

enumext

ENUMERATE EXERCISE SHEETS

v1.6 2025-06-30*

©2024–2025 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	1	5.7	Keys for multicol	11
1.1	Description and usage	2	5.8	Keys for minipage	11
1.2	The concept of left margin	3	5.8.1	The command <code>\miniright</code>	12
1.3	User interface	3	5.8.2	The key <code>mini-right</code>	12
1.3.1	Public counters	3	6	The storage system	12
1.3.2	Public dimension	3	6.1	Keys for storage system	12
1.3.3	Support for <i>multicol</i>	4	6.1.1	Keys for <code>label</code> and <code>ref</code>	13
1.3.4	Support for <i>minipage</i>	4	6.1.2	Keys for wrap and marks	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.1.3	Keys for debug and checking	14
1.3.6	Support for <code>\footnote</code>	4	6.2	The command <code>\anskey</code>	14
2	The environments provided	5	6.2.1	Keys for <code>\anskey</code>	14
2.1	The environment <code>enumext</code>	5	6.3	The environment <code>anskey*</code>	15
2.2	The environment <code>enumext*</code>	5	6.3.1	Keys for <code>anskey*</code>	15
2.3	The command <code>\item*</code>	5	6.4	The environment <code>keyans</code>	16
2.3.1	Keys for <code>\item*</code>	6	6.4.1	The <code>\item*</code> in <code>keyans</code>	17
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.5	The environment <code>keyanspic</code>	17
3	The command <code>\setenumext</code>	6	6.5.1	Keys for <code>keyanspic</code>	18
4	The command <code>\setenumextmeta</code>	6	6.5.2	The command <code>\anspic</code>	18
5	The <code>keyval</code> system	7	6.6	Printing stored content	19
5.1	Keys for <code>label</code> and <code>ref</code>	7	6.6.1	The command <code>\getkeyans</code>	19
5.2	Keys for penalties	8	6.6.2	The command <code>\foreachkeyans</code>	19
5.3	Keys for spaces	8	6.6.3	The command <code>\printkeyans</code>	20
5.3.1	Vertical spaces	8	7	Full examples	21
5.3.2	Horizontal spaces	9	8	Tagged PDF examples	24
5.4	Keys for add code	10	9	The way of non-enumerated lists	24
5.5	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	10	10	References	26
5.6	Keys for <code>reset</code>	11	11	Change history	27
5.6.1	The command <code>\resetenumext</code>	11	12	Index of Documentation	28
			13	Implementation	30
			14	Index of Implementation	153

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 a note of the main ones:

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

*This file describes a documentation for v1.6, last revised 2025-06-30.

†E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), 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[3] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TTeX. It has been tested with the standard classes provided by L^AT_EX: book, report, article and letter on 10pt, 11pt and 12pt.

The minimum requirement is L^AT_EX release 2025-06-01.

1 Introduction

In the L^AT_EX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[1] and packages like xsim[2] do the job perfectly, but they don’t always fit the basic day to day needs.

In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

1. Factor $x^2 - 2x + 1$

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

4. Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

Sometimes we are also interested in showing the “answers” along with the questions:

1. Factor $x^2 - 2x + 1$

*

$(x - 1)^2$

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

*

$3(x + y + z)$

3. True False

(a) $\alpha > \delta$

*

False

(b) L^AT_EXze is cool?

*

Very True!

4. Related to Linux
- (a) You use linux?

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn’t exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete

Or we are interested in referring 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 we are interested in printing 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 in my work is “multiple choice questions”, for example:

1. First type of questions

A) value

C) value

B) correct

D) value

2. Second type of questions

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

II. $\alpha = \delta$

III. $\angle EDF = 45^\circ$

A) I only

D) I and III only

B) II only

E) I, II, and III

C) I and II only

★ 3. Third type of questions

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

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

A) value

D) value

B) value

E) value

C) value
4. Question with image and label below:

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on right side:

A) value

B) value

C) value

D) correct

E) value

B

Where what we are interested in the $\langle label \rangle$ and a “short note” that we leave as an explanation, and then print them:

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 \TeX , 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 intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], 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 $\text{MiK}\text{\TeX}$, 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`.

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

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” that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment. Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem.

The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

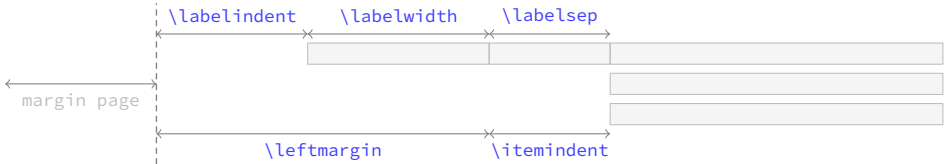


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

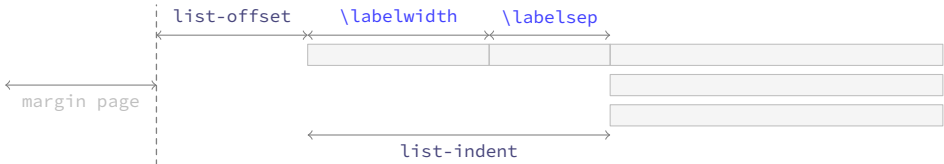


Figure 2: Representation of horizontal lengths concept in `enumext`.

In this way we reduce a *little* the amount of parameters we have to pass. 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.

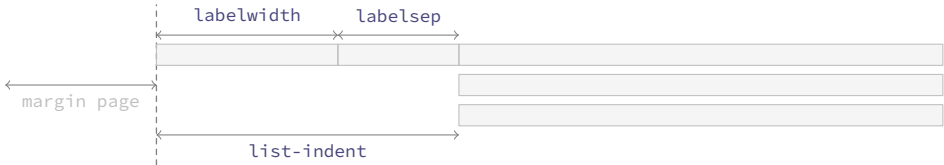


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

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` and `\foreachkeyans` to print all *stored content*, `\miniright` for `minipage`, `\setenumext` and `\setenumextmeta` to config [*key* = *val*] options.

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 *rigid length* and contains the “width of the content” of each `\item` regardless of `labelwidth` and `labelsep`.

- If any package defines `\itemwidth` or they are user-defined `\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`[3] package. This allows to obtain directly a two-column output as shown in the figure 4.

