keys_set:nn { enumext / anskey* } {#1}
          }
        \tl_if_blank:nTF {#2}
          {
            \msg_error:nn { enumext } { anskey-empty-arg }
          }
          {
            \__enumext_anskey_env_safe_inner:
            \__enumext_store_anskey_env:n {#2}
          }
```

```
      }
  } { }
```

(*End of definition for* `anskey*`. *This function is documented on page* 17.)

13.37 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.

```
\cs_new_protected:Nn \__enumext_execute_after_env:
  {
    \int_compare:nNnT { \l__enumext_level_int } = { 0 }
      {
        \tl_if_empty:NF \g__enumext_store_name_tl
          {
            \__enumext_stop_save_ans_msg:
            \__enumext_item_answer_diff:
            \__enumext_log_global_vars:
            \__enumext_log_answer_vars:
            \bool_if:NTF \g__enumext_check_ans_key_bool
              {
                \__enumext_check_ans_show:
              }
              { \__enumext_check_ans_log: }
          }
        \__enumext_reset_global_vars:
      }
  }
```

This function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` (§13.44) and `enumext*` (§13.49) 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:`.)

13.38 Common functions for `keyans`, `keyans*` and `keyanspic`

13.38.1 Definition of variables

`\l__enumext_mark_sep_tmpa_dim`
`\l__enumext_mark_sep_tmpb_dim`
`\l__enumext_show_pos_tmp_int`

Internal variables used by `mark-sep`, `show-ans` and `show-pos` keys (§13.32).

```
\dim_new:N \l__enumext_mark_sep_tmpa_dim
\dim_new:N \l__enumext_mark_sep_tmpb_dim
\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`.)

13.38.2 Definition of error messages

`keyans-wrong-level`
`wrong-place`

Messages used by `keyans`, `keyans*` and `keyanspic` environments.

```
\msg_new:nnnn { enumext } { keyans-wrong-level }
  {
    Wrong~level~position~for~'keyans'~\msg_line_context:.
  }
  {
    The~environment~'keyans'~can~only~be~in~the~first~level.
  }
\msg_new:nnnn { enumext } { wrong-place }
  {
    Wrong~place~for~'#1'~environment~\msg_line_context:.
  }
  {
    '#1'~is~only~found~with~'#2'~in~'enumext'.
  }
```

(*End of definition for* `keyans-wrong-level` *and* `wrong-place`.)

13.38.3 Storing content in prop list

__enumext_keyans_addto_prop:n

The function __enumext_keyans_addto_prop:n will pass the current ⟨*label*⟩ for \item* in keyans environment and the current ⟨*label*⟩ for \anspic* in keyanspic environment followed by the ⟨*contents*⟩ 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.

```
\cs_new_protected:Nn \__enumext_keyans_addto_prop:n
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
      }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
      }
```

If the *optional argument* is present and the save-sep key is not empty, it is saved.

```
    \tl_if_novalue:nF {#1}
      {
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
          {
            \tl_put_right:NV \l__enumext_store_current_label_tl
              \l__enumext_store_keyans_item_opt_sep_v_tl
          }
        \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
      }
    \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
  }
```

(*End of definition for* __enumext_keyans_addto_prop:n.)

13.38.4 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.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref:
  {
    \bool_if:NT \l__enumext_store_ref_key_bool
      {
        \cs_set_protected:Nn \__enumext_tmp:n
          {
            \tl_set_eq:cc { l__enumext_label_copy_ ##1 _tl }
              { l__enumext_label_ ##1 _tl }
            \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
            \tl_remove_once:cn { l__enumext_label_copy_ ##1 _tl } { . }
            \tl_reverse:c { l__enumext_label_copy_ ##1 _tl }
          }
        \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
        \__enumext_keyans_store_ref_aux_i:
      }
  }
```

The auxiliary function __enumext_keyans_store_ref_aux_i: analyzes whether the environment in which it is executed is enumext* or enumext and sets the variable \l__enumext_newlabel_arg_one_tl, which will contain {⟨*store name : position*⟩}.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
  {
    \bool_if:NT \g__enumext_starred_bool
      {
        \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
      }
```

```
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
          { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
      }
    \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
      {
        \l__enumext_store_name_tl \c_colon_str
        \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
      }
    \__enumext_keyans_store_ref_aux_ii:
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
  {
    \tl_put_right:Ne \l__enumext_write_aux_file_tl
      {
        \__enumext_newlabel:nn
          { \exp_not:V \l__enumext_newlabel_arg_one_tl }
          { \l__enumext_newlabel_arg_two_tl }
      }
    \l__enumext_write_aux_file_tl
  }
```

(*End of definition for* `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, *and* `\__enumext_keyans_store_ref_aux_ii:`.)

13.38.5 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*⟩ to the variable `\l__enumext_store_current_label_tl`. This is `\l__enumext_label_v_tl` for the `keyans` environment and `\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, and then to the sequence defined by the `save-ans` key.

```
\cs_new_protected:Nn \__enumext_keyans_addto_seq:n
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl
          { \item \l__enumext_label_vi_tl }
      }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl
          { \item \l__enumext_label_v_tl }
      }
    \tl_if_novalue:nF {#1}
      {
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
          {
            \tl_put_right:NV \l__enumext_store_current_label_tl
              \l__enumext_store_keyans_item_opt_sep_v_tl
          }
        \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
      }
    \__enumext_keyans_addto_seq_link:
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
  {
    \bool_lazy_and:nnT
      { \bool_if_p:N \l__enumext_store_ref_key_bool }
      { \bool_if_p:N \l__enumext_hyperref_bool }
      {
        \tl_put_right:Ne \l__enumext_store_current_label_tl
          {
            \hfill \exp_not:N \hyperlink
              {
                \exp_not:V \l__enumext_newlabel_arg_one_tl
              }
              { \exp_not:V \l__enumext_mark_ref_sym_tl }
          }
      }
    \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \int_gincr:N \g__enumext_item_anskey_int
      }
  }
```

(*End of definition for* __enumext_keyans_addto_seq:n *and* __enumext_keyans_addto_seq_link:.)

13.38.6 The show-ans and show-pos keys for keyans and keyanspic

__enumext_keyans_save_item_opt:n
__enumext_keyans_show_item_opt:
__enumext_keyans_show_item_opt_viii:

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.

```
\cs_new_protected:Nn \__enumext_keyans_save_item_opt:n
  {
    \tl_if_novalue:nF {#1}
      {
        \tl_set:Nn \l__enumext_store_current_opt_arg_tl {#1}
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_show_item_opt:
  {
    \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
      {
        \bool_lazy_or:nnT
          { \bool_if_p:N \l__enumext_show_answer_bool }
          { \bool_if_p:N \l__enumext_show_position_bool }
          {
            \__enumext_keyans_wrapper_opt_v:n
              { \l__enumext_store_current_opt_arg_tl } \c_space_tl
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
  {
    \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
      {
        \bool_lazy_or:nnT
          { \bool_if_p:N \l__enumext_show_answer_bool }
          { \bool_if_p:N \l__enumext_show_position_bool }
          {
            \__enumext_keyans_wrapper_opt_viii:n
              { \l__enumext_store_current_opt_arg_tl } \c_space_tl
```

```
            }
        }
    }
```

(*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*⟩.

```
\cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
  {
    \__enumext_label_width_by_box:Nn
      \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
    \str_case:Vn \l__enumext_align_label_pos_v_str
      {
        { l }
          {
            \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
          }
        { r }
          {
            \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
              { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
          }
        { c }
          {
            \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
              { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
          }
      }
```

Sets the default values for the keys `mark-ans*`, `mark-sep*` and `mark-pos*`.

```
    \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
      }
    \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
    \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
    \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
  }
```

The function `\__enumext_keyans_show_ans:` will print the ⟨*symbol*⟩ set by the `mark-ans*` key when the `show-ans` key is active.

```
\cs_new_protected:Nn \__enumext_keyans_show_ans:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_show_answer_bool   }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool }
      }
      {
        \__enumext_keyans_pos_mark_set:
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_show_pos:
  {
    \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
      {
        \int_incr:N \l__enumext_show_pos_tmp_int
      }
      {
        \int_zero:N \l__enumext_show_pos_tmp_int
      }
```

```
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_show_position_bool }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool }
      }
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_v_tl
          {
            \group_begin:
              \exp_not:N \normalfont
              \exp_not:N \footnotesize [ \int_eval:n
                {
                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
                  + \l__enumext_show_pos_tmp_int
                }
              ]
            \group_end:
          }
        \__enumext_keyans_pos_mark_set:
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
      }
  }
```

(*End of definition for* `\__enumext_keyans_pos_mark_set:`, `\__enumext_keyans_show_ans:`, *and* `\__enumext_keyans_show_pos:`.)

13.39 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 (§13.40).

```
\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`, and then sets the boolean variable `\l__enumext_wrap_label_X_bool` to `true` for the key `wrap-label`. After that it 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`.

```
\cs_new_protected:Nn \__enumext_default_item:n
  {
    \tl_if_novalue:nTF {#1}
      {
        \bool_if:NT \l__enumext_check_answers_bool
          {
            \int_gincr:N \g__enumext_item_number_int
            \bool_set_true:N \l__enumext_item_number_bool
          }
        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
        \__enumext_item_std:w
          \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
      }
      {
        \bool_set_eq:cc
          { l__enumext_wrap_label_ \__enumext_level: _bool }
          { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
        \__enumext_item_std:w [#1]
          \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
      }
  }
```

©2024–2026 by Pablo González L

(*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.

```
\cs_new_protected:Nn \__enumext_item_starred_exec:nn
  {
    \tl_if_novalue:nTF {#1}
      {
        \tl_gset_eq:Nc
          \g__enumext_item_symbol_aux_tl
            { l__enumext_item_symbol_ \__enumext_level: _tl }
      }
      {
        \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
      }
    \tl_if_novalue:nTF {#2}
      {
        \dim_set_eq:cc
          { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
          { l__enumext_labelsep_ \__enumext_level: _dim }
      }
      {
        \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
      }
    \bool_if:NT \l__enumext_check_answers_bool
      {
        \int_gincr:N \g__enumext_item_number_int
        \bool_set_true:N \l__enumext_item_number_bool
      }
    \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
    \__enumext_item_std:w
      \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
  }
```

The function `\__enumext_item_starred_exec:` will be responsible for executing `\item*` for the `enumext` environment.

```
\cs_new_protected:Nn \__enumext_item_starred_exec:
  {
    \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
      {
        \mode_leave_vertical:
        \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_
          \__enumext_level: _dim } }
        \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
        \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_
          \__enumext_level: _dim } }
      }
  }
```

(*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 (§13.44).

```
\cs_new_protected:Nn \__enumext_redefine_item:
  {
    \RenewDocumentCommand \item { s o o }
      {
        \bool_if:nTF {##1}
          {
```

```
3868              \__enumext_item_starred_exec:nn {##2} {##3}
3869            }
3870            { \__enumext_default_item:n {##2} }
3871        }
3872    }
```

(*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 the `enumext` environment. This function is passed to `\__enumext_list_arg_two_X:`, used in the definition of the `enumext` environment (§13.44).

```
3873 \cs_new_protected:Nn \__enumext_make_label:
3874   {
3875     \IfDocumentMetadataTF
3876       {
3877         \__enumext_make_label_box:
3878       }
3879       {
3880         \bool_if:NTF \l__enumext_mode_box_bool
3881           {
3882             \__enumext_make_label_box:
3883           }
3884           {
3885             \__enumext_make_label_std:
3886           }
3887       }
3888   }
```

Standard definition when `\DocumentMetadata` is not active.

```
3889 \cs_new_protected:Nn \__enumext_make_label_std:
3890   {
3891     \RenewDocumentCommand \makelabel { m }
3892       {
3893         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3894         \__enumext_item_starred_exec:
3895         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3896         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3897           {
3898             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } {##1}
3899           }
3900           {##1}
3901         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3902         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3903       }
3904   }
```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

The label is not smashed, so a tall label (an image, a duck drawn with `tikz`, ...) still increases the height of the line as it does with the standard `\makelabel`. A multi-line label made with `\parbox[t]` needs `\leavevmode` at the start of the paragraph (see `\labelbx` in §9), otherwise the box starts in *vertical mode* and the baseline is lost, with or without *tagged* PDF.

```
3905 \cs_new_protected:Nn \__enumext_make_label_content:n
3906   {
3907     \__enumext_item_starred_exec:
3908     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3909     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3910       { \use:c { __enumext_wrapper_label_ \__enumext_level: :n } {#1} }
3911       {#1}
3912   }
```

The label is measured first (with *tagged* PDF suspended) so that the box is never narrower than the label itself, as in the standard `\makelabel`.

```
3913 \cs_new_protected:Nn \__enumext_make_label_box:
3914   {
3915     \RenewDocumentCommand \makelabel { m }
3916       {
3917         \hbox_set:Nn \l__enumext_label_width_by_box
3918           {
```

©2024–2026 by Pablo González L

```
3919              \IfDocumentMetadataT { \SuspendTagging { enumext } }
3920              \__enumext_make_label_content:n {##1}
3921            }
3922          \makebox
3923            [ \dim_max:nn
3924                { \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } }
3925                { \box_wd:N \l__enumext_label_width_by_box } ]
3926            [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3927            { \__enumext_make_label_content:n {##1} }
3928          \tl_gclear:N \g__enumext_item_symbol_aux_tl
3929        }
3930    }
```

(*End of definition for* `\__enumext_make_label:` , `\__enumext_make_label_std:` , *and* `\__enumext_make_label_box:`.)

13.40 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*` `item-pos*`

Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```
3931 \cs_set_protected:Nn \__enumext_tmp:nn
3932   {
3933     \keys_define:nn { enumext / #1 }
3934       {
3935         item-sym* .tl_set:c = { l__enumext_item_symbol_ #2 _tl },
3936         item-sym* .value_required:n = true,
3937         item-sym* .initial:n = {\textborn},
3938         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_ #2 _dim },
3939         item-pos* .value_required:n = true,
3940       }
3941   }
3942 \clist_map_inline:nn
3943   {
3944     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3945   }
3946   { \__enumext_tmp:nn #1 }
```

(*End of definition for* `item-sym*` *and* `item-pos*`.)

13.41 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.

13.41.1 Definition of error messages

`starred-unknown-key` `starred-unknown-key-value`

Message used by unknown ⟨*keys*⟩ in `enumext*` environment.

```
3947 \msg_new:nnnn { enumext } { starred-unknown-key }
3948   {
3949     The~key~'#1'~is~unknown~by~environment~
3950     '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
3951   }
3952   {
3953     The~environment~'\l__enumext_envir_name_tl'~does~not~
3954     have~a~key~called~'#1'.~
3955     Check~that~you~have~spelled~the~key~name~correctly.
3956   }
3957 \msg_new:nnnn { enumext } { starred-unknown-key-value }
3958   {
3959     The~key~'#1=#2'~is~unknown~by~environment~
3960     '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
3961   }
3962   {
3963     The~environment~'\l__enumext_envir_name_tl'~does~not~
3964     have~a~key~called~'#1'.~
3965     Check~that~you~have~spelled~the~key~name~correctly.
3966   }
```

(*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.

```
\msg_new:nnnn { enumext } { standard-unknown-key }
  {
    The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl'~on~level~
    \int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored~
    \msg_line_context:.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not~
    have~a~key~called~'#1'~on~level~\int_use:N \l__enumext_level_int.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { standard-unknown-key-value }
  {
    The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl'~on~level~
    \int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored~
    \msg_line_context:.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not~
    have~a~key~called~'#1'~on~level~\int_use:N \l__enumext_level_int.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

(*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.

```
\msg_new:nnnn { enumext } { keyans-unknown-key }
  {
    The~key~'#1'~is~unknown~by~environment~
    '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not~
    have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { keyans-unknown-key-value }
  {
    The~key~'#1=#2'~is~unknown~by~environment~
    '\l__enumext_envir_name_tl'~and~is~being~ignored~\msg_line_context:.
  }
  {
    The~environment~'\l__enumext_envir_name_tl'~does~not~
    have~a~key~called~'#1'.~
    Check~that~you~have~spelled~the~key~name~correctly.
  }
```

(*End of definition for* `keyans-unknown-key` *and* `keyans-unknown-key-value`.)

13.41.2 Handling unknown keys for `keyans`, `keyans*` and `keyanspic`

`unknown`
`\__enumext_keyans_unknown_keys:n`
`\__enumext_keyans_unknown_keys:nn`
`\__enumext_keyans_unknown_keys:Vn`

Define and set `unknown` key for `keyans`, `keyans*` and `keyanspic` environments. Here it is necessary to set `\l__enumext_envir_name_tl` in case an `unknown` key is passed using `\setenumext`.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / #1 }
      {
        unknown .code:n =
          {
            \tl_set:Nn \l__enumext_envir_name_tl {#1}
            \__enumext_keyans_unknown_keys:n {##1}
          },
      }
  }
\clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
\cs_new_protected:Nn \__enumext_keyans_unknown_keys:n
  {
    \__enumext_keyans_unknown_keys:Vn \l_keys_key_str {#1}
  }
\cs_new_protected:Nn \__enumext_keyans_unknown_keys:nn
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { keyans-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__enumext_keyans_unknown_keys:nn { V }
```

(*End of definition for* unknown , __enumext_keyans_unknown_keys:n , *and* __enumext_keyans_unknown_keys:nn.)

13.41.3 Handling unknown keys for enumext*

unknown Define and set unknown key for enumext* environment.

```
\keys_define:nn { enumext / enumext* }
  {
    unknown .code:n =
      {
        \tl_set:Ne \l__enumext_envir_name_tl { enumext* }
        \__enumext_starred_unknown_keys:n {#1}
      },
  }
```

(*End of definition for* unknown.)

__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn
__enumext_starred_unknown_keys:Vn

Internal functions for handling unknown key.

```
\cs_new_protected:Nn \__enumext_starred_unknown_keys:n
  {
    \__enumext_starred_unknown_keys:Vn \l_keys_key_str {#1}
  }
\cs_new_protected:Nn \__enumext_starred_unknown_keys:nn
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { starred-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__enumext_starred_unknown_keys:nn { V }
```

(*End of definition for* __enumext_starred_unknown_keys:n *and* __enumext_starred_unknown_keys:nn.)

13.41.4 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
__enumext_standard_unknown_keys:Vn

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \keys_define:nn { enumext / level-#1 }
      {
        unknown .code:n =
          {
            \int_set:Nn \l__enumext_level_int {#1}
            \tl_set:Ne \l__enumext_envir_name_tl { enumext }
            \__enumext_standard_unknown_keys:n {##1}
          },
      }
  }
\clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
\cs_new_protected:Nn \__enumext_standard_unknown_keys:n
  {
    \__enumext_standard_unknown_keys:Vn \l_keys_key_str {#1}
  }
\cs_new_protected:Nn \__enumext_standard_unknown_keys:nn
  {
    \tl_if_blank:nTF {#2}
      {
        \msg_error:nne { enumext } { standard-unknown-key } {#1}
      }
      {
        \msg_error:nnee { enumext } { standard-unknown-key-value } {#1} {#2}
      }
  }
\cs_generate_variant:Nn \__enumext_standard_unknown_keys:nn { V }
```

(*End of definition for* unknown , __enumext_standard_unknown_keys:n , *and* __enumext_standard_unknown_keys:nn.)

13.42 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.

```
\cs_new_protected:Nn \__enumext_keyans_default_item:n
  {
    \tl_if_novalue:nTF {#1}
      {
        \bool_set_true:N \l__enumext_wrap_label_v_bool
        \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
      }
      {
        \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
        \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_keyans_starred_item:n
  {
    \__enumext_keyans_save_item_opt:n {#1}
    \bool_set_true:N \l__enumext_wrap_label_v_bool
    \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
    \__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.

```
    \__enumext_keyans_addto_prop:n {#1}
    \__enumext_keyans_store_ref:
    \__enumext_keyans_addto_seq:n {#1}
    \int_gincr:N \g__enumext_check_starred_cmd_int
  }
```

(*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 (§13.43).

```
\cs_new_protected:Nn \__enumext_keyans_redefine_item:
  {
    \RenewDocumentCommand \item { s o }
      {
        \bool_if:nTF {##1}
          {
            \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
            \peek_remove_spaces:n
              {
                \__enumext_keyans_starred_item:n {##2}
              }
          }
          {
            \bool_set_false:N \l__enumext_item_wrap_key_bool
            \__enumext_keyans_default_item:n {##2}
          }
      }
  }
```

(*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 (§13.43).

```
\cs_new_protected:Nn \__enumext_keyans_make_label:
  {
    \IfDocumentMetadataTF
      {
        \__enumext_keyans_make_label_box:
      }
      {
        \bool_if:NTF \l__enumext_mode_box_bool
          {
            \__enumext_keyans_make_label_box:
          }
          {
            \__enumext_keyans_make_label_std:
          }
      }
  }
```

Conditionals are added to the __enumext_keyans_wrapper_label:n function to handle the keys wrap-ans*, wrap-label and wrap-label*.

```
\cs_new_protected:Nn \__enumext_keyans_wrapper_label:n
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_wrap_label_v_bool         }
        { \bool_if_p:N \l__enumext_show_answer_bool          }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool        }
        { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
      }
      {
        \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
      }
    \bool_if:NTF \l__enumext_wrap_label_v_bool
      {
        \__enumext_wrapper_label_v:n {#1}
      }
      {#1}
  }
```

Standard definition when \DocumentMetadata is not active.

```
\cs_new_protected:Nn \__enumext_keyans_make_label_std:
  {
    \RenewDocumentCommand \makelabel { m }
      {
```

```
        \tl_use:N \l__enumext_label_fill_left_v_tl
        \__enumext_keyans_show_ans:
        \__enumext_keyans_show_pos:
        \tl_use:N \l__enumext_label_font_style_v_tl
        \__enumext_keyans_wrapper_label:n {##1}
        \tl_use:N \l__enumext_label_fill_right_v_tl
      }
  }
```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

```
\cs_new_protected:Nn \__enumext_keyans_make_label_content:n
  {
    \__enumext_keyans_show_ans:
    \__enumext_keyans_show_pos:
    \tl_use:N \l__enumext_label_font_style_v_tl
    \__enumext_keyans_wrapper_label:n {#1}
  }
\cs_new_protected:Nn \__enumext_keyans_make_label_box:
  {
    \RenewDocumentCommand \makelabel { m }
      {
        \hbox_set:Nn \l__enumext_label_width_by_box
          {
            \IfDocumentMetadataT { \SuspendTagging { enumext } }
            \__enumext_keyans_make_label_content:n {##1}
          }
        \makebox
          [ \dim_max:nn
              { \l__enumext_labelwidth_v_dim }
              { \box_wd:N \l__enumext_label_width_by_box } ]
          [ \l__enumext_align_label_pos_v_str ]
          { \__enumext_keyans_make_label_content:n {##1} }
      }
  }
```

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

13.43 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.