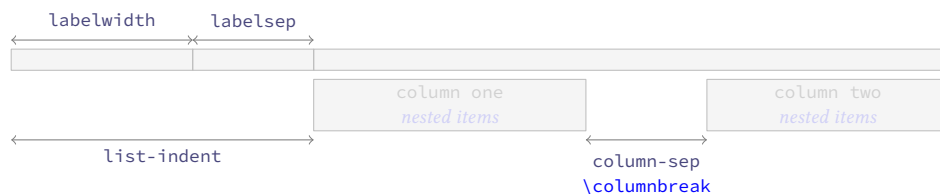


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 (see §5.7).

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

1.3.4 Support for minipage

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



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

The `minipage` environments on “left side” and “right side” is 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` (see §5.8).

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem`[6] 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` (§6.1) is active.

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, i.e. it is only used for typesetting and not directly. The `enumext` package provides an *internal implementation* for the command `\footnote` compatible with the `hyperref` package to work in the same way as if it were used anywhere in the document.

Unfortunately, if *tagging* PDF is not enabled, it will not produce the expected “links” because the internal implementation uses `\footnotetext[⟨number⟩]` and `\footnotemark[⟨number⟩]{⟨text⟩}` and support for these is limited by the `hyperref` package.

The best way to solve this if *tagged* PDF is NOT active is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the “links” if `hyperref` is loaded with the `hyperfootnotes=true` option (default). Load it is as follows:

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

At the moment the `footnotehyper` package is not compatible with *tagged* PDF.

2 The environments provided

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

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

2.1 The environment enumext

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

Example with `columns=2`

1. This text is in the first level.
- A. This text is in the fourth level.
- (a) This text is in the second level.
- X This text is in the first level.
- i. This text is in the third level.
- ★ 2. This text is in the first level.

2.2 The environment enumext*

The `enumext*` is a *horizontal list environment* similar to the `shortenumberate` or `tasks` environments provided by the `shortlst`[16] and `tasks`[17] packages, `\item` and `\item[⟨custom⟩]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[⟨key = val⟩]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “item content” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwidth`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded (see §1.3.6 for full support).
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

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⟩]` works like the numbered `\item`, but placing a `⟨symbol⟩` to the “left” of the `⟨label⟩` separated from it by the `⟨offset⟩` set by the 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*. Can be configure with the keys `item-sym*` and `item-pos*` locally in the environment or globally using `\setenumext` command (§3).

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

2.3.1 Keys for \item*

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

`item-pos*` = { $\langle rigid length \rangle$ } default: by levels
Sets the *offset* between the box containing the current $\langle label \rangle$ defined by `labelwidth` key and the $\langle symbol \rangle$ set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

2.4 The command \item in enumext*

The `\item` command for the `enumext*` environment provides an “first optional argument” `\item($\langle columns \rangle$)` which “joins items” between columns. Let’s consider the following examples adapted directly from the `task` 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 8. The eighth item is way too long for this and needs two columns (196.17749pt)
9. The ninth
- Z 10. The tenth (89.28171pt)

3 The command \setenumext

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

The command `\setenumext` sets the $\langle keys \rangle$ 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 $\langle keys \rangle$ 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`, `resume*`, `reset`, `reset*`, `series` and `save-ans` they can NOT be set by this command and are ignored. The key `save-ans` that activate the “storage system” must be passed directly in the *optional argument* of the “first level” of the environment in which they are executed.

4 The command \setenumextmeta

<code>\setenumextmeta</code>	<code>\setenumextmeta {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta* {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta [$\langle enumext* \rangle$] {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta [$\langle enumext, level \rangle$] {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the $\{ \langle key name \rangle \}$ must be different from those defined by the package. If the *optional argument* is not passed, the new “meta-key” will be created for the “first level” of the environment `enumext`.

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 we can use it like any other key so `\begin{enumext}[midsep]` and `\begin{enumext*}[midsep]` will be valid.

5 The keyval system

The $\langle key = val \rangle$ system used by the `enumext` package is implemented using `l3keys` so it must be taken into consideration that those keys marked as “*value forbidden*”, that is $\langle key \rangle$ is different from $\langle key = \rangle$.

All $\langle keys \rangle$ described in this section are available for the `enumext`, `enumext*`, `keyans` and `keyans*` environments with the exception of the keys `series`, `resume`, `resume*` which are only available for the “*first level*” of the environments `enumext` and `enumext*`; and the keys `mini-right`, `mini-right*` which are only available for the `enumext*` and `keyans*` environments.

All $\langle keys \rangle$ 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 $\langle key \rangle$ that sets a *rubber lengths* or *rigid lengths* for vertical or horizontal space on a level will influence the vertical and horizontal space for *inners levels* and `keyans`, `keyans*` and `keyanspic` environments.

5.1 Keys for label and ref

`mode-box` $\langle value forbidden \rangle$ 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` 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 `\makebox` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

`label` = { $\langle \backslash\alpha^* | \backslash\Alpha^* | \backslash\arabic^* | \backslash\roman^* | \backslash\Roman^* \rangle$ } default: *by levels*

Sets the $\langle label \rangle$ 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*` are `\arabic*`, for second level are $\langle \backslash\alpha^* \rangle$, for third level are `\roman*`, and for fourth level are `\Alpha*`. For `keyans` and `keyans*` environments the default value is `\Alpha*`.

- This key is intended to give the basic structure with which the $\langle label \rangle$ 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 $\langle label \rangle$ as an argument, for example `\emph{\langle \backslash\alpha^* \rangle}` will return an error. For full customization of how $\langle label \rangle$ is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

`labelsep` = { $\langle rigid length \rangle$ } default: `0.3333em`

Sets the *horizontal space* between the box containing the current $\langle label \rangle$ 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` = { $\langle rigid length \rangle$ } default: *by label*

Sets the *width* of the box containing the current $\langle label \rangle$ 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 `\Alpha*`, ‘m’ for `\alpha*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

`widest` = { $\langle integer | string \rangle$ } default: *empty*

Sets the `labelwidth` key pass the $\langle integer \rangle$ or converting the $\langle string \rangle$ of the form `\Alpha`, `\alpha`, `\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` = { $\langle font commands \rangle$ } default: *empty*

Sets the *font style* for the current $\langle label \rangle$ defined by `label` key. For example `font={\bfseries\small}`.

`align` = { $\langle left | right | center \rangle$ } default: *left*

Sets the *aligned* of $\langle label \rangle$ defined by `label` key on the current level in the label box.

`wrap-label` = { $\langle code \{ \#1 \} more code \rangle$ } default: *empty*

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

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

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

`wrap-label*` = { $\langle code \{ \#1 \} more code \rangle$ } default: *empty*

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

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

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

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

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

5.2 Keys for penalties

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

`beginpenalty = {\integer}` default: *-51*

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

`midpenalty = {\integer}` default: *-51*

Set the *page breaking* penalty for breaking between items of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@itempenalty`.

`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. L^AT_EX 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 L^AT_EX or T_EX manual about how penalties control page breaks.

5.3 Keys for spaces

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

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

5.3.1 Vertical spaces

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

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

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

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

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

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

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

- The value of this parameter also affects the *inner levels* and the environments `keyans`, `keyanspic` and `keyans*`. Caution should be taken with “blank lines” or `\par` command “before” each environment or nested level when formatting the source code of document. T_EX will enter *vertical mode* and apply this value to the “top” and “bottom” the environment or nested level.

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

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

minus `1.0pt`, for the rest of the levels are `2.0pt` plus `1.0pt` minus `1.0pt`. For `keyans` and `keyans*` environments the default value is `4.0pt` plus `2.0pt` minus `1.0pt`.

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

`noitemsep` $\langle \text{value forbidden} \rangle$ 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` $\langle \text{value forbidden} \rangle$ 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` $\langle \text{value forbidden} \rangle$ 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 $\langle \text{keys} \rangle$ 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 ‘*’ $\langle \text{keys} \rangle$ applies `\vspace*` so that \LaTeX does *not discard* this space at page break.

`above` = $\{ \langle \text{rubber length} \mid \text{rigid length} \rangle \}$ 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*` = $\{ \langle \text{rubber length} \mid \text{rigid length} \rangle \}$ 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` = $\{ \langle \text{rubber length} \mid \text{rigid length} \rangle \}$ default: *not used*

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

`below*` = $\{ \langle \text{rubber length} \mid \text{rigid length} \rangle \}$ default: *not used*

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

5.3.2 Horizontal spaces

`list-offset` = $\{ \langle \text{rigid length} \rangle \}$ default: `0pt`

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

`list-indent` = $\{ \langle \text{rigid length} \rangle \}$ 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 $\langle \text{label} \rangle$ 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` = $\{ \langle \text{rigid length} \rangle \}$ default: `0pt`

Sets the extra *horizontal indentation*, beyond `labelsep`, of the “first line” off 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 *blank line* limitation or the `\par` command followed by `\item` you can modify `labelwidth` and `list-indent` and get the same effect.

`rightmargin = {⟨rigid length⟩}` default: `0pt`

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: `0pt`

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

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

5.4 Keys for add code

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

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

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

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

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

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

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

- Keep in mind that the `{⟨code⟩}` set in this *⟨key⟩* will affect the entire “body” of the environment and therefore the inner levels of the list and the `keyans`, `keyans*` and `keyanspic` environments. It is recommended to set this *⟨key⟩* per level. In the `enumext*` and `keyans*` environments this *⟨key⟩* is executed “after” the `listparindent`, `parsep` and `itemindent` *⟨keys⟩* within the `minipage` environment in which the “item content” is placed.

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

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

5.5 Keys for start, series and resume

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

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

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

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

- For compatibility with tagged PDF, the *start value* are set “after” the *second argument* to the `list` environment and “before” the execution of the first `\item` and the `first` key: `\begin{list}{⟨arg one⟩}{⟨arg two⟩}\setcounter{enumX}\item`.
- 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 of the “same level” as the environment in which the `series={⟨series name⟩}` key was executed. 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*`.

5.6 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.

5.6.1 The command `\resetenumext`

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

The `\resetenumext` command “resets” the *start value* of the “counters” for the `enumext` and `enumext*` environments along with the “internal counters” used by the keys `resume without value` and `resume*` according to the value of *{ <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.

5.7 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`.

5.8 Keys for minipage

`mini-env = { <rigid length> }`

default: *not used*

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

`mini-sep = { <rigid length> }`

default: *0.3333em*

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

5.8.1 The command `\miniright`

```
\miniright \begin{enumext}[mini-env={⟨rigid length⟩}] ⟨item's before⟩ \item \miniright ⟨content⟩ \end{enumext}
\begin{enumext}[mini-env={⟨rigid length⟩}] ⟨item's 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 L^AT_EX justification is maintained in the `minipage` on the “right side”.

5.8.2 The key `mini-right`

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

```
mini-right = {⟨content⟩} default: not used
```

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