13.43.1 Definition of variables

\l__enumext_itemindent_X_dim

The function __enumext_calc_hspace:NNNNNNN sets the dimension \l__enumext_itemindent_X_dim (§13.43.4).

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \dim_new:c { l__enumext_itemindent_ #1 _dim }
  }
\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`.)

13.43.2 Definition of error messages

width-negative

Messages used in length calculation.

```
\msg_new:nnn { enumext } { width-negative }
  {
    Ignoring~negative~value~'#1=#2'~\msg_line_context:.\\
    The~key~'#1'~accepts~values~>=~0pt.
  }
```

(*End of definition for* `width-negative`.)

13.43.3 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.

```
\msg_new:nnn { enumext } { list-lengths }
  {
    ****~Lengths~used~by~'enumext'~level~'#2'~
    \msg_line_context:~\c_space_tl ****\\
    \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
    \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
    \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
    \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
    \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
    \__enumext_show_length:nnn { skip } { topsep        } {#1}
    \__enumext_show_length:nnn { skip } { parsep        } {#1}
    \__enumext_show_length:nnn { skip } { partopsep     } {#1}
    \__enumext_show_length:nnn { skip } { itemsep       } {#1}
    **************************************************
  }
```

(*End of definition for* list-lengths.)

list-lengths-not-nested Messages used by show-length key in enumext*, keyans* and keyans.

```
\msg_new:nnn { enumext } { list-lengths-not-nested }
  {
    ****~Lengths~used~by~'#2'~environment~
    \msg_line_context:~\c_space_tl ****\\
    \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
    \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
    \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
    \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
    \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
    \__enumext_show_length:nnn { skip } { topsep        } {#1}
    \__enumext_show_length:nnn { skip } { parsep        } {#1}
    \__enumext_show_length:nnn { skip } { partopsep     } {#1}
    \__enumext_show_length:nnn { skip } { itemsep       } {#1}
    **************************************************
  }
```

(*End of definition for* list-lengths-not-nested.)

13.43.4 Calculation of \leftmargin and \itemindent

Consider the figure 9 where the default margins (on the left) of a list are represented.

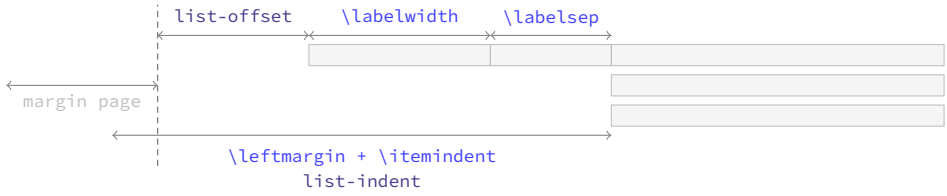


Figure 9: Representation of standard horizontal lengths in list environment.

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.

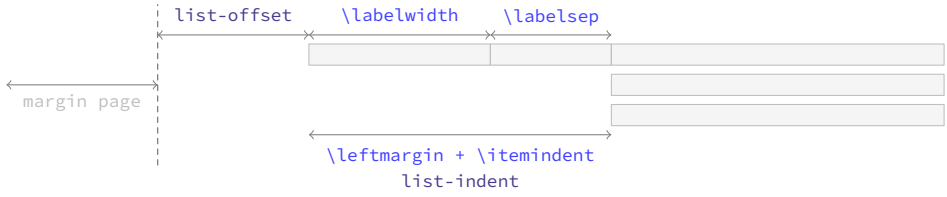


Figure 10: Representation of horizontal lengths concept in list in enumext.

Where the default values will look like in the figure 11.

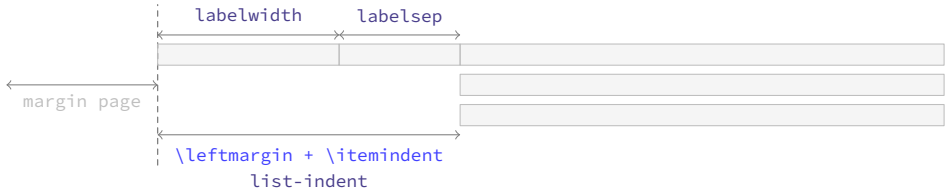


Figure 11: Default horizontal lengths in enumext.

__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.

```
\cs_new_protected:Nn \__enumext_calc_hspace:NNNNNNN
  {
    \dim_compare:nNnT {#1} < { \c_zero_dim }
      {
        \msg_warning:nnnV { enumext } { width-negative }{ labelwidth } #1
        \dim_set:Nn #1 { \dim_abs:n {#1} }
      }
    \dim_compare:nNnT {#2} < { \c_zero_dim }
      {
        \msg_warning:nnnV { enumext } { width-negative }{ labelsep }  #2
        \dim_set:Nn #2 { \dim_abs:n {#2} }
      }
```

If no value has been passed to the labelwidth and labelsep keys, the default values are set for \l__enumext_leftmargin_tmp_X_dim.

```
    \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2} }
```

Whatever the sign and the size of \l__enumext_leftmargin_tmp_X_dim, the values for \itemindent and \leftmargin reduce to two assignments: \itemindent is labelwidth plus labelsep minus list-indent, and \leftmargin is list-offset plus list-indent (§13.43.4).

```
    \dim_set:Nn #6 { #1 + #2 - #4 }
    \dim_set:Nn #5 { #3 + #4 }
  }
\cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

(*End of definition for* __enumext_calc_hspace:NNNNNNN.)

13.43.5 Setting second argument of the lists

__enumext_unnumbered_list:n

When *tagged* PDF is active the LaTeX kernel declares a list as *ordered* if \@nmbrlist is true, which \usecounter always sets, even if the label key has no counter, for example label={} or label={⟨\textbullet⟩}. The function __enumext_unnumbered_list:n takes the level and sets \@nmbrlist to false if the ⟨*label*⟩ of that level does not use the counter of the level, so the list is declared as *unordered*. The environments restore \@nmbrlist right after opening the list, since the \item command needs it to step the counter.

```
\cs_new_protected:Nn \__enumext_unnumbered_list:n
  {
    \str_if_in:vnF
      { l__enumext_label_ #1 _tl } { enumX #1 }
      { \legacy_if_set_false:n { @nmbrlist } }
  }
```

(*End of definition for* __enumext_unnumbered_list:n.)

13.43.6 Numbering style of the list

__enumext_list_numbering:n
__enumext_list_numbering_check:nnn

The LaTeX kernel declares every numbered list as *ordered*, whatever the counter style of the ⟨*label*⟩. The function __enumext_list_numbering:n takes the level and looks in \l__enumext_label_X_tl for the standard counter styles \arabic, \alph, \Alph, \roman and \Roman. When one of them is found, the variable \l__enumext_list_class_tl stores the name of the attribute class that declares the list as Decimal, LowerAlpha, UpperAlpha, LowerRoman or UpperRoman respectively. The function __enumext_list_numbering_check:nnn tests one style and keeps the first one found.

```
\cs_new_protected:Npn \__enumext_list_numbering_check:nnn #1 #2 #3
  {
    \tl_if_empty:NT \l__enumext_list_class_tl
      {
        \str_if_in:vnT { l__enumext_label_ #1 _tl } { #2 }
          { \tl_set:Nn \l__enumext_list_class_tl { enumext-#3 } }
      }
  }
\cs_new_protected:Npn \__enumext_list_numbering:n #1
  {
    \tl_clear:N \l__enumext_list_class_tl
    \str_if_in:vnT { l__enumext_label_ #1 _tl } { enumX #1 }
      {
        \__enumext_list_numbering_check:nnn {#1} { \arabic } { Decimal }
        \__enumext_list_numbering_check:nnn {#1} { \alph } { LowerAlpha }
        \__enumext_list_numbering_check:nnn {#1} { \Alph } { UpperAlpha }
        \__enumext_list_numbering_check:nnn {#1} { \roman } { LowerRoman }
        \__enumext_list_numbering_check:nnn {#1} { \Roman } { UpperRoman }
      }
  }
```

(*End of definition for* __enumext_list_numbering:n *and* __enumext_list_numbering_check:nnn.)

__enumext_apply_list_class:

The kernel chooses the attribute class of a list in \legacylistsetup, after running the second argument of the list, so the class cannot be set from there. When the kernel provides \legacylistsetup and the variable that stores the class, the function __enumext_apply_list_class: is added at the end of \legacylistsetup. If \l__enumext_list_class_tl is not empty it replaces the class chosen by the kernel and then it is emptied, so the lists that are not created by the package are not affected. The classes are declared with the role/new-attribute key of tagpdf.

```
\cs_new_protected:Npn \__enumext_apply_list_class:
  {
    \tl_if_empty:NF \l__enumext_list_class_tl
      {
        \tl_set_eq:NN \l__tag_L_attr_class_tl \l__enumext_list_class_tl
        \tl_clear:N \l__enumext_list_class_tl
      }
  }
\bool_lazy_and:nnT
  { \cs_if_exist_p:N \legacylistsetup }
  { \cs_if_exist_p:N \l__tag_L_attr_class_tl }
  {
    \tagpdfsetup
      {
        role/new-attribute = { enumext-Decimal }
          { /O /List /ListNumbering /Decimal },
        role/new-attribute = { enumext-LowerAlpha }
          { /O /List /ListNumbering /LowerAlpha },
        role/new-attribute = { enumext-UpperAlpha }
          { /O /List /ListNumbering /UpperAlpha },
        role/new-attribute = { enumext-LowerRoman }
          { /O /List /ListNumbering /LowerRoman },
        role/new-attribute = { enumext-UpperRoman }
          { /O /List /ListNumbering /UpperRoman }
      }
    \cs_new_eq:NN \__enumext_legacy_list_setup: \legacylistsetup
    \cs_gset_protected:Npn \legacylistsetup
      {
        \__enumext_legacy_list_setup:
        \__enumext_apply_list_class:
      }
  }
```

(*End of definition for* __enumext_apply_list_class:.)

©2024–2026 by Pablo González L

__enumext_list_arg_two_i:
__enumext_list_arg_two_ii:
__enumext_list_arg_two_iii:
__enumext_list_arg_two_iv:
__enumext_list_arg_two_v:

\leftmargini, \leftmarginii, \leftmarginiii and \leftmarginiv are "not set"; in this case, the parameters for vertical and horizontal list spacing per level are set directly. The function __enumext_list_numbering:n (§13.43.6) is also called to store the numbering style of the level.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \cs_new_protected:cpn { __enumext_list_arg_two_ #1 : }
      {
        \__enumext_calc_hspace:ccccccc
          { l__enumext_labelwidth_ #1 _dim } { l__enumext_labelsep_ #1 _dim }
          { l__enumext_listoffset_ #1 _dim } { l__enumext_leftmargin_tmp_ #1 _dim }
          { l__enumext_leftmargin_ #1 _dim } { l__enumext_itemindent_ #1 _dim }
          { l__enumext_leftmargin_tmp_ #1 _bool }
        \clist_map_inline:nn
          { labelsep, labelwidth, itemindent, leftmargin, rightmargin,
            listparindent }
          { \dim_set_eq:cc {####1} { l__enumext_####1 _ #1 _dim } }
        \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
          { \skip_set_eq:cc {####1} { l__enumext_####1 _ #1 _skip } }
        \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
          { \int_set_eq:cc {@####1} { l__enumext_####1 _ #1 _int } }
        \usecounter { enumX #1 }
        \setcounter { enumX #1 }
          { \int_eval:n { \int_use:c { l__enumext_start_ #1 _int } - 1 } }
        \__enumext_unnumbered_list:n {#1}
        \__enumext_list_numbering:n {#1}
        \str_if_eq:nnTF {#1} { v }
          {
            \__enumext_keyans_redefine_item:
            \__enumext_keyans_make_label:
            \__enumext_keyans_ref:
            \__enumext_keyans_fake_item_indent:
            \bool_if:cT { l__enumext_show_length_ #1 _bool }
              {
                \msg_term:nnnn { enumext } { list-lengths-not-nested } { v }
                  { keyans }
              }
          }
          {
            \__enumext_redefine_item:
            \__enumext_make_label:
            \__enumext_standard_ref:
            \__enumext_fake_item_indent:
            \bool_if:cT { l__enumext_show_length_ #1 _bool }
              {
                \msg_term:nnne { enumext } { list-lengths } {#1}
                  { \int_use:N \l__enumext_level_int }
              }
          }
      }
  }
\clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(*End of definition for* __enumext_list_arg_two_i: *and others.*)

__enumext_list_arg_two_vii:
__enumext_list_arg_two_viii:

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 LaTeX release 2026-06-01, \setcounter is moved to \begin{list}{⟨*arg one*⟩}{⟨*arg two*⟩}\setcounter in the definition of the environments enumext* and keyans* (§13.49, §13.50). This resolves the nesting problem reported by Tim Prescott (@teepeemm) in enumext* nested in enumext starts at 2 with DocumentMetadata this was caused by the latex-lab-block[15] update in this release.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \cs_new_protected:cpn { __enumext_list_arg_two_ #1 : }
      {
        \bool_set_true:c { l__enumext_leftmargin_tmp_ #1 _bool }
        \dim_zero:c { l__enumext_leftmargin_tmp_ #1 _dim }
        \__enumext_calc_hspace:ccccccc
```

```
      { l__enumext_labelwidth_ #1 _dim } { l__enumext_labelsep_ #1 _dim }
      { l__enumext_listoffset_ #1 _dim } { l__enumext_leftmargin_tmp_ #1 _dim }
      { l__enumext_leftmargin_ #1 _dim } { l__enumext_itemindent_ #1 _dim }
      { l__enumext_leftmargin_tmp_ #1 _bool }
    \clist_map_inline:nn
      { labelsep, labelwidth, itemindent, leftmargin, rightmargin,
        listparindent }
      { \dim_set_eq:cc {####1} { l__enumext_####1 _ #1 _dim } }
    \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
      { \skip_set_eq:cc {####1} { l__enumext_####1 _ #1 _skip } }
    \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
      { \int_set_eq:cc {@####1} { l__enumext_####1 _ #1 _int } }
    \skip_set_eq:Nc \parsep  { l__enumext_itemsep_ #1 _skip }
    \skip_zero:N \partopsep
    \usecounter { enumX #1 }
    % \setcounter { enumX #1 }
    %   { \int_eval:n { \int_use:c { l__enumext_start_ #1 _int } - 1 } }
    \__enumext_unnumbered_list:n {#1}
    \__enumext_list_numbering:n {#1}
    \__enumext_starred_ref:
    \str_if_eq:nnTF {#1} { vii }
      {
        \__enumext_fake_item_indent_vii:
        \bool_if:cT { l__enumext_show_length_vii_bool }
          { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii }
            { enumext* } }
      }
      {
        \__enumext_fake_item_indent_viii:
        \bool_if:cT { l__enumext_show_length_ #1 _bool }
          { \msg_term:nnnn { enumext } { list-lengths-not-nested } {#1}
            { keyans* } }
      }
  }
}
\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:`.)

13.44 The environment `enumext`

13.44.1 Definition of variables for `enumext`

`\l__enumext_store_upper_level_X_bool`

Internal variables used by the *"storing structure"* of the `enumext` environment when it is nested.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \bool_new:c { l__enumext_store_upper_level_ #1 _bool }
  }
\clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\l__enumext_store_upper_level_X_bool`.)

13.44.2 Definition of error messages for `enumext`

`list-too-deep`

Message for the nesting depth of the environment `enumext`.

```
\msg_new:nnnn { enumext } { list-too-deep }
  {
    Too~deep~nesting~for~'enumext'~\msg_line_context:.
  }
  {
    The~maximum~level~of~nesting~is~4.
  }
```

(*End of definition for* `list-too-deep`.)

13.44.3 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 it 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 and sets `\l__enumext_standard_bool` to

true. Finally it calls the function __enumext_is_on_first_level:, and only if the environment is NOT nested and is at the *"first level"* it sets \l__enumext_standard_first_bool to true.

```
\cs_new_protected:Nn \__enumext_safe_exec:
  {
    \__enumext_is_not_nested:
    \__enumext_internal_mini_page:
    \int_incr:N \l__enumext_level_int
    \int_compare:nNnT { \l__enumext_level_int } > { 4 }
      { \msg_fatal:nn { enumext } { list-too-deep } }
    \bool_set_true:N \l__enumext_standard_bool
    \bool_set_false:N \l__enumext_starred_bool
    \__enumext_is_on_first_level:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_parse_keys:n
  {
    \tl_if_novalue:nF {#1}
      {
        \str_clear:N \l__enumext_series_name_str
        \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
          {
            \keys_set:nn { enumext / level-1 } {#1}
            \bool_if:NF \l__enumext_print_keyans_cmd_bool
              {
                \__enumext_parse_series:n {#1}
              }
            \__enumext_nested_base_line_fix:
          }
          {
            \keys_set:en
              { enumext / level-\int_use:N \l__enumext_level_int } {#1}
            \bool_if:NF \l__enumext_print_keyans_cmd_bool
              {
                \__enumext_parse_series:n {#1}
              }
          }
        \__enumext_store_active_keys:n {#1}
      }
  }
```

(*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*.

```
\cs_new_protected:Nn \__enumext_start_store_level:
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_store_active_bool }
        { \bool_not_p:n { \l__enumext_keyans_env_bool } }
        { \bool_if_p:N \g__enumext_standard_bool }
      }
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
          {
            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
            \__enumext_store_level_open:
          }
      }
```

If enumext are nested in enumext* add __enumext_store_level_open: to preserve the *"storing structure"*.

```
    \bool_lazy_all:nT
```

```
      {
        { \bool_if_p:N \l__enumext_store_active_bool }
        { \bool_not_p:n { \l__enumext_keyans_env_bool } }
        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
      }
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
            \__enumext_store_level_open:
          }
      }
  }
```

(*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*`.

```
\cs_new_protected:Nn \__enumext_stop_store_level:
  {
    \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
      {
        \__enumext_store_level_close:
      }
  }
```

(*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=0pt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```
\cs_new_protected:Nn \__enumext_multicols_start:
  {
    \int_compare:nNnT
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \dim_compare:nNnT
          { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } }
          = { \c_zero_dim }
          {
            \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
              {
                ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
                  + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
                ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
                - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
              }
          }
        \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
          {
            \dim_zero:N \columnseprule
          }
```

The *vertical spacing* settings for the `multicols` environment are calculated using the function `\__enumext_multi_addvspace:`, the *"vertical adjust spacing"* is applied, and then the `multicols` environment is started.

```
        \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
          {
            \skip_zero:N \multicolsep
            \__enumext_multi_addvspace:
          }
        \raggedcolumns
        \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
      }
  }
```

(*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:`.

```
\cs_new_protected:Nn \__enumext_multicols_stop:
  {
    \int_compare:nNnTF
      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level:
        \end{multicols}
        \__enumext_unskip_unkern:
        \__enumext_unskip_unkern:
        \par
        \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
          {
            \__enumext_mini_capture_multicols:
          }
          {
            \addvspace
              { \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
          }
      }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level:
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_before_list:
  {
    \__enumext_vspace_above:
    \__enumext_before_args_exec:
    \__enumext_check_ans_active:
```

When the `mini-env` key is active it will set the value of the `\l__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 `\l__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\l__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.

```
    \dim_compare:nNnT
      { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
      > { \c_zero_dim }
      {
        \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
          {
            \linewidth
            - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
            - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
          }
```

The boolean variable `\l__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.

```
        \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_minipage_add_space:
        \noindent
        \__enumext_mini_page
          { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
```

©2024–2026 by Pablo González L

```
      }
    \__enumext_multicols_start:
  }
```

(*End of definition for* `\__enumext_before_list:`.)

`\__enumext_second_part:` First of all, the function `\__enumext_second_part:` will check the state of the boolean variable `\l__enumext_minipage_active_X_bool`. If it is `true`, a small test is executed to check if `\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.

```
\cs_new_protected:Nn \__enumext_second_part:
  {
    \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
      {
        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
          {
            \msg_warning:nn { enumext } { missing-miniright }
            \miniright
          }
        \int_gzero:N \g__enumext_minipage_stat_int
        \__enumext_unskip_unkern: % remove topsep + [partopsep]
        \end__enumext_mini_page
        \__enumext_mini_apply_end:
      }
      {
        \__enumext_multicols_stop:
      }
```

Now the function `\__enumext_after_stop_list:`, used by the key `after`, is executed. After that come the function `\__enumext_check_ans_key_hook:`, used by the key `check-ans`, and the function `\__enumext_vspace_below:`, used by the keys `below` and `below*`. Finally it sets `\l__enumext_standard_bool` to false and calls the function `\__enumext_standard_save_counter:`, used by the `series`, `resume` and `resume*` keys.

```
    \__enumext_after_stop_list:
    \__enumext_check_ans_key_hook:
    \__enumext_vspace_below:
    \bool_set_false:N \l__enumext_standard_bool
    \bool_if:NF \l__enumext_print_keyans_cmd_bool
      {
        \__enumext_standard_save_counter:
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_set_item_width:
  {
    \dim_set:Nn \itemwidth { \linewidth }
    \dim_compare:nT
      {
        \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
      }
      {
        \dim_sub:Nn \itemwidth
          {
            \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
          }
      }
  }
```

(*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 (§13.44.3) 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 (§13.44.3) and executes the code set by the key first.

```
\NewDocumentEnvironment{enumext}{ O{} }
  {
    \__enumext_safe_exec:
    \__enumext_parse_keys:n {#1}
    \__enumext_before_list:
    \__enumext_start_store_level:
    \__enumext_start_list:nn
      { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
      {
        \use:c { __enumext_list_arg_two_ \__enumext_level: : }
        \__enumext_before_keys_exec:
      }
    \legacy_if_set_true:n { @nmbrlist }
    \__enumext_set_item_width:
    \__enumext_after_args_exec:
  }
  {
    \__enumext_second_part:
  }
```

(*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.

```
\__enumext_after_env:nn {enumext}
  {
    \__enumext_execute_after_env:
  }
```

13.45 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"*.