```
mini-right* = {⟨content⟩} default: not used
```

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

6 The storage system

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

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

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

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

6.1 Keys for storage system

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

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

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

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

This key *overrides* the default “*stored keys*” of the *optional argument* of the inner levels or nested environment that will be passed to the *sequence*. The `⟨key list⟩` passed to this key ignores any `⟨keys⟩` in the “*stored structure*” and must be passed between braces. For example, if we execute 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* $\{\langle store\ name\rangle\}$ would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite and the “*stored key*” in the *sequence* $\{\langle store\ name\rangle\}$ are only `columns=3` ignoring all the others.

`save-sep = { $\langle text\ symbol\rangle$ }` default: {,}

Sets the *text symbol* that will separate the current $\langle label\rangle$ 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* $\{\langle store\ name\rangle\}$ set by `save-ans` key. The $\{\langle text\ symbol\rangle\}$ must always be passed between braces, whitespace ‘’ is preserved within the braces and only affects the “*stored content*” and not what is displayed when using the `show-ans` or `show-pos` keys.

`no-store $\langle value\ forbidden\rangle$` 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* $\{\langle store\ name\rangle\}$ 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 = { $\langle true\ |\ false\rangle$ }` default: *false*

Activates the “*internal label and ref*” mechanism for referencing “*stored content*” in *prop list* $\{\langle store\ name\rangle\}$ set by `save-ans` key. To reference the location of the “*stored content*” within the environment you must use `\ref{ $\langle store\ name\rangle$: $\langle position\rangle$ }`, where $\langle position\rangle$ corresponds to the position occupied by the “*stored content*” in the *prop list* $\{\langle store\ name\rangle\}$ 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 = { $\langle symbol\rangle$ }` 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 $\langle keys\rangle$ to set and manipulate “*symbol marks*” associated with “*answers*” and how they are displayed and stored in the *sequence* and *prop list*.

The $\langle keys\rangle$ 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 $\langle keys\rangle$ 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 $\langle keys\rangle$ 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 = { $\langle true\ |\ false\rangle$ }` default: *false*

Display the *symbol* set by the `mark-ans` key to the left of the *mandatory argument* $\langle content\rangle$ passed to the `\anskey` command and $\langle body\rangle$ 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 $\langle label\rangle$ 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 = { $\langle symbol\rangle$ }` 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 = { $\langle left\ |\ right\ |\ center\rangle$ }` default: *left*

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 = { $\langle rigid\ length\rangle$ }` default: `labelsep`

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

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

Wraps the *mandatory argument* $\langle content\rangle$ passed to the `\anskey` and the $\langle body\rangle$ in `anskey*` environment referenced by $\{\#1\}$ when using the `show-ans` or `show-pos` keys. The $\{\langle code\rangle\}$ must be passed between braces and only affects how the *argument* or *body* is displayed and NOT the “*stored content*” in the *sequence* and *prop*

`list {⟨store name⟩}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.

Keys for keyans, keyans* and keyanspic

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

`mark-pos*` = {⟨left | right | center⟩} default: `left`
 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 displaying an appropriate message on the terminal. 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`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L^AT_EX will be used.

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

6.2.1 Keys for `\anskey`

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

`break-col` `⟨value forbidden⟩` default: `not used`
 Stores `{⟨content⟩}` in the *sequence* `{⟨store name⟩}` of the form `\columnbreak \item ⟨content⟩`.

`item-join` = {⟨columns⟩} default: `not set`

Set the *number of columns* to be used for `\item(<columns>)` and stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item(<columns>) <content>`.

`item-star` $\langle value forbidden \rangle$

default: *not used*

Stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item* <content>`.

`item-sym*` = $\{\langle symbol \rangle\}$

default: *not set*

Sets the *symbol* for `\item*` when using the key `item-star` and stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item*[\langle symbol \rangle] <content>`. The *symbol* can be in text or math mode, for example `item-sym*={\ast}` stores `\item*[\ast] <content>`.

`item-pos*` = $\{\langle rigid length \rangle\}$

default: *not set*

Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item*[\langle symbol \rangle][\langle offset \rangle] <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.

*

2. Text containing our instructions or questions.

(a) Question.

*

3. Text containing our instructions or questions.

*

4. Text containing our instructions or questions.

*

6.3 The environment `anskey*`

`anskey*` `\begin{anskey*}[\langle key = val \rangle] \langle body content \rangle \end{anskey*}`

The environment `anskey*` takes a mandatory $\{\langle body content \rangle\}$ and “stores it” in the *sequence* and *prop list* $\{\langle store name \rangle\}$ set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] 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 $\langle body \rangle$ 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[13]. `\begin{anskey*}` and `\end{anskey*}` must be in different lines and should not appear within verbatim environments or commands. All $\langle keys \rangle$ must be passed separated by commas and “without separation” of the start of the environment.

Comments “%” or “any character” after `\begin{anskey*}` or `[\langle key = val \rangle]` 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 $\langle keys \rangle$ as the `\anskey` command next to the $\langle keys \rangle$ `write-env`, `overwrite` and `force-eol`. The environment is available for all levels of the `enumext` environment and the `enumxt*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

`write-env` = $\{\langle file.ext \rangle\}$

default: *not used*

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

`overwrite` = $\{\langle true | false \rangle\}$

default: *false*

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

`force-eol` = $\{\langle true | false \rangle\}$

default: *false*

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

Example

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

  \begin{anskey*}[item-star]
    \langle first answer \rangle
  \end{anskey*}

  \item Text containing our instructions or questions.

  \begin{enumext}
    \item Question.
    \begin{anskey*}
      \langle second answer \rangle
    \end{anskey*}
  \end{enumext}

  \item Text containing our instructions or questions.

  \begin{anskey*}
    \langle third answer \rangle
  \end{anskey*}

  \item Text containing our instructions or questions.

  \begin{anskey*}
    \langle fourth answer \rangle
  \end{anskey*}
\end{enumext}
```

- ★ 5. Text containing our instructions or questions.

[5] First answer with verbatim
6. Text containing our instructions or questions.

(a) Question.

[6] second answer
7. Text containing our instructions or questions.

[7] third answer
8. Text containing our instructions or questions.

[8] fourth answer

6.4 The environments keyans and keyans*

keyans	<code>\begin{keyans}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}</code>
keyans*	<code>\begin{keyans*}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}</code>

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[\langle custom \rangle]` work in the usual and the command `\item(\langle columns \rangle)` 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 \langle item content \rangle
  \begin{keyans}[\langle key = val \rangle]
    \item \langle item content \rangle
    \item [\langle custom \rangle] \langle item content \rangle
    \item* \langle item content \rangle
    \item*[\langle content \rangle] \langle item content \rangle
  \end{keyans}
\end{enumext}
```

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

The `\langle keys \rangle` 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[\langle keyans \rangle]{\langle key = val \rangle}` or `\setenumext[\langle keyans* \rangle]{\langle key = val \rangle}`. If the *optional argument* is not passed or the `\langle keys \rangle` 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 `\langle label \rangle` 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*[\langle content \rangle]`

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

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


Example

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

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

  \item Text containing a question and image.

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

1. Text containing a question.
A) Choice * B) Correct choice
C) Choice D) Choice
E) Choice
2. Text containing a question and image.
A) Choice
B) Choice
C) Choice
D) Choice
* E) [note] Correct choice
- 
Some text

6.5 The environment keyanspic

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

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

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

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

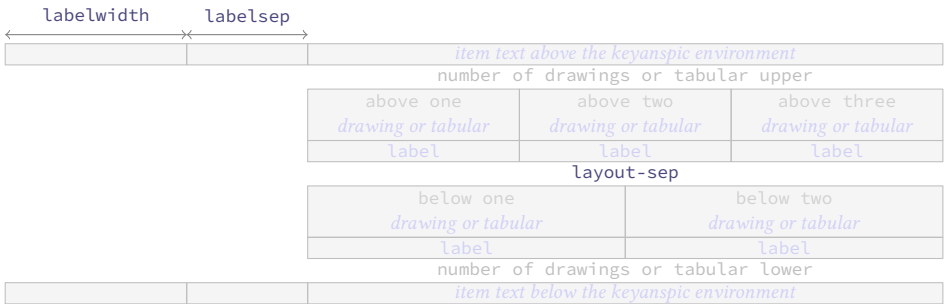


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

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

6.5.1 Keys for keyanspic

`label-pos = {⟨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: *internal adjustment*

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

`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: *adjusted parsep from keyans*

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

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

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

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

6.5.2 The command `\anspic`

```
\anspic {⟨drawing or tabular⟩}
\anspic* [⟨content⟩] {⟨drawing or tabular⟩}
```

The `\anspic` command take 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.

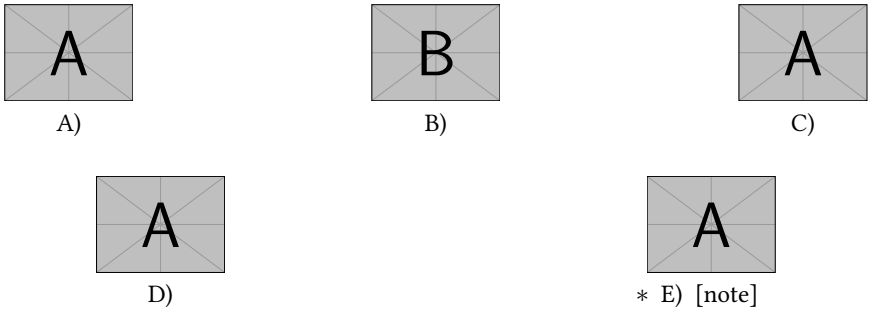
  \begin{keyanspic}[label-pos=above, layout-sty={3,2},layout-sep=0.25cm]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}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

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

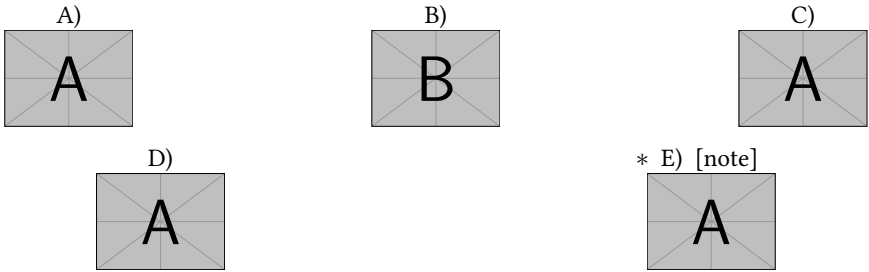
  \begin{keyanspic}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}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

\end{enumext}
```

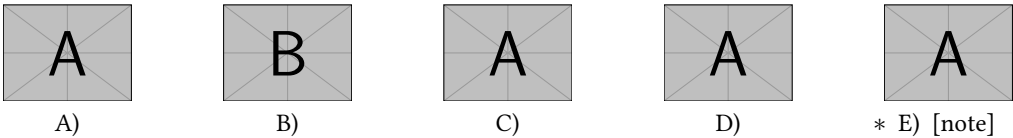
1. Question with images and labels below.



2. Question with images and labels above.



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



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*.

`before = {code}` default: *empty*

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

`after = $\langle code \rangle$` default: *empty*

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

`wrapper = $\langle code \rangle$ $\langle \#1 \rangle$ more code` default: *empty*

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

6.6.3 The command `\printkeyans`

```
\printkeyans  $\langle store name \rangle$ 
\printkeyans[ $\langle keys \rangle$ ] $\langle store name \rangle$ 
\printkeyans*[ $\langle keys \rangle$ ] $\langle store name \rangle$ 
```

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

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

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

The default values for the “first level” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the sequence $\langle store name \rangle$ the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the sequence $\langle store name \rangle$ it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

Since the command encapsulates by default the `enumext` environment or the `enumext*` environment, we must take some considerations:

- If we execute `\printkeyans* $\langle store name \rangle$` and the sequence $\langle store name \rangle$ already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans* $\langle store name \rangle$` and the sequence $\langle store name \rangle$ contains any `enumext` environments, they will start with the $\langle keys \rangle$ set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.
- If we execute `\printkeyans $\langle store name \rangle$` and the sequence $\langle store name \rangle$ contains any environment `enumext*`, they will start with the $\langle keys \rangle$ 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[$\langle print \rangle$, $\langle i \rangle$] $\langle keys \rangle$` and `\setenumext[$\langle print \rangle$ *, $\langle i \rangle$] $\langle keys \rangle$` .

If we need to set the $\langle keys \rangle$ for the environment `enumext` “saved” in the sequence $\langle store name \rangle$ we will use `\setenumext[$\langle print \rangle$, $\langle level \rangle$] $\langle keys \rangle$` and if we need to set the $\langle keys \rangle$ for the environment `enumext*` “saved” in the sequence $\langle store name \rangle$ we will use `\setenumext[$\langle print \rangle$ *, $\langle i \rangle$] $\langle keys \rangle$` .

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]

2. True False

(a) ~~TeX~~ze is cool?

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i. `xsim`

[4]

ii. `exsheets`

[5]

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

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

7 Full examples

Here I will leave as an example some adaptations questions taken from TeX-SX. The examples are attached to this documentation and can be extracted from your PDF viewer or from the command line by running:

```
$ pdftdetach -saveall enumext.pdf
```

and then you can use the excellent [arara](#)¹ tool to compile them.

Example 1

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

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

 36 km/h.
- B

 360 km/h.
- C

 27,8 km/h.
- D

 $3,60 \times 10^8$ km/h.

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

A

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

B

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

C

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

D

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

A

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

B

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

C


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

D

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

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

Example 2

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

¹The cool TeX automation tool: <https://www.ctan.org/pkg/arara>

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

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

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

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

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

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

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

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

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

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


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

✖ 2. A

✖

3. B

✖ 4. A

✖
- Example 3
- A “simple multiple choice” test .
1. First type of questions

A value

B correct

C value

D value

2. Second type of questions

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

II. $\alpha = \delta$

III. $\angle EDF = 45^\circ$

A I only

B II only

C I and II only

D I and III only

E I, II, and III

3. Third type of questions

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

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

A value

B value

C value

D value

E value

4. Question with image and label below:

A

A

B


B

A

C

A

D



E

5. Question with image on right side:

A value

B value

C value

D correct

E value

B

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 :) .

 Factor $x^2 - 2x + 1$


 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

(a) $\alpha > \delta$

(b) \LaTeX is cool?

 Related to Linux

(a) You use linux?

©2024–2025 by Pablo González L


22 / 169

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

Example 5

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

<div>1</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.	<div>3</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.
<div>2</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.	<div>4</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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 .

- 8.5a, KSC 10. sample
- A sample
 - ✓ B answer
 - C sample
 - D sample
- 9.5a, KSC 11. sample
- A sample
 - B sample
 - C sample
 - ✓ D answer
12. sample
- A sample
 - B answer
 - C sample
 - D sample
13. sample
- A sample
 - B sample
 - C sample
 - D answer

10. B (8.5a, KSC)
11. D (9.5a, KSC)

12. B (10.5a, KSC)
13. D (11.5a, KSC)













8 Tagged PDF examples

This section is just to show the compatibility of `enumext` with *tagged* PDF using `lualatex`. The attached files here 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 `multicol` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file  and *tagged* PDF .
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged* PDF .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged* PDF .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file  and *tagged* PDF .
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake itemize* and *description*. Source file  and *tagged* PDF .

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?.