13.45.1 Definition of error messages for keyans

keyans-nested Message used by keyans environment.

```
\msg_new:nnnn { enumext } { keyans-nested }
  {
    The~environment~'keyans'~can't~be~nested~\msg_line_context:.
  }
  {
    Close~the~first~'keyans'~environment~before~starting~
    another~one.
  }
```

(*End of definition for* keyans-nested.)

13.45.2 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.

```
\cs_new_protected:Nn \__enumext_keyans_safe_exec:
  {
    \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
      {
        \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
      }
    \int_incr:N \l__enumext_keyans_level_int
    \bool_set_true:N \l__enumext_keyans_env_bool
    \__enumext_keyans_name_and_start:
    % Set false for interfering with enumext nested in keyans
    % (yes, its possible and crayze)
    \bool_set_false:N \l__enumext_store_active_bool
    \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
      {
```

```
      \msg_error:nn { enumext } { keyans-nested }
    }
  \int_compare:nNnT { \l__enumext_level_int } > { 1 }
    {
      \msg_error:nn { enumext } { keyans-wrong-level }
    }
}
```

(*End of definition for* `\__enumext_keyans_safe_exec:`.)

`\__enumext_keyans_parse_keys:n` Parse [⟨*key* = *val*⟩] for `keyans` environment.

```
\cs_new_protected:Nn \__enumext_keyans_parse_keys:n
  {
    \keys_set:nn { enumext / keyans } {#1}
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_before_list_v:
  {
    \__enumext_vspace_above_v:
    \__enumext_before_args_exec_v:
    \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_minipage_left_v_dim
          {
            \linewidth
            - \l__enumext_minipage_right_v_dim
            - \l__enumext_minipage_hsep_v_dim
          }
        \bool_set_true:N \l__enumext_minipage_active_v_bool
        \int_gincr:N \g__enumext_minipage_stat_int
        \__enumext_keyans_minipage_add_space:
        \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
      }
    \__enumext_keyans_multicols_start:
  }
\cs_new_protected:Nn \__enumext_keyans_multicols_start:
  {
    \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
      {
        \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
          {
            \dim_set:Nn \l__enumext_columns_sep_v_dim
              {
                (
                  \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
                ) / \l__enumext_columns_v_int
                - \l__enumext_listoffset_v_dim
              }
          }
        \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
        \dim_zero:N \columnseprule % no rule here
        \bool_if:NF \l__enumext_minipage_active_v_bool
          {
            \skip_zero:N \multicolsep
            \__enumext_keyans_multi_addvspace:
          }
        \raggedcolumns
        \begin{multicols}{ \l__enumext_columns_v_int }
      }
  }
\cs_new_protected:Nn \__enumext_keyans_multicols_stop:
  {
    \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
      {
        \__enumext_stop_list:
        \end{multicols}
        \__enumext_unskip_unkern:
        \__enumext_unskip_unkern:
```

```
      \par
      \bool_if:NTF \l__enumext_minipage_active_v_bool
        {
          \__enumext_mini_capture_multicols:
        }
        {
          \addvspace{ \l__enumext_multicols_below_v_skip }
        }
    }
    {
      \__enumext_stop_list:
    }
  }
\cs_new_protected:Nn \__enumext_second_part_v:
  {
    \bool_if:NTF \l__enumext_minipage_active_v_bool
      {
        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
          {
            \msg_warning:nn { enumext } { missing-miniright }
            \miniright
          }
        \int_gzero:N \g__enumext_minipage_stat_int
        \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
        \end__enumext_mini_page
        \__enumext_mini_apply_end:
      }
      {
        \__enumext_keyans_multicols_stop:
      }
    \bool_set_false:N \l__enumext_keyans_env_bool
    \__enumext_after_stop_list_v:
    \__enumext_vspace_below_v:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_keyans_set_item_width:
  {
    \dim_set:Nn \itemwidth { \linewidth }
    \dim_compare:nT
      {
        \l__enumext_listoffset_v_dim != \c_zero_dim
      }
      {
        \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
      }
  }
```

(*End of definition for* `\__enumext_keyans_set_item_width:`.)

`keyans` Defines the environment `keyans`, also based on lists. It accepts an optional `[`⟨*key* = *val*⟩`]` argument in '`#1`'. It first checks that the environment is available and where it is running, and processes the ⟨*keys*⟩. Then it opens the list, sets `\itemwidth` (§13.45.2) and executes the code set by the key `first`. In the second part it checks the use of `\item*` and finishes the environment.

```
\NewDocumentEnvironment{keyans}{ O{} }
  {
    \__enumext_keyans_safe_exec:
    \__enumext_keyans_parse_keys:n {#1}
    \__enumext_before_list_v:
    \__enumext_start_list:nn
      { \tl_use:N \l__enumext_label_v_tl }
      {
        \__enumext_list_arg_two_v:
        \__enumext_before_keys_exec_v:
      }
    \legacy_if_set_true:n { @nmbrlist }
    \__enumext_keyans_set_item_width:
```

```
    \__enumext_after_args_exec_v:
  }
  {
    \__enumext_check_starred_cmd:n { item }
    \__enumext_second_part_v:
  }
```

(*End of definition for* keyans. *This function is documented on page 18.*)

13.46 Tagging PDF support for non-standard list environments

The LaTeX 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.

13.46.1 Socket for tagging support in 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

First, the necessary sockets and their behavior are defined for enumext* and keyans*.

```
\socket_new:nn {tagsupport/__enumext/starred}{ 1 }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
  {
    \tag_resume:n {#1}
      \tag_struct_begin:n {tag=LI}
        \tag_struct_begin:n {tag=Lbl}
          \tag_mc_begin:n {tag=Lbl}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
  {
          \tag_mc_end:
        \tag_struct_end:n {tag=Lbl}
        \tag_struct_begin:n {tag=LBody}
          % {tag=text-unit} now "semantic-para"
          % (https://github.com/latex3/latex2e/commit/83f054afc042e82a664085b6f2fd7c8d61705fb9)
          \tag_struct_begin:n {tag=semantic-para}
            % {tag=text} now "text-block"
            \tag_struct_begin:n {tag=text-block}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
  {
            \tag_struct_end:n {tag=text-block}
          \tag_struct_end:n {tag=semantic-para} % {tag=text-unit}
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
    \tag_suspend:n {#1}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags-par}
  {
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
    \tag_suspend:n {#1}
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags-break}
  {
          \tag_mc_end:
        \tag_struct_end:n {tag=Lbl}
        \tag_struct_begin:n {tag=LBody}
          \tag_socket_use:nn {para/semantic/begin} { }
          \tag_socket_use:nn {para/textblock/begin} { }
  }
\socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags-break}
  {
          \tag_socket_use:n {para/textblock/end}
          \tag_socket_use:nn {para/semantic/end} { }
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
```

```
    \tag_suspend:n {#1}
  }
```

And now they are wrapped so that they are only active when `\DocumentMetadata` is present.

```
\cs_new_protected_nopar:Nn \__enumext_start_list_tag:n
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
        \socket_use:nn {tagsupport/__enumext/starred} {#1}
      }
  }
\cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
        \socket_use:nn {tagsupport/__enumext/starred} { }
      }
  }
\cs_new_protected_nopar:Nn \__enumext_stop_start_break_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags-break}
        \socket_use:nn {tagsupport/__enumext/starred} { }
      }
  }
\cs_new_protected_nopar:Nn \__enumext_stop_list_tag:n
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
        \socket_use:nn {tagsupport/__enumext/starred} {#1}
      }
  }
```

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 LaTeX 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.

```
\cs_new_protected_nopar:Nn \__enumext_stop_item_break_tag:n
  {
    \IfDocumentMetadataT
      {
        \mode_if_vertical:TF
          {
            \par
            \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags-par}
          }
          {
            \socket_assign_plug:nn
              {tagsupport/__enumext/starred} {stop-list-tags-break}
          }
        \socket_use:nn {tagsupport/__enumext/starred} {#1}
      }
  }
```

©2024–2026 by Pablo González L

(*End of definition for* `start-list-tags` *and others.*)

13.46.2 Socket for tagging support in `keyanspic`

`start-list-tags`
`stop-start-tags`
`stop-list-tags`
`\__enumext_anspic_start_list_tag:`
`\__enumext_anspic_stop_start_list_tag:`
`\__enumext_anspic_stop_list_tag:`

First, the necessary `sockets` and their behavior are defined for the `keyanspic` environment.

```
\socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
  {
    \tag_resume:n {keyanspic}
      \tag_struct_begin:n {tag=LI}
        \tag_struct_begin:n {tag=Lbl}
          \tag_mc_begin:n {tag=Lbl}
  }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
  {
          \tag_mc_end:
        \tag_struct_end:n {tag=Lbl}
        \tag_struct_begin:n {tag=LBody}
          \tag_struct_begin:n {tag=semantic-para}
            \tag_struct_begin:n {tag=text-block}
              \tag_mc_begin:n {tag=text-block}
  }
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
  {
              \tag_mc_end:
            \tag_struct_end:n {tag=text-block}
          \tag_struct_end:n {tag=semantic-para}
        \tag_struct_end:n {tag=LBody}
      \tag_struct_end:n {tag=LI}
    \tag_suspend:n {keyanspic}
  }
```

And now they are wrapped so that they are only active when `\DocumentMetadata` is present.

```
\cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
  }
\cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
  }
\cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
  {
    \IfDocumentMetadataT
      {
        \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
        \socket_use:n {tagsupport/__enumext/keyanspic}
      }
  }
```

(*End of definition for* `start-list-tags` *and others.*)

13.47 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a `list` based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use `\item`. The ⟨*contents*⟩ are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the ⟨*label*⟩ centered *"above"* or *"below"*, adjusting *widths* and *position* according to the options passed to the environment.

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?.

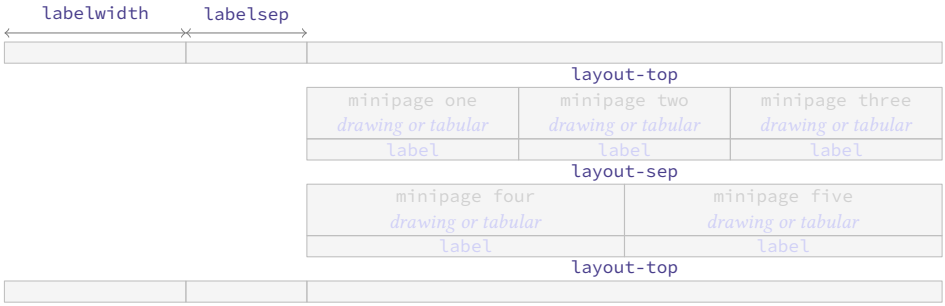


Figure 12: Representation of the keyanspic spacing in enumext.

13.47.1 Definition of variables for keyanspic and \anspic

\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_ht_dim
\l__enumext_anspic_label_dp_dim
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim
\l__enumext_anspic_gap_dim
\l__enumext_anspic_label_sep_skip
\l__enumext_anspic_top_skip
\l__enumext_anspic_label_sep_bool
\l__enumext_anspic_top_bool

Internal variables used by keyanspic environment and \anspic command (§13.47.3).

```
\seq_new:N  \l__enumext_anspic_args_seq
\dim_new:N  \l__enumext_anspic_mini_width_dim
\int_new:N  \l__enumext_anspic_above_int
\int_new:N  \l__enumext_anspic_below_int
\bool_new:N \l__enumext_anspic_label_above_bool
\str_new:N  \l__enumext_anspic_mini_pos_str
\box_new:N  \l__enumext_anspic_label_box
\box_new:N  \l__enumext_anspic_body_box
\dim_new:N  \l__enumext_anspic_label_ht_dim
\dim_new:N  \l__enumext_anspic_label_dp_dim
\dim_new:N  \l__enumext_anspic_label_htdp_dim
\dim_new:N  \l__enumext_anspic_body_htdp_dim
\dim_new:N  \l__enumext_anspic_gap_dim
\skip_new:N \l__enumext_anspic_label_sep_skip
\skip_new:N \l__enumext_anspic_top_skip
\bool_new:N \l__enumext_anspic_label_sep_bool
\bool_new:N \l__enumext_anspic_top_bool
```

(*End of definition for* `\l__enumext_anspic_args_seq` *and others.*)

13.47.2 Definition of error messages for keyanspic

keyanspic-nested
keyanspic-wrong-level
keyanspic-item-cmd

Messages used by keyanspic environment.

```
\msg_new:nnnn { enumext } { keyanspic-nested }
  {
    The~environment~'keyanspic'~can't~be~nested~\msg_line_context:.
  }
  {
    Close~the~first~'keyanspic'~environment~before~starting~
    another~one.
  }
\msg_new:nnnn { enumext } { keyanspic-wrong-level }
  {
    Wrong~level~position~for~'keyanspic'~\msg_line_context:.
  }
  {
    The~environment~'keyanspic'~can~only~be~in~the~first~level.
  }
\msg_new:nnnn { enumext } { keyanspic-item-cmd }
  {
    Can't~use~'\token_to_str:N \item'~in~'keyanspic'~\msg_line_context:.
  }
  {
    Remove~'\token_to_str:N \item'~from~the~'keyanspic'~environment.
  }
```

(*End of definition for* `keyanspic-nested`, `keyanspic-wrong-level`, *and* `keyanspic-item-cmd`.)

13.47.3 The environment keyanspic

label-pos
label-sep
layout-sty
layout-sep
layout-top
mark-ans

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 {⟨*n° upper, n° lower*⟩} and determines the number of minipage environments in which all arguments of \anspic are printed at the "upper" and

©2024–2026 by Pablo González L

"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`.

```
\keys_define:nn { enumext / keyanspic }
  {
    label-pos  .choices:nn =
      { above , below }
      {
        \int_case:nn { \l_keys_choice_int }
          {
            {1}
              {
                \bool_set_true:N \l__enumext_anspic_label_above_bool
                \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
              }
            {2}
              {
                \bool_set_false:N \l__enumext_anspic_label_above_bool
                \str_set:Nn \l__enumext_anspic_mini_pos_str { b }
              }
          }
      },
    label-pos  .initial:n = below,
    label-pos  .value_required:n = true,
    label-pos / unknown .code:n =
      \msg_error:nneee { enumext } { unknown-choice }
        { label-pos } { above,~ below } { \exp_not:n {#1} },
    label-sep  .code:n =
      {
        \skip_set:Nn \l__enumext_anspic_label_sep_skip {#1}
        \bool_set_true:N \l__enumext_anspic_label_sep_bool
      },
    label-sep  .value_required:n = true,
    layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
    layout-sty .value_required:n = true,
    layout-sep .meta:nn = { enumext / keyans } { parsep = {#1} },
    layout-sep .value_required:n = true,
    layout-top .code:n =
      {
        \skip_set:Nn \l__enumext_anspic_top_skip {#1}
        \bool_set_true:N \l__enumext_anspic_top_bool
      },
    layout-top .value_required:n = true,
    mark-ans   .meta:nn = { enumext / keyans } { mark-ans = {#1} },
    mark-ans   .value_required:n = true,
    mark-pos   .meta:nn = { enumext / keyans } { mark-pos = {#1} },
    mark-pos   .value_required:n = true,
    mark-sep   .meta:nn = { enumext / keyans } { mark-sep = {#1} },
    mark-sep   .value_required:n = true,
    save-sep   .meta:nn = { enumext / keyans } { save-sep = {#1} },
    save-sep   .value_required:n = true,
    wrap-opt   .meta:nn = { enumext / keyans } { wrap-opt = {#1} },
    wrap-opt   .value_required:n = true,
    wrap-ans*  .meta:nn = { enumext / keyans } { wrap-ans* = {#1} },
    wrap-ans*  .value_required:n = true,
    show-ans   .meta:nn = { enumext / keyans } { show-ans = {#1} },
    show-ans   .value_required:n = true,
    show-pos   .meta:nn = { enumext / keyans } { show-pos = {#1} },
    show-pos   .value_required:n = true,
    unknown    .code:n =
      {
        \tl_set:Ne \l__enumext_envir_name_tl { keyanspic }
        \__enumext_keyans_unknown_keys:n {#1}
      },
  }
```

(*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:
__enumext_anspic_top_glue:

The function `\__enumext_keyans_pic_safe_exec:` check the nested level position inside the `enumext` environment.

```
\cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
```

```
  {
    \int_incr:N \l__enumext_keyans_pic_level_int
    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyanspic-nested }
      }
    \__enumext_keyans_name_and_start:
  }
```

Parse [⟨*key* = *val*⟩] for `keyanspic` environment.

```
\cs_new_protected:Nn \__enumext_keyans_pic_parse_keys:n
  {
    \tl_if_novalue:nF {#1}
      {
        \keys_set:nn { enumext / keyanspic } {#1}
      }
  }
```

The function `\__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```
\cs_new_protected:Nn \__enumext_keyans_pic_skip_abs:N
  {
    \dim_compare:nNnT {#1} < { \c_zero_dim }
      {
        \skip_set:Nn #1 { -#1 }
      }
  }
```

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` and `parsep` 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.

```
\cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
  {
    \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
    \__enumext_list_arg_two_v:
    \__enumext_keyans_pic_skip_abs:N \parsep
```

Now the counter `enumXv` of the `keyans` environment is incremented and the dimensions of the ⟨*label*⟩ are saved in `\l__enumext_anspic_label_ht_dim`, `\l__enumext_anspic_label_dp_dim` and `\l__enumext_anspic_label_htdp_dim`, used by `\anspic`, the counter is restored if the key `label-pos` is set to *above*. If the key `label-sep` is not set, its value is `1pt` for *above* and `5pt` for *below*.

```
    \stepcounter { enumXv }
    \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
    \dim_set:Nn \l__enumext_anspic_label_ht_dim
      { \box_ht:N \l__enumext_anspic_label_box }
    \dim_set:Nn \l__enumext_anspic_label_dp_dim
      { \box_dp:N \l__enumext_anspic_label_box }
    \dim_set:Nn \l__enumext_anspic_label_htdp_dim
      {
        \l__enumext_anspic_label_ht_dim + \l__enumext_anspic_label_dp_dim
      }
    \bool_if:NT \l__enumext_anspic_label_above_bool
      {
        \addtocounter { enumXv } { -1 }
      }
    \bool_if:NF \l__enumext_anspic_label_sep_bool
      {
        \bool_if:NTF \l__enumext_anspic_label_above_bool
          {
            \skip_set:Nn \l__enumext_anspic_label_sep_skip { 1pt }
          }
          {
            \skip_set:Nn \l__enumext_anspic_label_sep_skip { 5pt }
          }
      }
```

Finally the value of \leftmargin is *adjusted*, then \topsep, \listparindent, \partopsep and \itemsep are set to zero so that the *horizontal* and *vertical* space is not affected. The spaces *"above"* and *"below"* the environment are added by __enumext_anspic_top_glue: and __enumext_anspic_bottom_glue:.

```
    \dim_add:Nn \leftmargin
      { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
    \ignorespaces
    \skip_zero:N \topsep
    \dim_zero:N  \listparindent
    \skip_zero:N \partopsep
    \skip_zero:N \itemsep
  }
```

The function __enumext_anspic_top_glue: adds the *vertical space "above"* the first row. If the key layout-top is set, that value is used. Otherwise the value of \topsep taken from the keyans environment is added, plus the space that TEX would add between the previous line and a line with the height of the ⟨*label*⟩, calculated with \baselineskip, \lineskip and \lineskiplimit, so that the space is the same as *"above"* the first \item of keyans. Finally \nointerlineskip avoids adding more space.

```
\cs_new_protected:Nn \__enumext_anspic_top_glue:
  {
    \bool_if:NTF \l__enumext_anspic_top_bool
      {
        \addvspace { \l__enumext_anspic_top_skip }
      }
      {
        \addvspace { \l__enumext_topsep_v_skip }
        \dim_compare:nNnF { \prevdepth } < { -999pt }
          {
            \dim_set:Nn \l__enumext_anspic_gap_dim
              {
                \baselineskip - \prevdepth - \l__enumext_anspic_label_ht_dim
              }
            \dim_compare:nNnTF { \l__enumext_anspic_gap_dim } < { \lineskiplimit }
              {
                \skip_vertical:N \lineskip
              }
              {
                \skip_vertical:n { \l__enumext_anspic_gap_dim }
              }
          }
      }
    \nointerlineskip
  }
```

The function __enumext_anspic_bottom_glue: adds the *vertical space "below"* the last row. If the key layout-top is set, that value is used and \nointerlineskip avoids adding more space. Otherwise \prevdepth takes the depth of the ⟨*label*⟩ and the value of \topsep taken from the keyans environment is added, so that the space is the same as *"below"* the last \item of keyans. It must be executed before closing the list environment and \parskip is set to zero, otherwise \endtrivlist adds it to the space.