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

Fake itemize environment

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

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

Fake description environment

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

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.

If we add `list-indent=0pt` you get *widest style*:

SomeThing A short one-line description.
This is an entry *without* a label.
Something A short *one-line* description text.
Something long A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- The small space at the beginning of the “unlabeled entry” corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.
- When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makeLabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[⟨content⟩]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

Here we set `label={}` and we will give a convenient value to `labelsep` and `labelwidth`, for example we can take as reference our *longest label* and pass 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, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

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

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

If we add `align=right` it will look like this:

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, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

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

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 what we will do is create a command in the style `\parleft` of `enumitem` that allows us to place *multiline labels* using `\parbox`.

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

Now we just need 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, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[5]` 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[11]` and `l3seq[11]` modules together with the `hyperref[8]` and `enumitem[6]` packages, which did the job, but with some limitations.

As time went by I took these limitations 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.

10 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2025.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2025.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2025.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2025.
- [7] BERRY, KARL. “ $\text{\LaTeX}_{2\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2025.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2025.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.

- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.
- [11] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.
- [12] The \LaTeX Project. “The \LaTeX 2 ϵ sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [13] The \LaTeX Project. “ \LaTeX News, Issue 41, June 2025”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [14] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2025.
- [15] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [16] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [17] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.
- [18] FISCHER, ULRIKE. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2025.
- [19] The \LaTeX Project. “latex-lab – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [20] MITTELBACH, FRANK. “ \LaTeX ’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.

11 Change history

- v1.6 (ctan), 2025-06-30**
 - Add `\resetenumext`, `reset` and `reset*` keys.
 - The `resume`, `resume*` and `series` keys can now be set per level.
 - Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.
- v1.5 (ctan), 2025-06-11**
 - Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
 - Add keys `beginpenalty`, `midpenalty` and `endpenalty`.
- v1.4 (ctan), 2025-06-09**
 - Improved implementation of the `start` key for *tagged* PDF.
 - Improved implementation of the `ref` key.
 - Fixed the behavior of the `save-sep` key.
 - Fixed the behavior of the `resume*` key.
- v1.3 (ctan), 2025-06-01**
 - Removed dependency on the `scontents` package.
 - The `anskey*` environment has been rewritten using the new `c`-type argument.
- v1.2 (ctan), 2025-03-28**
 - Replace signature (prevent expansion for optional argument).
 - Solve Inconsistent local/global assignment.
- v1.1 (ctan), 2024-11-14**
 - Fixed implementation for `font` and `base-fix` keys.
 - Added new keys for symbol marks.
 - Update and improvements in the internal code.
 - Adjustments in the documentation.
- v1.0 (ctan), 2024-11-01**
 - First public release.

12 Index of Documentation

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

C		F	
Document class:		\footnote	5
article	2	I	
book	2	\itemsep	8
exam	2	K	
letter	2	Keys for \anskey provide by enumext:	
report	2	break-col	14
\columnbreak	4, 14	item-join	14
\columnsep	11	item-pos*	15
Commands provide by enumext:		item-star	15
\anskey	12–15	item-sym*	15
\anspic	12–14, 17, 18	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	19	after	20
\getkeyans	14, 19	before	19
\item*	5–7, 12–14, 16, 17	sep	19
\item	5–7, 10, 12, 14, 16, 17	start	19
\miniright	12	step	19
\printkeyans	6, 13, 20	stop	19
\resetenumext	11	wrapper	20
\setenumextmeta	6	Keys for anskey* provide by enumext:	
\setenumext	5–7, 12, 14, 16, 20	break-col	14
Counters defined by enumext:		force-eol	15
enumXiii	4	item-join	14
enumXii	4	item-pos*	15
enumXiv	4	item-star	15
enumXi	4	item-sym*	15
enumXviii	4	overwrite	15
enumXvii	4	write-env	15
enumXvi	4	Keys for environments provide by enumext:	
enumXv	4	above*	9
E		above	9
Environments provide by enumext:		after	10
anskey*	12–15, 24	align	7, 14, 24, 25
enumext*	4–17, 20, 24	base-fix	9
enumext	4–17, 20, 24	before*	10
keyans*	4–16, 24	before	10
keyanspic	4, 7, 8, 10, 12–15, 17, 24	beginpenalty	8
keyans	4–18, 24	below*	9
Environments:		below	9
Verbatim	15	check-ans	13, 14
center	5	columns-sep	4, 11, 24
description	5, 24, 25	columns	4, 9, 11, 24
enumerate	1, 3, 5, 26	endpenalty	8
figure	5	first	10
flushleft	5	font	7, 13, 14
flushright	5	item-pos*	5, 6
itemize	5, 24	item-sym*	5, 6
list	3, 5, 10, 26	itemindent	9, 10
minipage	3–5, 8–12, 24, 26	itemsep	8, 9
multicols	3, 4, 11, 24	label-pos	18
quotation	5	label-sep	18
quote	5	labelsep	3–7, 9, 11, 24, 25
shortenenumerate	5	labelwidth	3, 4, 6, 7, 9, 11, 13, 14, 24, 25
tabbing	5	labelwith	5
table	5	label	7, 8, 10, 16, 17, 24, 25
tasks	5	labewdith	9
trivlist	5	layout-sep	18
verbatim	5	layout-sty	17, 18
verse	5	layout-top	18

list-indent	3, 9, 10
list-offset	3, 9, 25
listparindent	10
mark-ans*	13, 14, 16, 18
mark-ans	13
mark-pos*	14, 16, 18
mark-pos	13
mark-ref	13
mark-sep*	14, 16, 18
mark-sep	13
midpenalty	8
mini-env	4, 9, 11, 12, 24
mini-right*	7, 12
mini-right	7, 12, 24
mini-sep	4, 11, 12
mode-box	7
no-store	12–15, 24
noitemsep	9
nosep	9, 24
overwrite	15
parsep	8–10, 18
partopsep	8
ref	4, 8, 24
reset*	6, 10, 11
reset	6, 10, 11
resume*	6, 7, 10–12
resume	6, 7, 10–12
rightmargin	10
save-ans	4, 6, 10–20, 24
save-key	10, 12, 13, 20
save-ref	4, 7, 13–15, 19, 24
save-sep	13, 16, 18, 24
series	6, 7, 10–12
show-ans	13, 14, 16, 18, 24
show-length	8
show-pos	13, 14, 16, 18, 19
start*	10, 11
start	10, 11
topsep	8, 9, 18
widest	7
wrap-ans*	14, 16, 18
wrap-ans	13
wrap-label*	7, 25
wrap-label	7, 13, 14, 24, 25
wrap-opt	13, 14, 16, 18
write-env	15
L	
\label	4
Labels provide by enumext:	
\Alph*	7, 8, 16
\Roman*	7, 8
\alph*	7, 8
\arabic*	7, 8
\roman*	7, 8
\labelsep	3, 7
\labelwidth	3, 7
\linewidth	11
\listparindent	10
P	
Packages:	
enumerate	26
enumext	1–5, 7, 13, 17, 24, 26
enumitem	3, 4, 25, 26
fancyvrb	15
footnotehyper	5
geometry	24
graphicx	24
hyperref	4, 5, 13–15, 24, 26
l3keys	7
l3prop	26
l3seq	26
luamml	24
multicol	1, 2, 4, 24, 26
scontents	27
shortlst	5
tasks	5
task	6
unicode-math	24
xsim	2
\parsep	8
\partopsep	8
R	
\raggedcolumns	4
\ref	4
\rightmargin	10
T	
\topsep	8