```
\cs_new_protected:Nn \__enumext_anspic_bottom_glue:
  {
    \bool_if:NTF \l__enumext_anspic_top_bool
      {
        \nointerlineskip
        \skip_vertical:N \l__enumext_anspic_top_skip
      }
      {
        \prevdepth \l__enumext_anspic_label_dp_dim
        \skip_vertical:N \l__enumext_topsep_v_skip
      }
    \skip_zero:N \parskip
  }
```

(*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. The space *"above"* is added with __enumext_anspic_top_glue: before \item, and \nointerlineskip avoids adding the space between lines for the first row.

```
\NewDocumentEnvironment{keyanspic}{ o }
  {
    \__enumext_keyans_pic_safe_exec:
    \__enumext_keyans_pic_parse_keys:n {#1}
    \begin{list} { } { \__enumext_keyans_pic_arg_two: }
    \IfDocumentMetadataT
      {
        \tag_suspend:n {list}
      }
    \__enumext_anspic_top_glue:
    \item[] \scan_stop:
    \nointerlineskip
    \RenewDocumentCommand \item {}
      {
        \msg_error:nn { enumext } { keyanspic-item-cmd }
      }
    \IfDocumentMetadataT
      {
        \tag_resume:n {keyanspic}
        \tag_socket_use:n { para/off }
        \tag_suspend:n {keyanspic}
      }
  }
```

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. The space *"below"* is added before closing the list environment.

```
  {
    \IfDocumentMetadataT
      {
        \tag_resume:n {keyanspic}
      }
    \__enumext_anspic_exec:
    \IfDocumentMetadataT
      {
        \tag_suspend:n {keyanspic}
      }
    \par
    \__enumext_anspic_bottom_glue:
    \end{list}
    \IfDocumentMetadataT
      {
        \tag_struct_end:n {tag=L}
      }
```

Finally it checks if \anspic* has been used and sets the counter enumXvi to zero.

```
    \__enumext_check_starred_cmd:n { anspic }
    \setcounter { enumXvi } { 0 }
  }
```

(*End of definition for* keyanspic. *This function is documented on page 19.*)

13.47.4 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.

__enumext_anspic_body_dim:n
__enumext_anspic_label:nn
__enumext_anspic_label_pos:nnn
__enumext_anspic_args:nnn
__enumext_anspic_print:n
__enumext_anspic_print:e

If the key label-pos is set "below", 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*.

```
\cs_new_protected:Nn \__enumext_anspic_body_dim:n
  {
```

©2024–2026 by Pablo González L

```
    \bool_if:NF \l__enumext_anspic_label_above_bool
      {
        \IfDocumentMetadataT
          {
            \tag_suspend:n {keyanspic}
          }
        \vbox_set:Nn \l__enumext_anspic_body_box {#1}
        \dim_set:Nn \l__enumext_anspic_body_htdp_dim
          {
            \box_ht_plus_dp:N \l__enumext_anspic_body_box
          }
        \IfDocumentMetadataT
          {
            \tag_resume:n {keyanspic}
          }
      }
  }
```

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.

```
\cs_new_protected:Nn \__enumext_anspic_label:nn
  {
    \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
      {
        \bool_if:nTF {#1}
          {
            \bool_set_true:N \l__enumext_item_wrap_key_bool
            \bool_set_true:N \l__enumext_wrap_label_v_bool
            \__enumext_keyans_save_item_opt:n {#2}
            \__enumext_keyans_addto_prop:n {#2}
            \__enumext_keyans_store_ref:
            \__enumext_keyans_addto_seq:n {#2}
            \int_gincr:N \g__enumext_check_starred_cmd_int
            \__enumext_keyans_show_ans:
            \__enumext_keyans_show_pos:
            \makebox[ \l__enumext_labelwidth_v_dim ][c]
              {
                \tl_use:N \l__enumext_label_font_style_v_tl
                \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
              }
            \skip_horizontal:n { \l__enumext_labelsep_v_dim }
            \__enumext_keyans_show_item_opt:
          }
          {
            \bool_set_false:N \l__enumext_item_wrap_key_bool
            \tl_use:N \l__enumext_label_font_style_v_tl
            \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
          }
      }
  }
```

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. When the ⟨*label*⟩ goes *"below"* it is placed in a box without *height* and *depth* and lowered the *height plus depth* of the *drawing or tabular*, the value of `label-sep` and the *height* of the ⟨*label*⟩, so that it ends below the *drawing or tabular*.

```
\cs_new_protected:Nn \__enumext_anspic_label_pos:nnn
  {
    \stepcounter { enumXvi }
    \__enumext_anspic_body_dim:n {#3}
    \bool_if:NTF \l__enumext_anspic_label_above_bool
      {
        \__enumext_anspic_label:nn {#1} {#2}
      }
      {
        \raisebox
          {
            -\dim_eval:n
              {
```

```
                \l__enumext_anspic_body_htdp_dim
                + \l__enumext_anspic_label_sep_skip
                + \l__enumext_anspic_label_ht_dim
              }
          }
          [ 0pt ] [ 0pt ]
          {
            \__enumext_anspic_label:nn {#1} {#2}
          }
      }
  }
%
```

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. The ⟨*label*⟩ is closed with `\par` and `\nointerlineskip` avoids the space between lines, so the separation is exactly the value of `label-sep`. If the ⟨*label*⟩ goes *"below"*, that separation and the *height plus depth* of the ⟨*label*⟩ are reserved at the end with `\kern`, because the `minipage` environment removes the last glue with `\unskip`.

```
\cs_new_protected:Nn \__enumext_anspic_args:nnn
  {
    \__enumext_anspic_start_list_tag:
    \__enumext_anspic_label_pos:nnn {#1} {#2} {#3}
    \__enumext_anspic_stop_start_list_tag:
    \par
    \nointerlineskip
    \bool_if:NTF \l__enumext_anspic_label_above_bool
      {
        \skip_vertical:N \l__enumext_anspic_label_sep_skip
        #3
      }
      {
        #3
        \par
        \nointerlineskip
        \kern \dim_eval:n
          {
            \l__enumext_anspic_label_sep_skip + \l__enumext_anspic_label_htdp_dim
          }
      }
    \__enumext_anspic_stop_list_tag:
  }
```

The value {⟨*n° upper, n° lower*⟩} 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`.

```
\cs_new_protected:Nn \__enumext_anspic_print:n
  {
    \clist_map_function:nN {#1} \__enumext_anspic_row:n
  }
\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. After each row `\nointerlineskip` avoids the space between lines, so the separation between rows is exactly the value of `\parsep` set by the key `layout-sep`.

```
\cs_new_protected:Nn \__enumext_anspic_row:n
  {
    \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
    \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
    \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
    \int_step_inline:nnn
      { \l__enumext_anspic_above_int + 1 }
      { \l__enumext_anspic_below_int }
      {
        \IfDocumentMetadataT
          {
            \tag_suspend:n {minipage}
          }
```

©2024–2026 by Pablo González L

```
5286          \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{
5287            \l__enumext_anspic_mini_width_dim }
5288            \centering
5289            \seq_item:Nn \l__enumext_anspic_args_seq {##1}
5290          \end{minipage}
5291          \IfDocumentMetadataT
5292            {
5293              \tag_resume:n {minipage}
5294            }
5295        }
5296      \par
5297      \nointerlineskip
5298    }
```

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*.

```
5299 \cs_new_protected:Nn \__enumext_anspic_exec:
5300   {
5301     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
5302       {
5303         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
5304       }
5305       {
5306         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
5307       }
5308   }
```

(*End of definition for* `\__enumext_anspic_body_dim:n` *and others.*)

`\anspic` The command accepts a *starred argument* '`*`' in '`#1`', an optional `[`⟨*content*⟩`]` in '`#2`' and a mandatory `{`⟨*drawing or tabular*⟩`}` 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.

```
5309 \NewDocumentCommand \anspic { s o +m }
5310   {
5311     \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
5312       {
5313         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
5314       }
5315     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5316       {
5317         \msg_error:nn { enumext } { keyanspic-wrong-level }
5318       }
5319     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
5320       {
5321         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
5322       }
5323     \seq_put_right:Nn \l__enumext_anspic_args_seq
5324       {
5325         \__enumext_anspic_args:nnn {#1} {#2} {#3}
5326       }
5327   }
```

(*End of definition for* `\anspic`. *This function is documented on page* 20.)

13.48 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LaTeX 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 ⟨*content*⟩ are captured in a *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the ⟨*content*⟩ passed to `\item`; the *optional argument* (⟨*number*⟩) 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* (⟨*number*⟩).

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 §13.49.2. Because of this, the *standard list environments* and any environment defined using base list or \trivlist can be used in the ⟨*content*⟩ 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 (§13.49.4).

For compatibility with the *tagged* PDF, the environments are closed according to the presence or not of the mini-env key.

13.48.1 Definition of variables

\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_item_width_X_dim
\g__enumext_minipage_active_X_bool
\l__enumext_miniright_code_X_box

Internal variables used by enumext* and keyans* environments. The functions that calculate the item box width and join the item columns use the integers \l__enumext_joined_item_X_int and \l__enumext_joined_item_aux_X_int, the integer \l__enumext_tmpa_X_int and the dimension \l__enumext_item_width_X_dim. The keys mini-env, mini-right and mini-right* (§13.48.5) use the boolean \g__enumext_minipage_active_X_bool and the box \l__enumext_miniright_code_X_box.

```
\cs_set_protected:Nn \__enumext_tmp:n
  {
    \int_new:c  { l__enumext_joined_item_ #1 _int      }
    \int_new:c  { l__enumext_joined_item_aux_ #1 _int  }
    \int_new:c  { l__enumext_tmpa_ #1 _int             }
    \dim_new:c  { l__enumext_item_width_ #1 _dim       }
    \bool_new:c { g__enumext_minipage_active_ #1 _bool }
    \box_new:c  { l__enumext_miniright_code_ #1 _box   }
  }
\clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* \l__enumext_joined_item_X_int *and others.*)

13.48.2 Definition of error messages

nested-horizontal
item-joined
item-joined-columns

Messages used by enumext* and keyans* environments.

```
\msg_new:nnnn { enumext } { nested-horizontal }
  {
    The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~
    \msg_line_context:.
  }
  {
    'enumext*'~and~'keyans*'~are~horizontal~lists~and~can't~be~
    inside~each~other.
  }
\msg_new:nnn { enumext } { item-joined }
  {
    Items~joined~(#1)~>~#2~columns~\msg_line_context:.
  }
\msg_new:nnn { enumext } { item-joined-columns }
  {
    Not~enough~space~to~join~items~(#1)~>~#2~\msg_line_context:.
  }
```

(*End of definition for* nested-horizontal*,* item-joined*, and* item-joined-columns*.*)

13.48.3 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.

```
\cs_new_protected:Nn \__enumext_starred_columns_set_vii:
  {
    \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_columns_sep_vii_dim
          {
            ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
            / \l__enumext_columns_vii_int
          }
      }
    \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
    \dim_set:Nn \l__enumext_item_width_vii_dim
```

```
      {
        ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
        / \l__enumext_columns_vii_int
        - \l__enumext_labelwidth_vii_dim
        - \l__enumext_labelsep_vii_dim
      }
```

When the key `rightmargin` is active the values must be adjusted.

```
    \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
      {
        \dim_sub:Nn \l__enumext_item_width_vii_dim
          {
            ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
            / \l__enumext_columns_vii_int
          }
        \dim_add:Nn \l__enumext_columns_sep_vii_dim
          {
            \l__enumext_rightmargin_vii_dim
          }
      }
  }
```

Same implementation for the `keyans*` environment.

```
\cs_new_protected:Nn \__enumext_starred_columns_set_viii:
  {
    \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_columns_sep_viii_dim
          {
            ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
            / \l__enumext_columns_viii_int
          }
      }
    \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
    \dim_set:Nn \l__enumext_item_width_viii_dim
      {
        ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
        / \l__enumext_columns_viii_int
        - \l__enumext_labelwidth_viii_dim
        - \l__enumext_labelsep_viii_dim
      }
    \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
      {
        \dim_sub:Nn \l__enumext_item_width_viii_dim
          {
            ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
            / \l__enumext_columns_viii_int
          }
        \dim_add:Nn \l__enumext_columns_sep_viii_dim
          {
            \l__enumext_rightmargin_viii_dim
          }
      }
  }
```

(*End of definition for* `\__enumext_starred_columns_set_vii:` *and* `\__enumext_starred_columns_set_viii:`.)

13.48.4 Functions for join item columns

`\__enumext_starred_joined_item_vii:n`
`\__enumext_starred_joined_item_viii:n`

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, is set by the functions `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_viii:n`.

```
\cs_new_protected:Nn \__enumext_starred_joined_item_vii:n
  {
    \int_set:Nn \l__enumext_joined_item_vii_int {#1}
    \int_compare:nNnT
      { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
      {
        \msg_warning:nnee { enumext } { item-joined }
          { \int_use:N \l__enumext_joined_item_vii_int }
```

```
5425          { \int_use:N \l__enumext_columns_vii_int }
5426        \int_set:Nn \l__enumext_joined_item_vii_int
5427          {
5428            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
5429          }
5430      }
5431    \int_compare:nNnT
5432      { \l__enumext_joined_item_vii_int }
5433        >
5434      { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
5435      {
5436        \msg_warning:nnee { enumext } { item-joined-columns }
5437          { \int_use:N \l__enumext_joined_item_vii_int }
5438          {
5439            \int_eval:n
5440              { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
5441          }
5442        \int_set:Nn \l__enumext_joined_item_vii_int
5443          {
5444            \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
5445          }
5446      }
5447    \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
5448      {
5449        \int_set_eq:NN
5450          \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
5451        \int_decr:N \l__enumext_joined_item_aux_vii_int
5452        \int_add:Nn \l__enumext_item_column_pos_vii_int
5453          { \l__enumext_joined_item_aux_vii_int }
5454        \int_gadd:Nn \g__enumext_item_count_all_vii_int
5455          { \l__enumext_joined_item_aux_vii_int }
5456        \dim_set:Nn \l__enumext_joined_width_vii_dim
5457          {
5458            \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
5459            + (
5460                \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
5461                + \l__enumext_columns_sep_vii_dim
5462              ) * \l__enumext_joined_item_aux_vii_int
5463          }
5464        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
5465      }
5466      {
5467        \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
5468        \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
5469      }
5470  }
```

Same implementation for the keyans* environment.

```
5471 \cs_new_protected:Nn \__enumext_starred_joined_item_viii:n
5472   {
5473     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
5474     \int_compare:nNnT { \l__enumext_joined_item_viii_int } >
5475       { \l__enumext_columns_viii_int }
5476       {
5477         \msg_warning:nnee { enumext } { item-joined }
5478           { \int_use:N \l__enumext_joined_item_viii_int }
5479           { \int_use:N \l__enumext_columns_viii_int }
5480         \int_set:Nn \l__enumext_joined_item_viii_int
5481           {
5482             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
5483           }
5484       }
5485     \int_compare:nNnT
5486       { \l__enumext_joined_item_viii_int }
5487         >
5488       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
5489       {
5490         \msg_warning:nnee { enumext } { item-joined-columns }
5491           { \int_use:N \l__enumext_joined_item_viii_int }
5492           {
5493             \int_eval:n
```

```
5494                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
5495              }
5496            \int_set:Nn \l__enumext_joined_item_viii_int
5497              {
5498                \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
5499              }
5500          }
5501        \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
5502          {
5503            \int_set_eq:NN
5504              \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
5505            \int_decr:N \l__enumext_joined_item_aux_viii_int
5506            \int_add:Nn \l__enumext_item_column_pos_viii_int
5507              { \l__enumext_joined_item_aux_viii_int }
5508            \int_gadd:Nn \g__enumext_item_count_all_viii_int
5509              { \l__enumext_joined_item_aux_viii_int }
5510            \dim_set:Nn \l__enumext_joined_width_viii_dim
5511              {
5512                \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
5513                + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
5514                    + \l__enumext_columns_sep_viii_dim
5515                  )*\l__enumext_joined_item_aux_viii_int
5516              }
5517            \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
5518          }
5519          {
5520            \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
5521            \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
5522          }
5523      }
```

(*End of definition for* `\__enumext_starred_joined_item_vii:n` *and* `\__enumext_starred_joined_item_viii:n`.)

13.48.5 Functions for `mini-env`, `mini-right` and `mini-right*` keys

`\__enumext_start_mini_vii:`
`\__enumext_stop_mini_vii:`

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`.

```
5524 \cs_new_protected:Nn \__enumext_start_mini_vii:
5525   {
5526     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
5527       {
5528         \dim_set:Nn \l__enumext_minipage_left_vii_dim
5529           {
5530             \linewidth
5531             - \l__enumext_minipage_right_vii_dim
5532             - \l__enumext_minipage_hsep_vii_dim
5533           }
5534         \bool_set_true:N \l__enumext_minipage_active_vii_bool
5535         \dim_gset_eq:NN
5536           \g__enumext_minipage_right_vii_dim
5537           \l__enumext_minipage_right_vii_dim
5538         \__enumext_mini_addvspace_vii:
5539         \nointerlineskip\noindent
5540         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
5541       }
5542   }
```

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 (§13.49). Before closing the `\__enumext_mini_page` the function `\__enumext_mini_capture_end:` stores the space *"below"*.

```
5543 \cs_new_protected:Nn \__enumext_stop_mini_vii:
5544   {
5545     \bool_if:NTF \l__enumext_minipage_active_vii_bool
```

```
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level_vii:
        \__enumext_mini_capture_end:
        \IfDocumentMetadataT { \tag_resume:n {enumext*} }
        \end__enumext_mini_page
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_vii_bool
      }
      {
        \__enumext_stop_list:
        \__enumext_stop_store_level_vii:
      }
  }
```

(*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 and stored in the variable `\g__enumext_miniright_code_vii_tl` is executed in the `minipage` environment on the *"right side"*. 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 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.

```
\__enumext_after_env:nn {enumext*}
  {
    \bool_if:NT \g__enumext_minipage_active_vii_bool
      {
        \group_begin:
        \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
          \legacy_if_gset_false:n { @minipage }
          \skip_vertical:N \c_zero_skip
          \par\addvspace { \g__enumext_minipage_right_skip }
          \bool_if:NF \g__enumext_minipage_center_vii_bool
            {
              \tl_gput_left:Nn \g__enumext_miniright_code_vii_tl
                {
                  \centering
                }
            }
          \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
            {
              \tl_use:N \g__enumext_miniright_code_vii_tl
            }
          \box_use_drop:N \l__enumext_miniright_code_vii_box
          \skip_vertical:N \c_zero_skip
        \__enumext_endminipage:
        \group_end:
        \__enumext_mini_apply_end:
      }
    \bool_gset_false:N \g__enumext_minipage_active_vii_bool
    \bool_gset_true:N \g__enumext_minipage_center_vii_bool
    \tl_gclear:N \g__enumext_miniright_code_vii_tl
    \dim_gzero:N \g__enumext_minipage_right_vii_dim
    \bool_gset_false:N \g__enumext_starred_bool
  }
```

`\__enumext_start_mini_viii:`
`\__enumext_stop_mini_viii:`

The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

```
\cs_new_protected:Nn \__enumext_start_mini_viii:
  {
    \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_minipage_left_viii_dim
          {
            \linewidth
            - \l__enumext_minipage_right_viii_dim
            - \l__enumext_minipage_hsep_viii_dim
          }
        \bool_set_true:N \l__enumext_minipage_active_viii_bool
```

```
        \dim_gset_eq:NN
          \g__enumext_minipage_right_viii_dim
          \l__enumext_minipage_right_viii_dim
        \__enumext_mini_addvspace_viii:
        \nointerlineskip\noindent
        \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
      }
  }
\cs_new_protected:Nn \__enumext_stop_mini_viii:
  {
    \bool_if:NTF \l__enumext_minipage_active_viii_bool
      {
        \__enumext_stop_list:
        \__enumext_mini_capture_end:
        \IfDocumentMetadataT { \tag_resume:n {keyans*} }
        \end__enumext_mini_page
        \hfill
        \bool_gset_true:N \g__enumext_minipage_active_viii_bool
      }
      {
        \__enumext_stop_list:
      }
  }
\__enumext_after_env:nn {keyans*}
  {
    \bool_if:NT \g__enumext_minipage_active_viii_bool
      {
        \group_begin:
        \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
          \par\addvspace { \g__enumext_minipage_right_skip }
          \bool_if:NF \g__enumext_minipage_center_viii_bool
            {
              \tl_gput_left:Nn \g__enumext_miniright_code_viii_tl
                {
                  \centering
                }
            }
          \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
            {
              \tl_use:N \g__enumext_miniright_code_viii_tl
            }
          \box_use_drop:N \l__enumext_miniright_code_viii_box
        \end__enumext_mini_page
        \group_end:
        \__enumext_mini_apply_end:
      }
    \bool_gset_false:N \g__enumext_minipage_active_viii_bool
    \bool_gset_true:N \g__enumext_minipage_center_viii_bool
    \tl_gclear:N \g__enumext_miniright_code_viii_tl
    \dim_gzero:N \g__enumext_minipage_right_viii_dim
  }
```

(*End of definition for* `\__enumext_start_mini_viii:` *and* `\__enumext_stop_mini_viii:`.)

13.49 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:` (§13.49.2), 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 pdfTEX and LuaTEX.

```
\NewDocumentEnvironment{enumext*}{ o }
  {
    \__enumext_safe_exec_vii:
    \__enumext_parse_keys_vii:n {#1}
    \__enumext_before_list_vii:
```

```
    \__enumext_start_store_level_vii:
    \__enumext_nested_save:
    \__enumext_start_list:nn { }
      {
        \__enumext_list_arg_two_vii:
        \__enumext_before_keys_exec_vii:
      }
    \legacy_if_set_true:n { @nmbrlist }
    \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
    \__enumext_starred_columns_set_vii:
    \item[] \scan_stop:
```

For compatibility with *tagged* PDF in LaTeX release 2026-06-01, `\setcounter` is set at this point in the environment definition, thereby avoiding the problem reported by Tim Prescott (@teepeemm) in enumext* nested in enumext starts at 2 with DocumentMetadata this was caused by the `latex-lab-block`[15] update in this release.

```
    \setcounter { enumXvii }
      { \int_eval:n { \int_use:c { l__enumext_start_vii_int } - 1 } }
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
    \cs_set_eq:NN \item \__enumext_item_switch_vii:
    \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
    \bool_set_false:N \l__enumext_trivlist_bool
    \ignorespaces
  }
  {
    % {tag=text-unit}
    \IfDocumentMetadataT { \tag_struct_end:n {tag=semantic-para} }
    \__enumext_stop_item_tmp_vii:
    \__enumext_remove_extra_parsep_vii:
    \__enumext_after_list_vii:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_safe_exec_vii:
  {
    \__enumext_is_not_nested:
    \__enumext_internal_mini_page:
    \int_incr:N \l__enumext_level_h_int
    \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
      {
        \msg_error:nn { enumext } { nested }
      }
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
      {
        \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
      }
    \bool_set_true:N \l__enumext_starred_bool
    \bool_set_false:N \l__enumext_standard_bool
    \__enumext_is_on_first_level:
  }
```

(*End of definition for* `\__enumext_safe_exec_vii:`.)

`\__enumext_parse_keys_vii:n`

It first checks the state of the variable `\l__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 `\l__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 `\l__enumext_resume_count_vii_bool` in case the key `resume` is set using `\setenumext{enumext*}{resume}`.

```
\cs_new_protected:Nn \__enumext_parse_keys_vii:n
```

```
  {
    \bool_if:NT \l__enumext_resume_count_vii_bool
      {
        \__enumext_resume_last_counter:
      }
    \tl_if_novalue:nF {#1}
      {
        \str_clear:N \l__enumext_series_name_str
        \keys_set:nn { enumext / enumext* } {#1}
        \bool_if:NF \l__enumext_print_keyans_cmd_bool
          {
            \__enumext_parse_series:n {#1}
          }
        \__enumext_store_active_keys_vii:n {#1}
      }
  }
```

(*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*`.

```
\cs_new_protected:Nn \__enumext_before_list_vii:
  {
    \__enumext_vspace_above_vii:
    \__enumext_check_ans_active:
    \__enumext_before_args_exec_vii:
    \__enumext_start_mini_vii:
  }
```

(*End of definition for* `\__enumext_before_list_vii:`.)

`\__enumext_after_list_vii:`

The function `\__enumext_after_list_vii:` first calls `\__enumext_stop_mini_vii:`, which internally calls `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` (§13.48.5) used by the keys `mini-env`, `mini-right` and `mini-right*`. Then it calls `\__enumext_after_stop_list_vii:`, used by the key `after`. After that it calls `\__enumext_check_ans_key_hook:`, used by the key `check-ans`, and `\__enumext_vspace_below_vii:`, used by the keys `below` and `below*`. Finally it sets `\l__enumext_starred_bool` to false and calls `\__enumext_starred_save_counter:`, used by the `series`, `resume` and `resume*` keys.

```
\cs_new_protected:Nn \__enumext_after_list_vii:
  {
    \__enumext_stop_mini_vii:
    \__enumext_after_stop_list_vii:
    \__enumext_check_ans_key_hook:
    \__enumext_vspace_below_vii:
    \bool_set_false:N \l__enumext_starred_bool
    \bool_if:NF \l__enumext_print_keyans_cmd_bool
      {
        \__enumext_starred_save_counter:
      }
  }
```

(*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`.

```
\cs_new_protected:Nn \__enumext_start_store_level_vii:
  {
    \bool_if:NT \l__enumext_store_active_bool
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \__enumext_store_level_open_vii:
          }
      }
  }
\cs_new_protected:Nn \__enumext_stop_store_level_vii:
```

```
  {
    \bool_if:NT \l__enumext_store_active_bool
      {
        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
          {
            \__enumext_store_level_close_vii:
          }
      }
  }
```

(*End of definition for* `\__enumext_start_store_level_vii:` *and* `\__enumext_stop_store_level_vii:`.)

13.49.1 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.

```
\cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
  {
    \skip_horizontal:n
      {
        -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
      }
    \ignorespaces
  }
```

(*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`
`\__enumext_starred_item_vii_aux_i:w`
`\__enumext_starred_item_vii_aux_ii:w`
`\__enumext_starred_item_vii_aux_iii: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`.

```
\cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
  {
    \__enumext_stop_item_tmp_vii:
    \int_incr:N \l__enumext_item_column_pos_vii_int
    \int_gincr:N \g__enumext_item_count_all_vii_int
    \__enumext_item_peek_args_vii:
  }
```

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)`.

```
\cs_new_protected:Nn \__enumext_item_peek_args_vii:
  {
    \peek_meaning:NTF (
      { \__enumext_joined_item_vii:w }
      { \__enumext_joined_item_vii:w (1) }
  }
```

The function `\__enumext_joined_item_vii:w` first sets the *width* of the box that stores the content passed to `\item` by calling `\__enumext_starred_joined_item_vii:n`. Then it looks for the argument '`*`'; if it is present, it calls `\__enumext_starred_item_vii:w`, otherwise `\__enumext_standard_item_vii:w`.

```
\cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
  {
    \__enumext_starred_joined_item_vii:n {#1}
    \peek_meaning_remove:NTF *
      { \__enumext_starred_item_vii:w  }
      { \__enumext_standard_item_vii:w }
  }
```

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].

```
\cs_new_protected:Npn \__enumext_standard_item_vii:w
  {
    \bool_set_false:N \l__enumext_item_starred_vii_bool
    \peek_meaning:NTF [
      {
        \bool_set_eq:NN
          \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
        \__enumext_start_item_vii:w
      }
      {
        \bool_set_true:N \l__enumext_wrap_label_vii_bool
        \legacy_if_set_true:n { @noitemarg }
        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
      }
  }
```

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*⟩].

```
\cs_new_protected:Npn \__enumext_starred_item_vii:w
  {
    \bool_set_true:N \l__enumext_item_starred_vii_bool
    \bool_set_true:N \l__enumext_wrap_label_vii_bool
    \peek_meaning:NTF [
      { \__enumext_starred_item_vii_aux_i:w }
      { \__enumext_starred_item_vii_aux_ii:w }
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
  {
    \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
    \__enumext_starred_item_vii_aux_ii:w
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
  {
    \peek_meaning:NTF [
      { \__enumext_starred_item_vii_aux_iii:w }
      {
        \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
        \legacy_if_set_true:n { @noitemarg }
        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
      }
  }
\cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
  {
    \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
    \legacy_if_set_true:n { @noitemarg }
    \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
  }
```

(*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 lualatex

```
\cs_new_protected_nopar:Nn \__enumext_fake_make_label_vii:n
  {
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
```

```
      \legacy_if:nT { @nmbrlist }
        {
          \IfDocumentMetadataT
            {
              \bool_if:NT \l__enumext_hyperref_bool
                {
                  \legacy_if_set_true:n { @hyper@item }
                }
            }
          \refstepcounter{enumXvii}
          \bool_if:NT \l__enumext_check_answers_bool
            {
              \int_gincr:N \g__enumext_item_number_int
              \bool_set_true:N \l__enumext_item_number_bool
            }
        }
    }
  \bool_if:NT \l__enumext_item_starred_vii_bool
    {
      \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
        {
          \tl_gset_eq:NN
            \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
        }
      \mode_leave_vertical:
      \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
      \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
      \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
      \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
    }
  \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
    {
      \tl_use:N \l__enumext_label_font_style_vii_tl
      \bool_if:NTF \l__enumext_wrap_label_vii_bool
        {
          \__enumext_wrapper_label_vii:n {#1}
        }
        {#1}
    }
  \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
}
```

(*End of definition for* `\__enumext_fake_make_label_vii:n`.)

13.49.2 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.

©2024–2026 by Pablo González L

\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.

```
\int_new:N  \l__enumext_base_depth_int
\bool_new:N \l__enumext_trivlist_bool
\dim_new:N  \l__enumext_outer_labelwidth_dim
\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.

```
\cs_new_protected_nopar:Npn \__enumext_item_switch:N #1
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn { \@listdepth } > { \l__enumext_base_depth_int } }
      { \l__enumext_trivlist_bool }
      { \__enumext_item_std:w }
      { #1 }
  }
```

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.

```
\cs_new_protected_nopar:Nn \__enumext_nested_save:
  {
    \dim_set_eq:NN \l__enumext_outer_labelwidth_dim \labelwidth
    \dim_set_eq:NN \l__enumext_outer_labelsep_dim \labelsep
  }
\cs_new_protected_nopar:Nn \__enumext_nested_reset:
  {
    \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
    \bool_set_false:N \l__enumext_trivlist_bool
    \dim_set_eq:NN \labelwidth \l__enumext_outer_labelwidth_dim
    \dim_set_eq:NN \labelsep \l__enumext_outer_labelsep_dim
  }
```

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`.

```
\__enumext_at_begin_document:n
  {
    \cs_new_eq:NN \__enumext_trivlist_orig: \trivlist
    \cs_set_protected_nopar:Npn \trivlist
      {
        \__enumext_trivlist_orig:
        \bool_set_true:N \l__enumext_trivlist_bool
      }
  }
```

Finally the definition of \item in enumext* and keyans* respectively.

```
\cs_new_protected_nopar:Nn \__enumext_item_switch_vii:
  { \__enumext_item_switch:N \__enumext_start_item_tmp_vii: }
\cs_new_protected_nopar:Nn \__enumext_item_switch_viii:
  { \__enumext_item_switch:N \__enumext_start_item_tmp_viii: }
```

(*End of definition for* `\l__enumext_base_depth_int` *and others.*)

13.49.3 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
__enumext_start_item_box_vii:w
__enumext_stop_item_vii:

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 (§13.49.4), otherwise __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.

```
\cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
  {
    \bool_lazy_and:nnTF
      { \l__enumext_item_break_vii_bool }
      {
        \int_compare_p:nNn
          { \l__enumext_joined_item_vii_int } = { \l__enumext_columns_vii_int }
      }
      { \__enumext_start_item_break_vii:n {#1} }
      { \__enumext_start_item_box_vii:w [#1] }
  }
\cs_new_protected_nopar:Npn \__enumext_start_item_box_vii:w [#1]
  {
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
    \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
      {
        \l__enumext_joined_width_vii_dim
        + \l__enumext_labelwidth_vii_dim
        + \l__enumext_labelsep_vii_dim
      }
```

Redefine the \footnote command.

```
      \__enumext_renew_footnote_starred:
```

Now the *sockets* for *tagging* PDF support are inserted and \item is run.

```
      \__enumext_start_list_tag:n {enumext*}
      \__enumext_fake_make_label_vii:n {#1}
      \__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 (§13.49.2) 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.

```
      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
        \__enumext_nested_reset:
        \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
        \__enumext_unskip_unkern:
        \__enumext_unskip_unkern:
        \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
        \tl_use:N \l__enumext_fake_item_indent_vii_tl
        \tl_use:N \l__enumext_after_list_args_vii_tl
  }
```

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*.

```
\cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
  {
      \__enumext_endminipage:
    \__enumext_stop_list_tag:n {enumext*}
    \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*.

```
    \int_set:Nn \hbadness { 10000 }
    \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.

```
    \int_compare:nNnTF
```

©2024–2026 by Pablo González L

```
      { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
      {
        \par\noindent
        \int_zero:N \l__enumext_item_column_pos_vii_int
      }
      {
        \skip_horizontal:N \l__enumext_columns_sep_vii_dim
      }
  }
```

(*End of definition for* `\__enumext_start_item_vii:w`, `\__enumext_start_item_box_vii:w`, *and* `\__enumext_stop_item_vii:`.)

13.49.4 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 LaTeX 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.

```
\cs_new_protected_nopar:Nn \__enumext_start_item_break_vii:n
  {
    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_break_vii:
    \group_begin:
      \dim_add:Nn \@totalleftmargin
        { \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim }
      \dim_sub:Nn \linewidth
        { \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim }
      \parshape \c_one_int \@totalleftmargin \linewidth
      \skip_horizontal:n
        { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
      \__enumext_start_list_tag:n {enumext*}
      \__enumext_fake_make_label_vii:n {#1}
      \__enumext_nested_reset:
      \__enumext_stop_start_break_tag:
      \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
      \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
      \tl_use:N \l__enumext_fake_item_indent_vii_tl
      \tl_use:N \l__enumext_after_list_args_vii_tl
  }
\cs_new_protected_nopar:Nn \__enumext_stop_item_break_vii:
  {
    \__enumext_stop_item_break_tag:n {enumext*}
    \par
    \group_end:
    \int_zero:N \l__enumext_item_column_pos_vii_int
    \noindent
  }
```

(*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 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.

```
\cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
```

```
  {
    \int_compare:nNnT
      {
        \int_mod:nn
          {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
      }
      =
      { 0 }
      {
        \para_end:
        \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
        \skip_vertical:N \c_zero_skip
        \int_gzero:N \g__enumext_item_count_all_vii_int
      }
  }
```

(*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`.

```
\__enumext_after_env:nn {enumext*}
  {
    \__enumext_execute_after_env:
  }
```

13.50 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.

```
\NewDocumentEnvironment{keyans*}{ o }
  {
    \__enumext_safe_exec_viii:
    \__enumext_parse_keys_viii:n {#1}
    \__enumext_before_list_viii:
    \__enumext_nested_save:
    \__enumext_start_list:nn { }
      {
        \__enumext_list_arg_two_viii:
        \__enumext_before_keys_exec_viii:
      }
    \legacy_if_set_true:n { @nmbrlist }
    \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
    \__enumext_starred_columns_set_viii:
    \item[] \scan_stop:
    % Set start counter
    \setcounter { enumXviii }
      { \int_eval:n { \int_use:c { l__enumext_start_viii_int } - 1 } }
    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
    \cs_set_eq:NN \item \__enumext_item_switch_viii:
    \int_set:Nn \l__enumext_base_depth_int { \@listdepth }
    \bool_set_false:N \l__enumext_trivlist_bool
    \ignorespaces
  }
  {
    % {tag=text-unit}
    \IfDocumentMetadataT { \tag_struct_end:n {tag=semantic-para} }
    \__enumext_stop_item_tmp_viii:
    \__enumext_remove_extra_parsep_viii:
    \__enumext_check_starred_cmd:n { item }
    \__enumext_after_list_viii:
  }
```

(*End of definition for* `keyans*`. *This function is documented on page 18.*)

`\__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.

```
\cs_new_protected:Nn \__enumext_safe_exec_viii:
  {
    \bool_lazy_or:nnF { \l__enumext_store_active_bool } { \__enumext_in_keyans_p: }
      {
        \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
      }
    \int_incr:N \l__enumext_keyans_level_h_int
    \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
      {
        \msg_error:nn { enumext } { nested }
      }
    \__enumext_keyans_name_and_start:
    \bool_if:NT \l__enumext_starred_bool
      {
        \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
      }
    \bool_set_true:N \l__enumext_starred_bool
    % Set false for interfering with enumext nested in keyans*
    % (yes, its possible and crayze)
    \bool_set_false:N \l__enumext_store_active_bool
    \int_compare:nNnT { \l__enumext_level_int } > { 1 }
      {
        \msg_error:nn { enumext } { keyans-wrong-level }
      }
  }
```

(*End of definition for* __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n Parse [⟨*key* = *val*⟩] for keyans*.

```
\cs_new_protected:Nn \__enumext_parse_keys_viii:n
  {
    \tl_if_novalue:nF {#1}
      {
        \keys_set:nn { enumext / keyans* } {#1}
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_before_list_viii:
  {
    \__enumext_vspace_above_viii:
    \__enumext_before_args_exec_viii:
    \__enumext_start_mini_viii:
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_after_list_viii:
  {
    \__enumext_stop_mini_viii:
    \__enumext_after_stop_list_viii:
    \__enumext_vspace_below_viii:
  }
```

(*End of definition for* __enumext_after_list_viii:.)

13.50.1 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.

```
\cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
  {
    \skip_horizontal:n
      {
        -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
      }
    \ignorespaces
  }
```

(*End of definition for* __enumext_first_item_tmp_viii:.)

__enumext_start_item_tmp_viii:
__enumext_item_peek_args_viii:
__enumext_joined_item_viii:w
__enumext_standard_item_viii:w

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.

```
\cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
  {
    \__enumext_stop_item_tmp_viii:
    \int_incr:N \l__enumext_item_column_pos_viii_int
    \int_gincr:N \g__enumext_item_count_all_viii_int
    \__enumext_item_peek_args_viii:
  }
```

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).

```
\cs_new_protected:Nn \__enumext_item_peek_args_viii:
  {
    \peek_meaning:NTF (
      { \__enumext_joined_item_viii:w }
      { \__enumext_joined_item_viii:w (1) }
  }
```

The function __enumext_joined_item_viii:w first sets the *width* of the box that stores the content passed to \item by calling __enumext_starred_joined_item_viii:n. Then it looks for the argument '*': if it is present, it calls __enumext_starred_item_viii:w, otherwise __enumext_standard_item_viii:w.

```
\cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
  {
    \__enumext_starred_joined_item_viii:n {#1}
    \peek_meaning_remove:NTF *
      { \__enumext_starred_item_viii:w }
      { \__enumext_standard_item_viii:w }
  }
```

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].

```
\cs_new_protected:Npn \__enumext_standard_item_viii:w
  {
    \bool_set_false:N \l__enumext_item_starred_viii_bool
    \bool_set_false:N \l__enumext_item_wrap_key_bool
```

```
    \peek_meaning:NTF [
      {
        \bool_set_eq:NN \l__enumext_wrap_label_viii_bool
          \l__enumext_wrap_label_opt_viii_bool
        \__enumext_start_item_viii:w
      }
      {
        \bool_set_true:N \l__enumext_wrap_label_viii_bool
        \legacy_if_set_true:n { @noitemarg }
        \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
      }
  }
```

(*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*⟩`]`.

```
\cs_new_protected:Npn \__enumext_starred_item_viii:w
  {
    \bool_set_true:N \l__enumext_item_starred_viii_bool
    \bool_set_true:N \l__enumext_item_wrap_key_bool
    \bool_set_true:N \l__enumext_wrap_label_viii_bool
    \peek_meaning:NTF [
      { \__enumext_starred_item_viii_aux_i:w }
      { \__enumext_starred_item_viii_aux_ii:w }
  }
```

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`.

```
\cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
  {
    \tl_clear:N \l__enumext_store_current_label_tl
    \tl_if_novalue:nF {#1}
      {
        \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
          {
            \tl_put_right:NV \l__enumext_store_current_label_tl
              \l__enumext_store_keyans_item_opt_sep_viii_tl
            \tl_put_right:Nn \l__enumext_store_current_label_tl {#1}
          }
        \tl_set:Nn \l__enumext_store_current_opt_arg_tl {#1}
      }
    \__enumext_starred_item_viii_aux_ii:w
  }
\cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
  {
    \legacy_if_set_true:n { @noitemarg }
    \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
  }
```

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.

```
\cs_new_protected:Nn \__enumext_keyans_starred_item_star:
  {
    \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
    \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
    \__enumext_keyans_store_ref:
    \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
    \__enumext_keyans_addto_seq_link:
    \int_gincr:N \g__enumext_check_starred_cmd_int
    \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
      {
        \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
      }
    \bool_if:NT \l__enumext_show_answer_bool
```

```
      {
        \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
      }
    \bool_if:NT \l__enumext_show_position_bool
      {
        \tl_set:Ne \l__enumext_mark_answer_sym_tl
          {
            \group_begin:
              \exp_not:N \normalfont
              \exp_not:N \footnotesize [ \int_eval:n
                {
                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
                }
              ]
            \group_end:
          }
        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
        \__enumext_print_keyans_box:NN
          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_keyans_wrapper_label_viii:n
  {
    \bool_lazy_all:nT
      {
        { \bool_if_p:N \l__enumext_wrap_label_viii_bool         }
        { \bool_if_p:N \l__enumext_show_answer_bool             }
        { \bool_if_p:N \l__enumext_item_wrap_key_bool           }
        { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
      }
      {
        \cs_set_eq:NN
          \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
      }
    \bool_if:NTF \l__enumext_wrap_label_viii_bool
      {
        \__enumext_wrapper_label_viii:n {#1}
      }
      {#1}
  }
\cs_new_protected_nopar:Nn \__enumext_fake_make_label_viii:n
  {
    \legacy_if:nT { @noitemarg }
      {
        \legacy_if_set_false:n { @noitemarg }
        \legacy_if:nT { @nmbrlist }
          {
            \IfDocumentMetadataT
              {
                \bool_if:NT \l__enumext_hyperref_bool
                  {
                    \legacy_if_set_true:n { @hyper@item }
                  }
              }
            \refstepcounter{enumXviii}
          }
      }
    \bool_if:NT \l__enumext_item_starred_viii_bool
      {
        \__enumext_keyans_starred_item_star:
      }
    \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
      {
        \tl_use:N \l__enumext_label_font_style_viii_tl
```

```
6241          \__enumext_keyans_wrapper_label_viii:n {#1}
6242        }
6243      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
6244    }
```

(*End of definition for* `\__enumext_keyans_wrapper_label_viii:n`, `\__enumext_fake_make_label_viii:n`, *and* `\__enumext_wrapper_label_viii:n`.)

13.50.2 Real definition of `\item` in keyans*

`\__enumext_start_item_viii:w`
`\__enumext_start_item_box_viii:w`
`\__enumext_stop_item_viii:`

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.