13 Implementation

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

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

13.1 General conventions

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

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

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

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

13.2 Initial set up

Start the DocStrip guards.

```
1 (*package)
```

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

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

13.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L^AT_EX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2025-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage {enumext} {2025-06-30} {1.6} {Enumerate exercise sheets}
```

Finally check if the `multicol` package are loaded, if not we load it.

```
5 \hook_gput_code:nnn {begindocument} {enumext}
6 {
7   \IfPackageLoadedTF { multicol }
8   {
9     \msg_info:nnn { enumext } { package-load } { multicol }
10  }
11  {
12    \msg_info:nnn { enumext } { package-not-load } { multicol }
13    \RequirePackage{multicol}[2025-05-25]
14  }
15 }
```

13.4 Definition of variables

Variables that do not appear in this section are created by means of `\keys_define:nn` or some function described below.

Integer variables will control the nesting levels of the environments, `anskey*` environment and `\anskey` command.

```
\__enumext_level_int
\__enumext_level_h_int
\__enumext_anskey_level_int
\__enumext_keyans_level_int
\__enumext_keyans_level_h_int
\__enumext_keyans_pic_level_int

16 \int_new:N \__enumext_level_int
17 \int_new:N \__enumext_level_h_int
18 \int_new:N \__enumext_anskey_level_int
19 \int_new:N \__enumext_keyans_level_int
20 \int_new:N \__enumext_keyans_level_h_int
21 \int_new:N \__enumext_keyans_pic_level_int
```

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

```

\l__enumext_starred_bool
\g__enumext_starred_bool
  \l_enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
  \l_enumext_standar_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:` (§13.5.1).

```

22 \bool_new:N \l__enumext_starred_bool
23 \bool_new:N \g__enumext_starred_bool
24 \bool_new:N \l__enumext_starred_first_bool
25 \bool_new:N \l__enumext_standar_bool
26 \bool_new:N \g__enumext_standar_bool
27 \bool_new:N \l__enumext_standar_first_bool
28 \bool_new:N \l__enumext_keyans_env_bool
29 \tl_new:N \g__enumext_start_line_tl
30 \tl_new:N \g__enumext_envir_name_tl
31 \tl_new:N \l__enumext_envir_name_tl

```

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

```

\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 (§13.14).

```

32 \cs_set_protected:Npn \__enumext_tmp:n #1
33 {
34   \tl_new:c { \l__enumext_counter_#1_tl }
35 }
36 \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.)

```

\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 (§13.14).

```

37 \tl_new:N \l__enumext_ref_key_arg_tl
38 \tl_new:N \l__enumext_ref_the_count_tl
39 \cs_set_protected:Npn \__enumext_tmp:n #1
40 {
41   \tl_new:c { \l__enumext_renew_counter_#1_tl }
42   \tl_new:c { \l__enumext_the_counter_#1_tl }
43   \tl_set:ce { \l__enumext_the_counter_#1_tl } { \exp_not:c { \theenumX#1 } }
44 }
45 \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.)

```

\l__enumext_series_name_tl
\l__enumext_resume_X_bool
  \g_enumext_save_last_keys_X_tl
\g__enumext_series_name_X_tl

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.26).

```

46 \tl_new:N \l__enumext_series_name_tl
47 \bool_new:N \l__enumext_resume_count_bool
48 \cs_set_protected:Npn \__enumext_tmp:n #1
49 {
50   \bool_new:c { \l__enumext_resume_#1_bool }
51   \bool_new:c { \l__enumext_resume_empty_#1_bool }
52   \bool_new:c { \l__enumext_resume_series_#1_bool }
53   \tl_new:c { \g__enumext_save_last_keys_#1_tl }
54   \tl_new:c { \l__enumext_series_name_#1_tl }
55 }
56 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

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

```

\l_enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
  \l_enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default *label style* and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§13.15) and `label` (§13.13) keys.

```

57 \dim_new:N \l__enumext_current_widest_dim
58 \tl_new:N \g__enumext_counter_styles_tl
59 \tl_new:N \g__enumext_widest_label_tl
60 \box_new:N \l__enumext_label_width_by_box

```

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

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

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.19). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.41.1).

```
61 \cs_set_protected:Npn \__enumext_tmp:n #1
62 {
63   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
64   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
65   \dim_new:c { \l__enumext_leftmargin_#1_dim }
66   \dim_new:c { \l__enumext_itemindent_#1_dim }
67 }
68 \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` and others.)

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str
```

Internal variables used by `columns` key (§13.23) and `align` key (§13.13).

```
69 \cs_set_protected:Npn \__enumext_tmp:n #1
70 {
71   \skip_new:c { \l__enumext_multicols_above_#1_skip }
72   \skip_new:c { \l__enumext_multicols_below_#1_skip }
73   \skip_new:c { \g__enumext_multicols_right_#1_skip }
74   \str_new:c { \l__enumext_align_label_pos_#1_str }
75 }
76 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

```
\g__enumext_minipage_stat_int
\l__enumext_minipage_temp_skip
\l__enumext_minipage_left_skip
\l__enumext_minipage_right_skip
\l__enumext_minipage_after_skip
\g__enumext_minipage_right_skip
\g__enumext_minipage_after_skip
\l__enumext_minipage_left_X_dim
\l__enumext_minipage_active_X_bool
```