```
6245 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
6246   {
6247     \bool_lazy_and:nnTF
6248       { \l__enumext_item_break_viii_bool }
6249       {
6250         \int_compare_p:nNn
6251           { \l__enumext_joined_item_viii_int } = { \l__enumext_columns_viii_int }
6252       }
6253       { \__enumext_start_item_break_viii:n {#1} }
6254       { \__enumext_start_item_box_viii:w [#1] }
6255   }
6256 \cs_new_protected_nopar:Npn \__enumext_start_item_box_viii:w [#1]
6257   {
6258     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
6259     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
6260       {
6261         \l__enumext_joined_width_viii_dim
6262         + \l__enumext_labelwidth_viii_dim
6263         + \l__enumext_labelsep_viii_dim
6264       }
6265       \__enumext_renew_footnote_starred:
6266       \__enumext_start_list_tag:n {keyans*}
6267       \__enumext_fake_make_label_viii:n {#1}
6268       \__enumext_stop_start_list_tag:
6269       \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
6270         \__enumext_nested_reset:
6271         \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
6272         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
6273         \__enumext_unskip_unkern:
6274         \__enumext_unskip_unkern:
6275         \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
6276         \tl_use:N \l__enumext_fake_item_indent_viii_tl
6277         \bool_if:NT \l__enumext_item_starred_viii_bool
6278           {
6279             \__enumext_keyans_show_item_opt_viii:
6280           }
6281         \tl_use:N \l__enumext_after_list_args_viii_tl
6282   }
6283 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
6284   {
6285       \__enumext_endminipage:
6286     \__enumext_stop_list_tag:n {keyans*}
6287     \hbox_set_end:
6288     \int_set:Nn \hbadness { 10000 }
6289     \box_use_drop:N \l__enumext_item_text_viii_box
6290     \int_compare:nNnTF
6291       { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
6292       {
6293         \par\noindent
6294         \int_zero:N \l__enumext_item_column_pos_viii_int
6295       }
6296       {
6297         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
6298       }
6299   }
```

(*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:

For the keyans* environment the implementation is the same as __enumext_start_item_break_vii:n and __enumext_stop_item_break_vii:. The item option shown by \item* is placed at the beginning of the ⟨*content*⟩.

```
\cs_new_protected_nopar:Nn \__enumext_start_item_break_viii:n
  {
    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_break_viii:
    \group_begin:
      \dim_add:Nn \@totalleftmargin
        { \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim }
      \dim_sub:Nn \linewidth
        { \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim }
      \parshape \c_one_int \@totalleftmargin \linewidth
      \skip_horizontal:n
        { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
      \__enumext_start_list_tag:n {keyans*}
      \__enumext_fake_make_label_viii:n {#1}
      \__enumext_nested_reset:
      \__enumext_stop_start_break_tag:
      \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
      \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
      \tl_use:N \l__enumext_fake_item_indent_viii_tl
      \bool_if:NT \l__enumext_item_starred_viii_bool
        { \__enumext_keyans_show_item_opt_viii: }
      \tl_use:N \l__enumext_after_list_args_viii_tl
  }
\cs_new_protected_nopar:Nn \__enumext_stop_item_break_viii:
  {
    \__enumext_stop_item_break_tag:n {keyans*}
    \par
    \group_end:
    \int_zero:N \l__enumext_item_column_pos_viii_int
    \noindent
  }
```

(*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.

```
\cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
  {
    \int_compare:nNnT
      {
        \int_mod:nn
          { \g__enumext_item_count_all_viii_int }
          { \l__enumext_columns_viii_int }
      }
      =
      { 0 }
      {
        \para_end:
        \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
        \skip_vertical:N \c_zero_skip
        \int_gzero:N \g__enumext_item_count_all_viii_int
      }
  }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

13.51 The command \getkeyans

13.51.1 Definition of error messages for \getkeyans

missing-colon

Message used by \getkeyans command.

```
\msg_new:nnnn { enumext } { missing-colon }
  {
    The~argument~'#1'~has~no~colon~\msg_line_context:.
  }
  {
    The~argument~of~'\token_to_str:N \getkeyans'~must~have~the~form~
    'store~name:position'.
  }
```

(*End of definition for* missing-colon.)

©2024–2026 by Pablo González L

13.51.2 Internal functions for \getkeyans

__enumext_getkeyans_aux:n
__enumext_getkeyans_aux:e
__enumext_getkeyans:nn

The internal function __enumext_getkeyans_aux:n is in charge of *splitting* the *mandatory argument* using ":". If ":" is omitted it will return an error.

```
\cs_new_protected:Nn \__enumext_getkeyans_aux:n
  {
    \str_if_in:nnTF {#1} { : }
      {
        \use:e
          {
            \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
              { {##1} {##2} }
          }
        \exp_after:w \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
      }
      { \msg_error:nnn { enumext } { missing-colon } {#1} }
  }
\cs_generate_variant:Nn \__enumext_getkeyans_aux:n { e }
```

The internal function __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*.

```
\cs_new_protected:Nn \__enumext_getkeyans:nn
  {
    \prop_if_exist:cTF { g__enumext_ #1 _prop }
      {
        \prop_item:cn { g__enumext_ #1 _prop }{#2}
      }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
      }
  }
```

(*End of definition for* __enumext_getkeyans_aux:n *and* __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.

```
\NewDocumentCommand \getkeyans { m }
  {
    \__enumext_getkeyans_aux:e
      { \tl_to_str:e { \text_expand:n {#1} } }
  }
```

(*End of definition for* \getkeyans. *This function is documented on page* 21.)

13.52 The command \printkeyans

The \printkeyans command prints *"all stored content"* in the *sequence* defined by the save-ans key.

13.52.1 Definition of variables for \printkeyans

\c__enumext_print_envs_clist
\c__enumext_default_vars_clist
\c__enumext_default_label_X_tl

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 ⟨*name*⟩⟨*type*⟩, 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 ⟨*label*⟩ is saved by the label key in \c__enumext_default_label_X_tl.

```
\clist_const:Nn \c__enumext_print_envs_clist
  {
    {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
  }
\clist_const:Nn \c__enumext_default_vars_clist
  {
    {labelsep}{dim}, {labelwidth}{dim}, {fake_item_indent}{dim},
    {listoffset}{dim}, {rightmargin}{dim}, {listparindent}{dim},
    {leftmargin_tmp}{dim}, {leftmargin_tmp}{bool}, {columns_sep}{dim},
```

```
    {minipage_hsep}{dim}, {topsep}{skip}, {partopsep}{skip}, {parsep}{skip},
    {itemsep}{skip}, {columns}{int}, {start}{int}, {beginparpenalty}{int},
    {itempenalty}{int}, {endparpenalty}{int}, {label_fill_left}{tl},
    {label_fill_right}{tl}, {align_label_pos}{str}, {label_font_style}{tl},
    {after_list_args}{tl}, {wrap_label_opt}{bool}, {item_break}{bool}
  }
```

(*End of definition for* `\c__enumext_print_envs_clist`, `\c__enumext_default_vars_clist`, *and* `\c__enumext_default_label_X_tl`.)

`\l__enumext_print_keyans_starred_tl` The token list `\l__enumext_print_keyans_starred_tl` is used by the command `\printkeyans` (§13.52).

```
\tl_new:N \l__enumext_print_keyans_starred_tl
```

(*End of definition for* `\l__enumext_print_keyans_starred_tl`.)

13.52.2 Definition of error messages for `\printkeyans`

`print-starred` Messages used by `\printkeyans*` command.

```
\msg_new:nnnn { enumext } { print-starred }
  {
    '\token_to_str:N \printkeyans*':~The~sequence~'#1'~already~contains~
    #2~environment~\msg_line_context:.
  }
  {
    '\token_to_str:N \printkeyans*'~opens~'enumext*';~use~
    '\token_to_str:N \printkeyans'~without~the~star.
  }
```

(*End of definition for* `print-starred`.)

13.52.3 Definition of keys for `\printkeyans`

`print*` `print-*` `print-1` `print-2` `print-3` `print-4` First a set of ⟨*filtered keys*⟩ 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`.

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 ⟨*keys*⟩ are run (§13.52).

The variable `\l__enumext_print_keyans_starred_tl` will have the default ⟨*keys*⟩ for `\printkeyans*` and will be set by `\setenumext[`⟨*print**⟩`]` 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[`⟨*print* ,*⟩`]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[`⟨*print* , *level*⟩`]`.

```
\keys_define:nn  { enumext / print }
  {
    print*  .code:n =
      {
        \keys_precompile:neN { enumext / enumext* }
          { \__enumext_filter_save_key:n {#1} }
          \l__enumext_print_keyans_starred_tl
      }, % starred cmd
    print*  .initial:n =
      {
        nosep,
        columns = 2,
        first   = \small,
        font    = \small
      },
    print-1 .code:n =
      {
        \keys_precompile:neN { enumext / level-1 }
          { \__enumext_filter_save_key:n {#1} }
          \l__enumext_print_keyans_i_tl
      },
    print-1 .initial:n =
      {
        nosep,
        columns = 2,
        first   = \small,
```

```
6435          font    = \small
6436        },
6437      print-2 .code:n =
6438        {
6439          \keys_precompile:neN { enumext / level-2 }
6440            { \__enumext_filter_save_key:n {#1} }
6441            \l__enumext_print_keyans_ii_tl
6442        },
6443      print-2 .initial:n =
6444        {
6445          nosep,
6446          first = \small,
6447          font  = \small
6448        },
6449      print-3 .code:n =
6450        {
6451          \keys_precompile:neN { enumext / level-3 }
6452            { \__enumext_filter_save_key:n {#1} }
6453            \l__enumext_print_keyans_iii_tl
6454        },
6455      print-3 .initial:n =
6456        {
6457          nosep,
6458          first = \small,
6459          font  = \small
6460        },
6461      print-4 .code:n =
6462        {
6463          \keys_precompile:neN { enumext / level-4 }
6464            { \__enumext_filter_save_key:n {#1} }
6465            \l__enumext_print_keyans_iv_tl
6466        },
6467      print-4 .initial:n =
6468        {
6469          nosep,
6470          first = \small,
6471          font  = \small
6472        },
6473      print-* .code:n =
6474        {
6475          \keys_precompile:neN { enumext / enumext* }
6476            { \__enumext_filter_save_key:n {#1} }
6477            \l__enumext_print_keyans_vii_tl
6478        }, % starred nested
6479      print-* .initial:n =
6480        {
6481          nosep,
6482          first = \small,
6483          font  = \small
6484        },
6485    }
```

(*End of definition for* `print*` *and others.*)

13.52.4 Internal functions for `\printkeyans`

`\__enumext_default_save:nnn`
`\__enumext_default_restore:nnn`
`\__enumext_default_save_env:nn`
`\__enumext_default_restore_env:nn`
`\__enumext_default_restore:`

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.

```
6486 \cs_new_protected:Nn \__enumext_default_save:nnn
6487   {
6488     \cs_if_exist:cT { l__enumext_ #1 _ #3 _ #2 }
6489       {
6490         \use:c { #2 _new:c } { g__enumext_default_ #1 _ #3 _ #2 }
6491         \use:c { #2 _gset_eq:cc }
6492           { g__enumext_default_ #1 _ #3 _ #2 } { l__enumext_ #1 _ #3 _ #2 }
6493       }
6494   }
6495 \cs_new_protected:Nn \__enumext_default_restore:nnn
6496   {
6497     \cs_if_exist:cT { g__enumext_default_ #1 _ #3 _ #2 }
```

```
      {
        \use:c { #2 _set_eq:cc }
          { l__enumext_ #1 _ #3 _ #2 } { g__enumext_default_ #1 _ #3 _ #2 }
      }
  }
```

The function `\__enumext_default_save_env:nn` saves, for one environment, all the variables of `\c__enumext_default_vars_clist` 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.

```
\cs_new_protected:Nn \__enumext_default_save_env:nn
  {
    \clist_map_inline:Nn \c__enumext_default_vars_clist
      { \__enumext_default_save:nnn ##1 {#2} }
    \cs_gset_eq:cc { __enumext_default_wrapper_label_ #2 :n }
      { __enumext_wrapper_label_ #2 :n }
  }
\cs_new_protected:Nn \__enumext_default_restore_env:nn
  {
    \clist_map_inline:Nn \c__enumext_default_vars_clist
      { \__enumext_default_restore:nnn ##1 {#2} }
    \cs_set_eq:cc { __enumext_wrapper_label_ #2 :n }
      { __enumext_default_wrapper_label_ #2 :n }
    \keys_set:ne { enumext / #1 }
      { label = { \exp_not:v { c__enumext_default_label_ #2 _tl } } }
  }
```

Finally the function `\__enumext_default_restore:` restores, for all the environments, the values saved when the package was loaded.

```
\cs_new_protected:Nn \__enumext_default_restore:
  {
    \clist_map_inline:Nn \c__enumext_print_envs_clist
      { \__enumext_default_restore_env:nn ##1 }
  }
\clist_map_inline:Nn \c__enumext_print_envs_clist
  { \__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.

```
\cs_new_protected:Nn \__enumext_printkeyans:nnn
  {
    \seq_if_exist:cTF { g__enumext_ #3 _seq }
      {
        \seq_if_empty:cF { g__enumext_ #3 _seq }
          {
```

If the *starred argument* '`*`' is present, it checks that the environment `enumext*` is not saved in the *sequence*, then executes the variable `\l__enumext_print_keyans_starred_tl` that contains the default ⟨*keys*⟩ for the environment `enumext*`, sets `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, opens the `enumext*` environment passing the *optional argument* and maps the *sequence*, then sets `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```
            \bool_if:nTF {#1}
              {
                \seq_if_in:cnTF { g__enumext_ #3 _seq } { \end{enumext*} }
                  {
                    \msg_error:nnnn { enumext } { print-starred } {#3}
                      { enumext* }
                  }
                  {
                    \tl_use:N \l__enumext_print_keyans_starred_tl
                    \bool_set_true:N \l__enumext_base_line_fix_bool
                    \bool_set_true:N \l__enumext_print_keyans_star_bool
                    \begin{enumext*}[#2]
                      \seq_map_inline:cn { g__enumext_ #3 _seq } {##1}
                    \end{enumext*}
                    \bool_set_false:N \l__enumext_base_line_fix_bool
```

```
                    \bool_set_false:N \l__enumext_print_keyans_star_bool
                  }
              }
```

Otherwise it will open the environment `enumext` passing the *optional argument* to the *"first level"* then map the *sequence*.

```
              {
                \begin{enumext}[#2]
                  \seq_map_inline:cn { g__enumext_ #3 _seq } {##1}
                \end{enumext}
              }
          }
      }
      {
        \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
      }
  }
```

(*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.

```
\NewDocumentCommand \printkeyans { s O{} m }
  {
    \group_begin:
      \__enumext_default_restore:
      \bool_set_true:N \l__enumext_print_keyans_cmd_bool
      \tl_use:N \l__enumext_print_keyans_i_tl
      \tl_use:N \l__enumext_print_keyans_ii_tl
      \tl_use:N \l__enumext_print_keyans_iii_tl
      \tl_use:N \l__enumext_print_keyans_iv_tl
      \tl_use:N \l__enumext_print_keyans_vii_tl
      \__enumext_printkeyans:nnn {#1} {#2} {#3}
      \bool_set_false:N \l__enumext_print_keyans_cmd_bool
    \group_end:
  }
```

(*End of definition for* `\printkeyans`. *This function is documented on page* 22.)

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.

13.53 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.

13.53.1 Definition of variables for `\setenumext`

`\l__enumext_setkey_tmpa_tl` `\l__enumext_setkey_tmpb_tl` `\l__enumext_setkey_tmpa_int` `\l__enumext_setkey_tmpa_seq` `\l__enumext_setkey_tmpb_seq`

Internal variables used by the command `\setenumext` (§13.53).

```
\tl_new:N  \l__enumext_setkey_tmpa_tl
\tl_new:N  \l__enumext_setkey_tmpb_tl
\int_new:N \l__enumext_setkey_tmpa_int
\seq_new:N \l__enumext_setkey_tmpa_seq
\seq_new:N \l__enumext_setkey_tmpb_seq
```

(*End of definition for* `\l__enumext_setkey_tmpa_tl` *and others.*)

13.53.2 Definition of error messages for `\setenumext`

`invalid-level` `unknown-key-family`

Messages used by the `\setenumext` command.

```
\msg_new:nnnn { enumext } { invalid-level }
  {
    The~level~'#1'~is~not~valid~\msg_line_context:.
  }
  {
```

```
    The~optional~argument~must~be~a~family~followed~by~the~levels~
    1,~2,~3,~4~or~'*',~for~example~'print,1'.
  }
\msg_new:nnnn { enumext } { unknown-key-family }
  {
    Unknown~key~family~'\l_keys_key_str'~for~enumext~\msg_line_context:.
  }
  {
    The~first~element~of~the~optional~argument~must~be~
    'enumext',~'enumext*',~'keyans',~'keyans*'~or~'print'.
  }
```

(*End of definition for* `invalid-level` *and* `unknown-key-family`.)

13.53.3 Definition of keys for \setenumext

enumext-1
enumext-2
enumext-3
enumext-4
enumext*
keyans
keyans*
print*
print-1
print-2
print-3
print-4
print-*
unknown

Now define a *"meta families"* of ⟨*keys*⟩ to access from `\setenumext`.

```
\keys_define:nn { enumext / meta-families }
  {
    enumext-1 .code:n =
      {
        \keys_set:ne { enumext / level-1 }
          {
            \__enumext_filter_level:n {#1}
          }
      },
    enumext-2 .code:n =
      {
        \keys_set:ne { enumext / level-2 }
          {
            \__enumext_filter_level:n {#1}
          }
      },
    enumext-3 .code:n =
      {
        \keys_set:ne { enumext / level-3 }
          {
            \__enumext_filter_level:n {#1}
          }
      },
    enumext-4 .code:n =
      {
        \keys_set:ne { enumext / level-4 }
          {
            \__enumext_filter_level:n {#1}
          }
      },
    enumext*  .code:n =
      {
        \keys_set:ne { enumext / enumext* }
          {
            \__enumext_filter_level:n {#1}
          }
      },
    keyans    .meta:nn = { enumext / keyans } {#1},
    keyans*   .meta:nn = { enumext / keyans* } {#1},
    print*    .meta:nn = { enumext / print } { print* = {#1} },
    print-1   .meta:nn = { enumext / print } { print-1 = {#1} },
    print-2   .meta:nn = { enumext / print } { print-2 = {#1} },
    print-3   .meta:nn = { enumext / print } { print-3 = {#1} },
    print-4   .meta:nn = { enumext / print } { print-4 = {#1} },
    print-*   .meta:nn = { enumext / print } { print-* = {#1} },
    unknown   .code:n  = { \msg_error:nn { enumext } { unknown-key-family } },
  }
```

(*End of definition for* `enumext-1` *and others.*)

13.53.4 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.

```
\cs_new:Nn \__enumext_filter_level:n
```

```
6644    {
6645      \use:e
6646        {
6647          \keyval_parse:NNn
6648            \__enumext_filter_level_key:n
6649            \__enumext_filter_level_pair:nn {#1}
6650        }
6651    }
```

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.

```
6652 \cs_new:Nn \__enumext_filter_level_key:n
6653   {
6654     \str_case:nnF {#1}
6655       {
6656         { resume* } { } { reset } { } { reset* } { }
6657       }
6658       { , { \exp_not:n {#1} } }
6659   }
```

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.

```
6660 \cs_new:Nn \__enumext_filter_level_pair:nn
6661   {
6662     \str_case:nnF {#1}
6663       {
6664         { series } { }  { save-ans } { } { resume } { }
6665       }
6666       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
6667   }
```

(*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.

```
6668 \cs_new_protected:Nn \__enumext_set_parse:n
6669   {
6670     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
6671     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
6672       { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
6673     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
6674       {
6675         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
6676           { \tl_trim_spaces:n {#1} }
6677       }
6678       { \__enumext_set_error:n {#1} }
6679   }
6680 \cs_new_protected:Nn \__enumext_set_error:n
6681   { \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.

```
6682 \NewDocumentCommand \setenumext { O{enumext,1} +m }
6683   {
6684     \seq_clear:N \l__enumext_setkey_tmpa_seq
6685     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
6686     \int_set:Nn \l__enumext_setkey_tmpa_int
6687       {
6688         \seq_count:N \l__enumext_setkey_tmpb_seq
6689       }
6690     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
6691       {
6692         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
6693         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
```

```
      \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
        {
          \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
        }
    }
    {
      \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
    }
  \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq
    {
      \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
    }
}
```

(*End of definition for* `\setenumext`. *This function is documented on page* 7.)

13.54 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).

13.54.1 Definition of variables for `\setenumextmeta`

`\l__enumext_meta_path_str` The string `\l__enumext_meta_path_str` is used by the `\setenumextmeta` command (§13.54).

```
\str_new:N \l__enumext_meta_path_str
```

(*End of definition for* `\l__enumext_meta_path_str`.)

13.54.2 Definition of error messages for `\setenumextmeta`

`already-defined` `prohibited-unknown` Messages used by `\setenumextmeta` command.

```
\msg_new:nnnn { enumext } { already-defined }
  {
    The~key~'#1'~is~already~defined~\msg_line_context:.
  }
  {
    Choose~another~name~for~the~meta~key.
  }
\msg_new:nnnn { enumext } { prohibited-unknown }
  {
    The~name~'unknown'~can't~be~chosen~for~a~meta~key~\msg_line_context:.
  }
  {
    'unknown'~is~reserved~by~the~package;~choose~another~name.
  }
```

(*End of definition for* `already-defined` *and* `prohibited-unknown`.)

13.54.3 Definition of keys for `\setenumextmeta`

`enumext` `enumext*` These ⟨*keys*⟩ are the selectors that can be passed in the *optional argument* of `\setenumextmeta` (§13.54) and `\setenumext` (§13.53). First, the selectors for each level, `1`, `2`, `3` and `4`, of type `.code:n` are created for *"all levels"* of the `enumext` environment.

```
\int_step_inline:nn { 4 }
  {
    \keys_define:nn { enumext }
      {
        #1 .code:n = { \str_set:Nn \l__enumext_meta_path_str { level-#1 } },
        #1 .value_forbidden:n = true,
      }
  }
```

And now the ⟨*keys*⟩ are defined for the environments using `.code:n` for the `enumext` environment and `.meta:n` for the `enumext*` environment.

```
\clist_map_inline:nn { enumext }
  {
    \keys_define:nn { enumext }
      {
        % ignored for now, might do something useful in the future
        #1  .code:n = { },
```

```
      #1  .value_forbidden:n = true,
      #1* .code:n = { \str_set:Nn \l__enumext_meta_path_str { #1* } },
      #1* .value_forbidden:n = true,
    }
  }
\keys_define:nn { enumext }
  {
    * .meta:n = enumext*,
    * .value_forbidden:n = true,
  }
```

(*End of definition for* `enumext` *and* `enumext*`.)

13.54.4 Internal functions for `\setenumextmeta`

`\__enumext_key_set_meta:nnn` `\__enumext_key_def_meta:nnn` `\__enumext_key_def_meta:Vnn`

The internal functions `\__enumext_key_set_meta:nnn` and `\__enumext_key_def_meta:nnn` will check the *optional argument* and create the *"meta-key"*.

```
\cs_new_protected:Npn \__enumext_key_set_meta:nnn #1
  {
    \keys_set:nn { enumext } {#1}
    \__enumext_key_def_meta:Vnn \l__enumext_meta_path_str
  }
\cs_new_protected:Nn \__enumext_key_def_meta:nnn
  {
    \bool_lazy_or:nnTF
      { \keys_if_exist_p:nn { enumext / #1} {#2} }
      { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
      { \msg_error:nnn { enumext } { already-defined } {#2} }
      {
        \keys_define:nn { enumext / #1 }
          {
            #2 .meta:n = {#3},
            #2 .value_forbidden:n = true
          }
      }
  }
\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*`.

```
\NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
  {
    \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
      { \msg_error:nn { enumext } { prohibited-unknown } }
      {
        \bool_if:nTF {#1}
          {
            \int_step_inline:nn { 4 }
              { \__enumext_key_set_meta:nnn { enumext, ##1 } {#3} {#4} }
            \__enumext_key_set_meta:nnn { enumext* } {#3} {#4}
          }
          { \__enumext_key_set_meta:nnn {#2} {#3} {#4} }
      }
  }
```

(*End of definition for* `\setenumextmeta`. *This function is documented on page* 7.)

13.55 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.

13.55.1 Definition of variables for \foreachkeyans

\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

Internal variables used by the \foreachkeyans command (§13.55).

```
\seq_new:N \l__enumext_foreach_print_seq
\tl_new:N  \l__enumext_foreach_name_prop_tl
\tl_new:N  \l__enumext_foreach_default_keys_tl
```

(*End of definition for* `\l__enumext_foreach_print_seq`, `\l__enumext_foreach_name_prop_tl`, *and* `\l__enumext_foreach_default_keys_tl`.)

13.55.2 Definition of error messages for \foreachkeyans

for-key-unknown
for-key-value-unknown

Message used by unknown ⟨*keys*⟩ in \foreachkeyans.

```
\msg_new:nnnn { enumext } { for-key-unknown }
  {
    The~key~'#1'~is~unknown~by~'\token_to_str:N \foreachkeyans'~and~is~being~
    ignored~\msg_line_context:.
  }
  {
    The~command~'\token_to_str:N \foreachkeyans'~does~not~have~a~key~called~
    '#1'.~Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnnn { enumext } { for-key-value-unknown }
  {
    The~key~'#1=#2'~is~unknown~by~'\token_to_str:N \foreachkeyans'~and~is~
    being~ignored~\msg_line_context:.
  }
  {
    The~command~'\token_to_str:N \foreachkeyans'~does~not~have~a~key~called~
    '#1'.~Check~that~you~have~spelled~the~key~name~correctly.
  }
```

(*End of definition for* `for-key-unknown` *and* `for-key-value-unknown`.)

13.55.3 Definition of keys for \foreachkeyans

before
after
sep
start
step
stop
wrapper
unknown

A set of ⟨*keys*⟩ is defined for the command. The default values of these are saved in the variable \l__enumext_foreach_default_keys_tl to avoid the use of a group.

```
\keys_define:nn { enumext / foreach }
  {
    before  .tl_set:N = \l__enumext_foreach_before_tl,
    before  .value_required:n = true,
    after   .tl_set:N = \l__enumext_foreach_after_tl,
    after   .value_required:n = true,
    start   .int_set:N = \l__enumext_foreach_start_int,
    start   .value_required:n = true,
    stop    .int_set:N = \l__enumext_foreach_stop_int,
    stop    .value_required:n = true,
    step    .int_set:N = \l__enumext_foreach_step_int,
    step    .value_required:n = true,
    wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
    wrapper .value_required:n = true,
    sep     .tl_set:N = \l__enumext_foreach_sep_tl,
    sep     .value_required:n = true,
    unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
  }
\keys_precompile:nnN { enumext / foreach }
  {
    before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
  }
  \l__enumext_foreach_default_keys_tl
```

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

13.55.4 Internal functions for \foreachkeyans

__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:Vn
__enumext_parse_foreach_keys:n

Functions for handling unknown ⟨*keys*⟩.

```
\cs_new_protected:Nn \__enumext_parse_foreach_keys:nn
  {
    \tl_if_blank:nTF {#2}
      {
```

```
6828          \msg_error:nnn { enumext } { for-key-unknown } {#1}
6829        }
6830        {
6831          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
6832        }
6833    }
6834  \cs_generate_variant:Nn \__enumext_parse_foreach_keys:nn { V }
6835  \cs_new_protected:Nn \__enumext_parse_foreach_keys:n
6836    {
6837      \__enumext_parse_foreach_keys:Vn \l_keys_key_str {#1}
6838    }
```

(*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.

```
6839  \cs_new_protected:Nn \__enumext_foreach_keyans:nn
6840    {
6841      \tl_use:N \l__enumext_foreach_default_keys_tl
6842      \keys_set:nn { enumext / foreach } {#1}
6843      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
6844      \prop_if_exist:cF { g__enumext_ #2 _prop }
6845        {
6846          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
6847        }
6848      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
6849        {
6850          \int_set:Nn \l__enumext_foreach_stop_int
6851            { \prop_count:c { g__enumext_ #2 _prop } }
6852        }
6853      \seq_clear:N \l__enumext_foreach_print_seq
6854      \int_step_function:nnnN
6855        { \l__enumext_foreach_start_int }
6856        { \l__enumext_foreach_step_int }
6857        { \l__enumext_foreach_stop_int }
6858        \__enumext_foreach_add_body:n
6859        \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
6860    }
6861  \cs_new_protected:Nn \__enumext_foreach_add_body:n
6862    {
6863      \seq_put_right:Ne \l__enumext_foreach_print_seq
6864        {
6865          \exp_not:V \l__enumext_foreach_before_tl
6866          \__enumext_foreach_wrapper:n
6867            {
6868              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
6869            }
6870          \exp_not:V \l__enumext_foreach_after_tl
6871        }
6872    }
```

(*End of definition for* `\__enumext_foreach_keyans:nn` *and* `\__enumext_foreach_add_body:n`.)

`\foreachkeyans`

The command accepts an optional `[`⟨*key = val*⟩`]` argument in '`#1`' and a mandatory `{`⟨*store name*⟩`}` in '`#2`', and runs the loop over the content stored in it.

```
6873  \NewDocumentCommand \foreachkeyans { +O{} m }
6874    {
6875      \__enumext_foreach_keyans:nn {#1} {#2}
6876    }
```

(*End of definition for* `\foreachkeyans`. *This function is documented on page* 21.)

13.56 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.

```
6877  \msg_new:nnn { enumext } { package-load }
```

```
  {
    The~'#1'~package~is~already~loaded.
  }
\msg_new:nnn { enumext } { package-not-load }
  {
    The~'#1'~package~will~be~loaded~as~a~dependency.
  }
```

(*End of definition for* `package-load` *and* `package-not-load`.)

`unknown-choice` Message used by the keys `align` in §13.14, `mark-pos` in §13.32 and `label-pos` in §13.47.3.

```
\msg_new:nnnn { enumext } { unknown-choice }
  {
    Invalid~value~'#3'~for~key~'#1';~use~one~of:~#2~\msg_line_context:.
  }
  {
    Use~one~of~the~listed~values.
  }
```

(*End of definition for* `unknown-choice`.)

`command-wrong-place` Message used by the commands `\anskey` in §13.35 and `\anspic` in §13.47.4 when they are used in the wrong place.

```
\msg_new:nnnn { enumext } { command-wrong-place }
  {
    Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.
  }
  {
    '\c_backslash_str #1'~works~outside~the~environment~'#2'.
  }
```

(*End of definition for* `command-wrong-place`.)

`undefined-storage-anskey` Message used by the commands `\getkeyans` in §13.51, `\printkeyans` in §13.52 and `\foreachkeyans` in §13.55 when the storage is not defined.

```
\msg_new:nnnn { enumext } { undefined-storage-anskey }
  {
    Storage~named~'#1'~is~not~defined~\msg_line_context:.
  }
  {
    Store~content~first~with~the~key~'save-ans=#1'.
  }
```

(*End of definition for* `undefined-storage-anskey`.)

`missing-miniright` Message used by `enumext` in §13.44 and `keyans` in §13.45 when the key `mini-env` needs `\miniright` from §13.26.7.

```
\msg_new:nnn { enumext } { missing-miniright }
  {
    Missing~'\token_to_str:N \miniright'~\msg_line_context:.\\
    The~key~'mini-env'~needs~'\token_to_str:N \miniright'.
  }
```

(*End of definition for* `missing-miniright`.)

`nested` Message used when an environment is nested in itself, raised by `enumext*` in §13.49 and `keyans*` in §13.50.

```
\msg_new:nnnn { enumext } { nested }
  {
    The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~
    \msg_line_context:.
  }
  {
    Starred~environments~are~horizontal~lists~and~can't~be~nested.
  }
```

(*End of definition for* `nested`.)

13.57 Finish package

Finish package implementation.

```
\file_input_stop:
⟨/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 1750
above* 1750
\AddEnumextCounter *9*, *50*, 718
after 1271, 6801
align 755
already-defined 6708
\anskey *16*, *98*, *101*, 3316
anskey* *17*, 3518
anskey-cmd-key-unknown 3208
anskey-cmd-key-value-unknown 3208
anskey-empty-arg 3073
anskey-env-error 3344
anskey-env-key-unknown 3353
anskey-env-key-value-unknown 3353
anskey-env-wrong 3353
anskey-math-mode 3073
anskey-nested 3208
anskey-unnumber-item 3073
anskey-wrong-place 3208
\anspic *20*, *136*, *141*, 5309
\anspic* *90*

B

base-fix 1107
before 1271, 6801
before* 1271
beginpenalty 1049
below 1750
below* 1750
break-col 3241, 3391

C

c@ internal commands:
\c@__enumext_resume_i_int 644
\c@__enumext_resume_ii_int 644
\c@__enumext_resume_iii_int 644
\c@__enumext_resume_iv_int 644
\c@__enumext_resume_vii_int 644
check-ans 2524
Document class :
article *58*
l3doc *34*
columns 1342
columns-sep 1342
command-wrong-place 6892
Commands provided by enumext:
\AddEnumextCounter *50*
\anskey *37*
\anspic* *37*
\anspic *137*
\foreachkeyans *174*
\getkeyans *166*
\item* *37*
\item *113*
\miniright *36*
\printkeyans* *167*
\printkeyans *37*, *166*, *167*
\resetenumext *83*
\setenumextmeta *173*
\setenumext *166*, *172*
counter-not-cs 688
counter-registered 688
Counters defined by enumext:
enumXiii *48*
enumXii *48*
enumXiv *48*
enumXi *48*
enumXviii *48*
enumXvii *48*
enumXvi *48*
enumXv *48*
counters 618

E

endpenalty 1049
enumext *5*, 4610, 6722
enumext internal commands:
__enumext_add_counter:nn 706
__enumext_add_pre_parsep: 1386
__enumext_after_args_exec: 1286
__enumext_after_args_exec_v: 1302
__enumext_after_env:nn 161
__enumext_after_hyperref: 345
__enumext_after_list_vii: 5726
__enumext_after_list_viii: 6082
__enumext_after_stop_list: 1286
__enumext_after_stop_list_v: 1302
__enumext_after_stop_list_vii: 1318
__enumext_after_stop_list_viii: 1318
\l__enumext_align_label_pos_X_str 53
\l__enumext_align_label_X_str 88
\c__enumext_all_envs_clist 137
__enumext_anskey_env_file_if_-writable:n 3434
__enumext_anskey_env_file_if_-writable:nTF 3434
__enumext_anskey_env_file_write:nn . . . 3434
\c__enumext_anskey_env_hidden_space_-str 3341
__enumext_anskey_env_safe_inner: 3468
__enumext_anskey_env_safe_outer: 3468
__enumext_anskey_env_unknown:n 3419
__enumext_anskey_env_unknown:nn 3419
\l__enumext_anskey_level_int 3207
__enumext_anskey_safe_inner: 3279
__enumext_anskey_safe_outer: 3279
__enumext_anskey_show_wrap_arg:n 3159
__enumext_anskey_show_wrap_left:n . . . 3185
__enumext_anskey_unknown:n 3257
__enumext_anskey_unknown:nn 3257
\l__enumext_anspic_above_int 4929
__enumext_anspic_args:nnn 5171
\l__enumext_anspic_args_seq 4929
\l__enumext_anspic_below_int 4929
\l__enumext_anspic_body_box 4929

©2024–2026 by Pablo González L

__enumext_anspic_body_dim:n 5171
\l__enumext_anspic_body_htdp_dim 4929
__enumext_anspic_bottom_glue: 5030
__enumext_anspic_exec: 5171
\l__enumext_anspic_gap_dim 4929
__enumext_anspic_label:nn 5171
\l__enumext_anspic_label_above_bool ... 4929
\l__enumext_anspic_label_box 4929
\l__enumext_anspic_label_dp_dim 4929
\l__enumext_anspic_label_ht_dim 4929
\l__enumext_anspic_label_htdp_dim 4929
__enumext_anspic_label_pos:nnn 5171
\l__enumext_anspic_label_sep_bool 4929
\l__enumext_anspic_label_sep_skip 4929
\l__enumext_anspic_mini_pos_str 4929
\l__enumext_anspic_mini_width_dim 4929
__enumext_anspic_print:n 5171
__enumext_anspic_row:n 5171
__enumext_anspic_start_list_tag: 4879
__enumext_anspic_stop_list_tag: 4879
__enumext_anspic_stop_start_list_tag: 4879
\l__enumext_anspic_top_bool 4929
__enumext_anspic_top_glue: 5030
\l__enumext_anspic_top_skip 4929
__enumext_apply_list_class: 4283
__enumext_at_begin_document:n 157
\l__enumext_base_depth_int 5876
__enumext_before_args_exec: 1286
__enumext_before_args_exec_v: 1302
__enumext_before_args_exec_vii: 1318
__enumext_before_args_exec_viii: 1318
__enumext_before_keys_exec: 1286
__enumext_before_keys_exec_v: 1302
__enumext_before_keys_exec_vii: 1318
__enumext_before_keys_exec_viii: 1318
__enumext_before_list: 4546
__enumext_before_list_v: 4666
__enumext_before_list_vii: 5719
__enumext_before_list_viii: 6076
__enumext_calc_hspace:NNNNNNN 4240
__enumext_check_ans_active: 2544
\g__enumext_check_ans_key_bool 107
__enumext_check_ans_key_hook: 2602
__enumext_check_ans_level: 2544
__enumext_check_ans_log: 2649
__enumext_check_ans_log_msg_greater: . 2649
__enumext_check_ans_log_msg_less: ... 2649
__enumext_check_ans_log_msg_same_ok: . 2649
__enumext_check_ans_msg_greater: 2624
__enumext_check_ans_msg_less: 2624
__enumext_check_ans_msg_same_ok: 2624
__enumext_check_ans_show: 2624
\l__enumext_check_answers_bool 107
__enumext_check_starred_cmd:n 2673
\g__enumext_check_starred_cmd_int 107
\l__enumext_check_start_line_env_tl ... 107
\l__enumext_counter_i_tl 613
\l__enumext_counter_ii_tl 613
\l__enumext_counter_iii_tl 613
\l__enumext_counter_iv_tl 613
\g__enumext_counter_styles_tl 653
\l__enumext_counter_v_tl 613
\l__enumext_counter_vi_tl 613
\l__enumext_counter_vii_tl 613
\l__enumext_counter_viii_tl 613
\l__enumext_current_widest_dim 40
__enumext_default_item:n 3800
\c__enumext_default_label_X_tl 6384
__enumext_default_restore: 6486
__enumext_default_restore:nnn 6486
__enumext_default_restore_env:nn 6486
__enumext_default_save:nnn 6486
__enumext_default_save_env:nn 6486
\c__enumext_default_vars_clist 6384
__enumext_define_counter:Nn 627
__enumext_endminipage: 334
\g__enumext_envir_name_tl 30
\l__enumext_envir_name_tl 30
__enumext_execute_after_env: 3537
__enumext_fake_item_indent: 1198
__enumext_fake_item_indent_vii: 1198
__enumext_fake_item_indent_viii: 1198
\l__enumext_fake_item_indent_X_tl 47
__enumext_fake_make_label_vii:n 5830
__enumext_fake_make_label_viii:n 6198
__enumext_filter_level:n 6643
__enumext_filter_level_key:n 6643
__enumext_filter_level_pair:nn 6643
__enumext_filter_save_key:n 2866
__enumext_filter_save_key_key:n 2866
__enumext_filter_save_key_pair:nn ... 2866
__enumext_filter_series:n 2075
__enumext_filter_series_key:n 2075
__enumext_filter_series_pair:nn 2075
__enumext_first_item_tmp_vii: 5758
__enumext_first_item_tmp_viii: 6088
\g__enumext_footnote_box 393
\g__enumext_footnote_standard_arg_seq . 393
\g__enumext_footnote_standard_int 393
\g__enumext_footnote_standard_int_seq . 393
\g__enumext_footnote_starred_arg_seq .. 393
\g__enumext_footnote_starred_int 393
\g__enumext_footnote_starred_int_seq .. 393
\l__enumext_footnote_wd_dim 393
\l__enumext_footnotehyper_tag_bool ... 126
\l__enumext_footnotes_key_bool 126
__enumext_footnotetext:nn 399
__enumext_foreach_add_body:n 6839
\l__enumext_foreach_default_keys_tl ... 6780
__enumext_foreach_keyans:nn 6839
\l__enumext_foreach_name_prop_tl 6780
\l__enumext_foreach_print_seq 6780
__enumext_getkeyans:nn 6355
__enumext_getkeyans_aux:n 6355
\l__enumext_hyperref_bool 126
__enumext_if_is_int:nTF 169
__enumext_in_keyans: 3272
__enumext_internal_mini_page: 594
__enumext_is_not_nested: 207
__enumext_is_on_first_level: 207
\g__enumext_item_anskey_int 107
__enumext_item_answer_diff: 2617

©2024–2026 by Pablo González L

\g__enumext_item_answer_diff_int 107
\l__enumext_item_break_X_bool 88
\l__enumext_item_column_pos_X_int 88
\g__enumext_item_count_all_X_int 88
\l__enumext_item_number_bool 107
\g__enumext_item_number_int 107
__enumext_item_peek_args_vii: 5766
__enumext_item_peek_args_viii: 6096
__enumext_item_starred_exec: 3821
__enumext_item_starred_exec:nn 3821
\l__enumext_item_starred_X_bool 88
__enumext_item_std:w 334
__enumext_item_switch:N 5876
__enumext_item_switch_vii: 5876
__enumext_item_switch_viii: 5876
\g__enumext_item_symbol_aux_tl 3799
\g__enumext_item_symbol_aux_X_tl 88
\l__enumext_item_text_X_box 88
\l__enumext_item_width_X_dim 5328
\l__enumext_item_wrap_key_bool 107
\l__enumext_itemindent_X_dim 4198
\l__enumext_joined_item_aux_X_int 5328
__enumext_joined_item_vii:w 5766
__enumext_joined_item_viii:w 6096
\l__enumext_joined_item_X_int 5328
\l__enumext_joined_width_X_dim 88
__enumext_key_def_meta:nnn 6746
__enumext_key_set_meta:nnn 6746
__enumext_keyans_addto_prop:n 3573
__enumext_keyans_addto_seq:n 3648
__enumext_keyans_addto_seq_link: 3648
__enumext_keyans_default_item:n 4087
\l__enumext_keyans_env_bool 30
__enumext_keyans_fake_item_indent: 1198
\l__enumext_keyans_level_h_int 25
\l__enumext_keyans_level_int 25
__enumext_keyans_make_label: 4128
__enumext_keyans_make_label_box: 4128
__enumext_keyans_make_label_std: 4128
__enumext_keyans_mini_right_cmd:n 1716
__enumext_keyans_minipage_add_space: 1598
__enumext_keyans_minipage_set_skip: 1598
__enumext_keyans_multi_addvspace: 1439
__enumext_keyans_multi_set_vskip: 1439
__enumext_keyans_multicols_start: 4666
__enumext_keyans_multicols_stop: 4666
__enumext_keyans_name_and_start: 264
__enumext_keyans_parse_keys:n 4662
__enumext_keyans_pic_arg_two: 5030
\l__enumext_keyans_pic_level_int 25
__enumext_keyans_pic_parse_keys:n 5030
__enumext_keyans_pic_safe_exec: 5030
__enumext_keyans_pic_skip_abs:N 5030
__enumext_keyans_pos_mark_set: 3725
__enumext_keyans_redefine_item: 4110
__enumext_keyans_ref: 966
__enumext_keyans_ref:n 966
__enumext_keyans_safe_exec: 4641
__enumext_keyans_save_item_opt:n 3692
__enumext_keyans_set_item_width: 4752
__enumext_keyans_show_ans: 3725
__enumext_keyans_show_item_opt: 3692
__enumext_keyans_show_item_opt_viii: 3692
__enumext_keyans_show_pos: 3725
__enumext_keyans_starred_item:n 4099
__enumext_keyans_starred_item_star: 6132
__enumext_keyans_store_ref: 3594
__enumext_keyans_store_ref_aux_i: 3594
__enumext_keyans_store_ref_aux_ii: 3594
__enumext_keyans_unknown_keys:n 4009
__enumext_keyans_unknown_keys:nn 4009
__enumext_keyans_wrapper_label:n 4128
__enumext_keyans_wrapper_label_viii:n 6198
\l__enumext_label_copy_X_tl 129
\l__enumext_label_fill_left_X_tl 53
\l__enumext_label_fill_right_X_tl 53
\l__enumext_label_i_tl 834
\l__enumext_label_ii_tl 834
\l__enumext_label_iii_tl 834
\l__enumext_label_iv_tl 834
__enumext_label_style:Nnn 672
\l__enumext_label_v_tl 950
\l__enumext_label_vi_tl 950
\l__enumext_label_vii_tl 879
\l__enumext_label_viii_tl 879
\l__enumext_label_width_by_box 653
__enumext_label_width_by_box:Nn 666
\l__enumext_leftmargin_tmp_X_bool 66
\l__enumext_leftmargin_tmp_X_dim 66
\l__enumext_leftmargin_X_dim 66
__enumext_level: 165
\l__enumext_level_h_int 25
\l__enumext_level_int 25
__enumext_list_arg_two_i: 4315
__enumext_list_arg_two_ii: 4315
__enumext_list_arg_two_iii: 4315
__enumext_list_arg_two_iv: 4315
__enumext_list_arg_two_v: 4315
__enumext_list_arg_two_vii: 4363
__enumext_list_arg_two_viii: 4363
\l__enumext_list_class_tl 73
__enumext_list_numbering:n 4263
__enumext_list_numbering_check:nnn 4263
__enumext_log_answer_vars: 318
__enumext_log_global_vars: 318
__enumext_make_label: 3873
__enumext_make_label_box: 3873
__enumext_make_label_std: 3873
\l__enumext_mark_position_str 116
\l__enumext_mark_position_v_str 116
\l__enumext_mark_position_viii_str 116
\l__enumext_mark_sep_tmpa_dim 3556
\l__enumext_mark_sep_tmpb_dim 3556
\l__enumext_meta_path_str 6707
__enumext_mini_addvspace_vii: 1641
__enumext_mini_addvspace_viii: 1641
__enumext_mini_apply_end: 1524
__enumext_mini_baselineskip: 1510
__enumext_mini_capture_end: 1524
__enumext_mini_capture_multicols: 1524
\g__enumext_mini_dp_dim 1471
\g__enumext_mini_end_bool 1471

©2024–2026 by Pablo González L

\g__enumext_mini_end_skip 1471
\l__enumext_mini_gap_dim 1471
\l__enumext_mini_last_box 1471
__enumext_mini_page 594
\l__enumext_mini_prev_dim 1471
__enumext_mini_right_cmd:n 1684
__enumext_mini_set_vskip_vii: 1633
__enumext_mini_set_vskip_viii: 1633
__enumext_minipage:w 334
\g__enumext_minipage_active_X_bool . . . 5328
\l__enumext_minipage_active_X_bool 80
__enumext_minipage_add_space: 1560
\g__enumext_minipage_center_X_bool 88
\l__enumext_minipage_left_X_dim 80
\g__enumext_minipage_right_skip 80
\l__enumext_minipage_right_skip 1471
\g__enumext_minipage_right_X_dim 88
__enumext_minipage_set_skip: 1560
\g__enumext_minipage_stat_int 80
\l__enumext_miniright_code_X_box 5328
__enumext_multi_addvspace: 1422
__enumext_multi_set_vskip: 1374
\l__enumext_multicols_above_X_skip 74
\l__enumext_multicols_below_X_skip 74
__enumext_multicols_start: 4490
__enumext_multicols_stop: 4521
__enumext_nested_base_line_fix: 1113
__enumext_nested_reset: 5876
__enumext_nested_save: 5876
__enumext_newlabel:nn 377
\l__enumext_newlabel_arg_one_tl 129
\l__enumext_newlabel_arg_two_tl 129
\l__enumext_outer_labelsep_dim 5876
\l__enumext_outer_labelwidth_dim 5876
__enumext_parse_foreach_keys:n 6824
__enumext_parse_foreach_keys:nn 6824
__enumext_parse_keys:n 4429
__enumext_parse_keys_vii:n 5702
__enumext_parse_keys_viii:n 6069
__enumext_parse_save_key:n 2853
__enumext_parse_save_key_vii:n 2853
__enumext_parse_series:n 2136
\c__enumext_print_envs_clist 6384
__enumext_print_footnote: 399
__enumext_print_footnote_mini: 399
__enumext_print_footnote_mini_tagged: 399
__enumext_print_footnote_standard: . . . 516
__enumext_print_footnote_starred: . . . 516
__enumext_print_keyans_box:NN 2983
\l__enumext_print_keyans_cmd_bool 116
\l__enumext_print_keyans_star_bool . . . 116
\l__enumext_print_keyans_starred_tl . . . 6399
\l__enumext_print_keyans_X_tl 116
__enumext_printkeyans:nnn 6526
__enumext_redefine_item: 3862
\l__enumext_ref_key_arg_tl 818
\l__enumext_ref_the_count_tl 818
__enumext_register_default_label_wd:Nn 656
__enumext_remove_extra_parsep_vii: . . . 5992
__enumext_remove_extra_parsep_viii: . . 6330
\l__enumext_renew_counter_X_tl 818
__enumext_renew_footnote: 399
__enumext_renew_footnote_mini: 399
__enumext_renew_footnote_mini_tagged: 399
__enumext_renew_footnote_standard: . . . 516
__enumext_renew_footnote_starred: . . . 516
__enumext_reset_count_resume:nn 2311
__enumext_reset_count_resume_all:n . . . 2311
__enumext_reset_count_resume_levels:n 2311
__enumext_reset_global_bool: 294
__enumext_reset_global_int: 294
__enumext_reset_global_tl: 294
__enumext_reset_global_vars: 294
__enumext_resume:n 2180
\l__enumext_resume_count_bool 41
\l__enumext_resume_count_X_bool 41
__enumext_resume_counter: 2026
__enumext_resume_integer_series: 2208
__enumext_resume_last_counter: 2102
\g__enumext_resume_last_keys_X_tl 1869
__enumext_resume_series:n 2208
\l__enumext_resume_series_X_bool 1869
__enumext_resume_star: 2273
\l__enumext_resume_star_key_X_bool . . . 1869
__enumext_safe_exec: 4418
__enumext_safe_exec_vii: 5685
__enumext_safe_exec_viii: 6044
__enumext_save_last_keys:n 2102
__enumext_second_part: 4570
__enumext_second_part_v: 4666
\l__enumext_series_name_tl 1869
__enumext_set_error:n 6668
__enumext_set_item_width: 4596
__enumext_set_parse:n 6668
\l__enumext_setkey_tmpa_int 6575
\l__enumext_setkey_tmpa_seq 6575
\l__enumext_setkey_tmpa_tl 6575
\l__enumext_setkey_tmpb_seq 6575
\l__enumext_setkey_tmpb_tl 6575
__enumext_show_length:nnn 175
\l__enumext_show_pos_tmp_int 3556
\g__enumext_standard_bool 30
\l__enumext_standard_bool 30
\l__enumext_standard_first_bool 30
__enumext_standard_item_vii:w 5766
__enumext_standard_item_viii:w 6096
__enumext_standard_ref: 856
__enumext_standard_ref:nn 856
__enumext_standard_reset:nn 2311
__enumext_standard_reset_key: 2384
__enumext_standard_reset_key_star: . . . 2384
__enumext_standard_save_counter: 1938
__enumext_standard_save_counter_aux: . 1938
__enumext_standard_unknown_keys:n . . . 4059
__enumext_standard_unknown_keys:nn . . . 4059
\g__enumext_starred_bool 30
\l__enumext_starred_bool 30
__enumext_starred_columns_set_vii: . . . 5355
__enumext_starred_columns_set_viii: . . 5355
\l__enumext_starred_first_bool 30
__enumext_starred_item_vii:w 5766
__enumext_starred_item_vii_aux_i:w . . . 5766

©2024–2026 by Pablo González L

__enumext_starred_item_vii_aux_ii:w .. 5766
__enumext_starred_item_vii_aux_iii:w . 5766
__enumext_starred_item_viii:w 6132
__enumext_starred_item_viii_aux_i:w .. 6132
__enumext_starred_item_viii_aux_ii:w . 6132
__enumext_starred_joined_item_vii:n .. 5417
__enumext_starred_joined_item_viii:n . 5417
__enumext_starred_ref: 899
__enumext_starred_ref:n 899
__enumext_starred_reset:n 2311
__enumext_starred_reset_key: 2384
__enumext_starred_save_counter: 1938
__enumext_starred_save_counter_aux: .. 1938
__enumext_starred_unknown_keys:n 4044
__enumext_starred_unknown_keys:nn ... 4044
__enumext_start_from:NNn 989
__enumext_start_item_box_vii:w 5913
__enumext_start_item_box_viii:w 6245
__enumext_start_item_break_vii:n 5964
__enumext_start_item_break_viii:n ... 6300
__enumext_start_item_tmp_vii: 5766
__enumext_start_item_tmp_viii: 6096
__enumext_start_item_vii:w 5913
__enumext_start_item_viii:w 6245
\g__enumext_start_line_tl 30
__enumext_start_list:nn 334
__enumext_start_list_tag:n 4782
__enumext_start_mini_vii: 5524
__enumext_start_mini_viii: 5592
__enumext_start_save_ans_msg: 2473
__enumext_start_store_level: 4454
__enumext_start_store_level_vii: 5738
\l__enumext_start_X_int 47
__enumext_stop_item_break_tag:n 4782
__enumext_stop_item_break_vii: 5964
__enumext_stop_item_break_viii: 6300
__enumext_stop_item_vii: 5913
__enumext_stop_item_viii: 6245
__enumext_stop_list: 334
__enumext_stop_list_tag:n 4782
__enumext_stop_mini_vii: 5524
__enumext_stop_mini_viii: 5592
__enumext_stop_save_ans_msg: 2473
__enumext_stop_start_break_tag: 4782
__enumext_stop_start_list_tag: 4782
__enumext_stop_store_level: 4483
__enumext_stop_store_level_vii: 5738
\l__enumext_store_active_bool 102
__enumext_store_active_keys:n 2820
__enumext_store_active_keys_vii:n ... 2820
__enumext_store_addto_prop:n 2899
__enumext_store_addto_seq:n 2912
__enumext_store_anskey_arg:n 3094
\l__enumext_store_anskey_arg_tl 3072
__enumext_store_anskey_env:n 3468
\l__enumext_store_anskey_env_tl 3337
\l__enumext_store_current_label_tl ... 102
\l__enumext_store_current_opt_arg_tl .. 102
__enumext_store_internal_ref: 2997
__enumext_store_level_close: 2920
__enumext_store_level_close_vii: 2952
__enumext_store_level_open: 2920
__enumext_store_level_open_vii: 2952
\g__enumext_store_name_tl 102
\l__enumext_store_name_tl 102
\l__enumext_store_save_key_X_bool 2690
\l__enumext_store_save_key_X_tl 2690
\l__enumext_store_upper_level_X_bool .. 4406
__enumext_storing_exec: 2483
__enumext_storing_set:n 2483
\l__enumext_the_counter_X_tl 818
\l__enumext_tmpa_X_int 5328
\l__enumext_trivlist_bool 5876
__enumext_trivlist_orig: 5876
__enumext_unnumbered_list:n 4257
__enumext_unskip_unkern: 181
\l__enumext_vspace_a_star_X_bool 1744
__enumext_vspace_above: 1773
__enumext_vspace_above_v: 1803
__enumext_vspace_above_vii: 1825
__enumext_vspace_above_viii: 1825
\l__enumext_vspace_b_star_X_bool 1744
__enumext_vspace_below: 1788
__enumext_vspace_below_v: 1814
__enumext_vspace_below_vii: 1847
__enumext_vspace_below_viii: 1847
__enumext_widest_from:nNNn 1004
\g__enumext_widest_label_tl 653
\l__enumext_wrap_label_opt_X_bool 53
\l__enumext_wrap_label_X_bool 53
__enumext_wrapper_label_v:n 4128
__enumext_wrapper_label_viii:n 6198
\l__enumext_write_anskey_env_bool 3337
\l__enumext_write_anskey_env_file_iow . 3337
\l__enumext_write_anskey_env_file_name_-
tl 3337
\l__enumext_write_aux_file_tl 129
enumext* 5, 5654, 6596, 6722
enumext-1 6596
enumext-2 6596
enumext-3 6596
enumext-4 6596
enumXi 627
enumXii 627
enumXiii 627
enumXiv 627
enumXv 627
enumXvi 627
enumXvii 627
enumXviii 627
Environments provided by enumext:
enumext* 34
enumext 130
keyans* 159
keyanspic 137, 140
keyans 133

F

first 1271
font 732
\footnote 44
for-key-unknown 6783
for-key-value-unknown 6783

©2024–2026 by Pablo González L

force-eol 3391
\foreachkeyans 21, 174, 6873

G

\getkeyans 21, 165, 6379

I

invalid-level 6580
\item 6, 113, 119, 153, 155, 156, 161, 164
\item* 6, 18, 90, 4110
item-answer-hook 142
item-break 1261
item-greater-answer 2449
item-join 3241, 3391
item-joined 5338
item-joined-columns 5338
item-less-answer 2449
item-pos* 3241, 3391, 3931
item-star 3241, 3391
item-sym* 3241, 3391, 3931
\itemindent 122
itemindent 1152
items-same-answer 2449
itemsep 1065

K

key-ref-empty 827
keyans 18, 4763, 6596
keyans* 18, 6012, 6596
keyans-nested 4633
keyans-unknown-key 3989
keyans-unknown-key-value 3989
keyans-wrong-level 3559
keyanspic 19, 5128
keyanspic-item-cmd 4946
keyanspic-nested 4946
keyanspic-wrong-level 4946
Keys for \anskey provided by enumext:
break-col 99, 102
force-eol 105
item-join 99, 102
item-pos* 99, 102, 114, 116
item-star 99, 102
item-sym* 99, 102, 114, 116
overwrite 105
write-env 105
Keys for \foreachkeyans provided by enumext:
after 61, 62, 130, 152, 160
before 61, 62
start 39, 56, 78, 79
Keys for anskey* provided by enumext:
break-col 99, 102
force-eol 105
item-join 99, 102
item-pos* 99, 102, 114, 116
item-star 99, 102
item-sym* 99, 102, 114, 116
overwrite 105, 106
write-env 105, 106
Keys for environments provided by enumext:
above* 58, 72–74, 129, 152
above 58, 72–74, 129, 130, 152, 160
after 61, 62, 130, 152, 160
align 51, 112, 115, 120, 154, 177
base-fix 58, 78, 94, 127
before* 61, 62, 129, 130, 152, 160
before 61, 62
below* 72–74, 130, 152
below 72–74, 130, 152, 160
check-ans 39, 40, 85, 87–90, 94, 108, 111, 130, 152
columns-sep 63, 128, 157
columns 63, 128, 156, 167
first 61, 62, 131, 133, 157, 167
font 51, 115, 120, 142, 154, 167
item-break 36, 61, 135, 156, 158
item-pos* 114, 116
item-sym* 114, 116
itemindent 59, 60, 113, 114, 119, 120, 157
itemsep 57, 125, 157, 158
label-pos 137, 139, 141, 177
label-sep 137, 139, 142, 143
labelsep 51, 123, 154
labelwidth 49–51, 53–56, 123, 154
label 49, 56, 123, 139, 166, 169
layout-sep 138, 143
layout-sty 137, 143
layout-top 138, 140
list-indent 59, 139
list-offset 59, 130, 133
listparindent 59, 125, 157
mark-ans* 91, 94, 112
mark-ans 92, 94, 100
mark-pos* 91, 94, 112
mark-pos 92, 94, 177
mark-ref 92, 94, 100
mark-sep* 91, 94, 112
mark-sep 92, 94, 100, 162
mini-env . 44, 46, 47, 63, 71, 94, 128, 129, 148, 149, 152, 160, 177
mini-right* 39, 63, 94, 149, 152
mini-right 39, 63, 70, 94, 149, 152
mini-sep 63, 94, 129
mode-box 51, 115, 120
no-store 87, 88, 94, 103, 107, 113, 114
noitemsep 57
nosep 57, 167
parsep 57, 125, 139, 157
partopsep 57
ref 52–55
reset* 78, 84, 94, 172
reset 78, 84, 94, 172
resume* . . . 48, 75, 77–79, 82, 83, 87, 94, 130, 152, 172
resume . . . 41, 48, 56, 75–83, 87, 94, 130, 151, 152, 172
rightmargin 59, 146
save-ans 41, 76–78, 81, 85, 86, 88, 89, 93–95, 102, 106, 108–110, 131, 144, 154, 159, 162, 166, 172
save-key 78, 93, 94, 127, 151
save-ref 43, 92, 94, 99, 109, 111, 119, 162
save-sep 91, 94, 109, 162
series . 75, 76, 78, 79, 81, 82, 94, 127, 130, 151, 152, 172
show-ans . . . 91, 92, 94, 96, 99, 100, 111, 112, 142, 162
show-length 39, 61, 122
show-pos 91, 92, 94, 96, 99, 100, 111, 142, 162
start* 56, 78, 79
start 39, 56, 78, 79
topsep 57, 58
widest 39, 53, 56
wrap-ans* 91, 94, 119, 120, 142
wrap-ans 92, 94, 96, 100

©2024–2026 by Pablo González L

wrap-label* 51, 113, 115, 119, 120, 153, 154, 161
wrap-label 51, 113–115, 119, 120, 139, 142, 153, 154, 161, 169
wrap-opt 91, 94, 111, 119, 142
keys commands:
\keys_precompile:nnN 153
\keys_set:nn 153

L

label 834, 879, 950
label-pos 4968
label-sep 4968
Labels provided by enumext:
\Alph* 49
\Roman* 49
\alph* 49
\arabic* 49
\roman* 49
labelsep 732
labelwidth 732
layout-sep 4968
layout-sty 4968
layout-top 4968
\leftmargin 122
list-indent 1152
list-lengths 4208
list-lengths-not-nested 4224
list-offset 1152
list-too-deep 4411
listparindent 1152

M

\makelabel 113, 119
many-miniright-used 1478
many-starred 2421
mark-ans 2737, 4968
mark-ans* 2696, 2737
mark-pos 2737, 4968
mark-pos* 2696, 2737
mark-ref 2737
mark-sep 2737, 4968
mark-sep* 2696, 2737
midpenalty 1049
mini-env 1342
mini-right 1342
mini-right* 1342
mini-sep 1342
\miniright 13, 70, 1653
missing-colon 6347
missing-miniright 6906
missing-starred 2421
mode-box 722

N

nested 6911
nested-horizontal 5338
no-store 2524
noitemsep 1065
nosep 1065
not-writing 3379

O

out-of-range 2304
overwrite 3391
overwrite-file 3379

P

package-load 6877
package-not-load 6877
Packages :
caption 149
enumitem 48
expl3 144
footnotehyper 43
hyperref 37
indent 158
latex-lab-block 42
ltcmd 42
ltsockets 134
multicol 34
scontents 104
shortlst 144
tagpdf 134
parsep 1065
partopsep 1065
print* 6409, 6596
print-* 6409, 6596
print-1 .... 6409, 6596
print-2 .... 6409, 6596
print-3 .... 6409, 6596
print-4 .... 6409, 6596
print-starred 6400
\printkeyans 22, 166, 6561
prohibited-unknown 6708
prop-seq-int-hook 142

R

ref 834, 879, 950
reset 2366
reset* 2366
\resetenumext 13, 82, 2339
resume 1896
resume* 1896
rightmargin 1152

S

save-ans 2464
save-ans-empty 2414
save-ans-log 2429
save-ans-log-hook 2429
save-key 2839
save-ref 2737
save-sep 2696, 2737, 4968
sep 6801
seq commands:
\seq_use:Nn 153
series 1896
\setenumext 7, 170, 6682
\setenumextmeta 7, 173, 6766
show-ans 2696, 2737, 4968
show-length 1251
show-pos 2696, 2737, 4968
standard-unknown-key 3967
standard-unknown-key-value 3967
starred-unknown-key 3947
starred-unknown-key-value 3947

©2024–2026 by Pablo González L

start 1020, 6801
start* 1020
start-list-tags 4782, 4879
step 6801
stop 6801
stop-list-tags 4782, 4879
stop-list-tags-break 4782
stop-list-tags-par 4782
stop-start-tags 4782, 4879
stop-start-tags-break 4782
store-int 2437
store-prop 2437
store-seq 2437
str commands:
\str_if_in:nnTF 153

T

topsep 1065

U

undefined-storage-anskey 6899
unknown . . . 3241, 3391, 4009, 4036, 4059, 4968, 6596, 6801
unknown-choice 6885
unknown-key-family 6580
unknown-series-standard 1877
unknown-series-starred 1877

W

widest 1020
width-negative 4203
wrap-ans 2737
wrap-ans* 2696, 2737, 4968
wrap-label 732
wrap-label* 732
wrap-opt 2696, 2737, 4968
wrapper 6801
write-env 3391
writing-file 3379
wrong-miniright-place 1478
wrong-miniright-starred 1478
wrong-miniright-use 1478
wrong-place 3559

©2024–2026 by Pablo González L