Internal variables used by `\miniright` command (§13.24.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.22, §13.24).

```
77 \int_new:N \g__enumext_minipage_stat_int
78 \skip_new:N \l__enumext_minipage_temp_skip
79 \skip_new:N \l__enumext_minipage_left_skip
80 \skip_new:N \l__enumext_minipage_right_skip
81 \skip_new:N \l__enumext_minipage_after_skip
82 \skip_new:N \g__enumext_minipage_right_skip
83 \skip_new:N \g__enumext_minipage_after_skip
84 \cs_set_protected:Npn \__enumext_tmp:n #1
85 {
86   \dim_new:c { \l__enumext_minipage_left_#1_dim }
87   \bool_new:c { \l__enumext_minipage_active_#1_bool }
88 }
89 \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_wrap_label_X_bool
\l__enumext_wrap_label_opt_X_bool
\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl
\l__enumext_label_fill_left_X_tl
\l__enumext_label_fill_right_X_tl
\l__enumext_vspace_a_star_X_bool
\l__enumext_vspace_b_star_X_bool
```

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.19.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.21).

```
90 \cs_set_protected:Npn \__enumext_tmp:n #1
91 {
92   \bool_new:c { \l__enumext_wrap_label_#1_bool }
93   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
94   \int_new:c { \l__enumext_start_#1_int }
95   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
96   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
97   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
98   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
99   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
100 }
101 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```

\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` setting by `save-ans` key (§13.29.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` saves the $\{\langle store\ name\rangle\}$ set by the `save-ans` key of the *sequence* and *prop list* in which we will store, the variable `\g__enumext_store_name_tl` it's just a global copy of $\{\langle store\ name\rangle\}$ used by different functions.

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.40) and `\anspic*` (§13.45.2) for the `keyans`, `keyans*` and `keyanspic` environments.

```

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

```

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

```

\l__enumext_store_anskey_arg_tl
\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_arg_tl` save the *argument* of `\anskey` (§13.33) and the variables `\l__enumext_store_anskey_env_tl` save the *(body)* of the environment `anskey*` (§13.34).

The variables `\l__enumext_write_anskey_env_bool`, `\l__enumext_write_anskey_env_file_name_tl` and `\l__enumext_write_anskey_env_file_iow` they are used by the `write-env` and `overwrite` keys in the `anskey*` environment implementation.

```

107 \tl_new:N \l__enumext_store_anskey_arg_tl
108 \tl_new:N \l__enumext_store_anskey_env_tl
109 \bool_new:N \l__enumext_write_anskey_env_bool
110 \tl_new:N \l__enumext_write_anskey_env_file_name_tl
111 \iow_new:N \l__enumext_write_anskey_env_file_iow

```

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

```

\c__enumext_anskey_env_hidden_space_str

```

The `\c__enumext_anskey_env_hidden_space_str` is a constant *string* to used to hide the *(forced space)* added by T_EX when recording content in a macro. This *string* contains the *reserved phrase* “`%^^Aenumextheol%`” which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```

112 \str_const:Ne \c__enumext_anskey_env_hidden_space_str
113 { \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`.)

```

\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` (§13.51).

```

114 \tl_new:N \l__enumext_setkey_tmpa_tl
115 \tl_new:N \l__enumext_setkey_tmpb_tl
116 \int_new:N \l__enumext_setkey_tmpa_int
117 \seq_new:N \l__enumext_setkey_tmpa_seq
118 \seq_new:N \l__enumext_setkey_tmpb_seq

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§13.50) and `\foreachkeyans` command (§13.53).

```

119 \tl_new:N \l__enumext_meta_path_tl
120 \seq_new:N \l__enumext_foreach_print_seq
121 \tl_new:N \l__enumext_foreach_name_prop_tl
122 \tl_new:N \l__enumext_foreach_default_keys_tl

```

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

```

\l__enumext_print_keyans_starred_tl
\l__enumext_print_keyans_star_bool
\l__enumext_print_keyans_cmd_bool
\l__enumext_mark_position_str
\l__enumext_mark_position_v_str
\l__enumext_mark_position_viii_str
\l__enumext_mark_sep_tmpa_dim
\l__enumext_mark_sep_tmpb_dim
\l__enumext_show_pos_tmp_int
\g__enumext_item_symbol_aux_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool

```

Internal variables used by command `\printkeyans` (§13.50), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.30), `item-sym*` key (§13.38), `save-key` key (§13.30.3) and “*storing structure*”.

```

123 \tl_new:N \l__enumext_print_keyans_starred_tl
124 \bool_new:N \l__enumext_print_keyans_star_bool
125 \bool_new:N \l__enumext_print_keyans_cmd_bool
126 \str_new:N \l__enumext_mark_position_str
127 \str_new:N \l__enumext_mark_position_v_str
128 \str_new:N \l__enumext_mark_position_viii_str
129 \dim_new:N \l__enumext_mark_sep_tmpa_dim
130 \dim_new:N \l__enumext_mark_sep_tmpb_dim
131 \int_new:N \l__enumext_show_pos_tmp_int
132 \tl_new:N \g__enumext_item_symbol_aux_tl
133 \cs_set_protected:Npn \l__enumext_tmp:n #1
134 {
135   \tl_new:c { l__enumext_print_keyans_#1_tl }

```

```

136      \tl_new:c { \__enumext_store_save_key_#1_tl }
137      \bool_new:c { \__enumext_store_save_key_#1_bool }
138      \bool_new:c { \__enumext_store_upper_level_#1_bool }
139    }
140    \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

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

Internal variables used by `keyanspic` environment and `\anspic` command (§13.45.1).

```

141 \seq_new:N \__enumext_anspic_args_seq
142 \dim_new:N \__enumext_anspic_mini_width_dim
143 \int_new:N \__enumext_anspic_above_int
144 \int_new:N \__enumext_anspic_below_int
145 \bool_new:N \__enumext_anspic_label_above_bool
146 \str_new:N \__enumext_anspic_mini_pos_str
147 \box_new:N \__enumext_anspic_label_box
148 \box_new:N \__enumext_anspic_body_box
149 \dim_new:N \__enumext_anspic_label_htdp_dim
150 \dim_new:N \__enumext_anspic_body_htdp_dim

```

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

Internal variables used by “*internal check answer*” mechanism (§13.29.3) 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.

```

151 \bool_new:N \__enumext_check_answers_bool
152 \bool_new:N \g__enumext_check_ans_key_bool
153 \tl_new:N \__enumext_check_start_line_env_tl
154 \bool_new:N \__enumext_item_wrap_key_bool
155 \int_new:N \g__enumext_check_starred_cmd_int
156 \int_new:N \g__enumext_item_anskey_int
157 \int_new:N \g__enumext_item_number_int
158 \bool_new:N \__enumext_item_number_bool
159 \int_new:N \g__enumext_item_answer_diff_int

```

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

The boolean variable `__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.7). The boolean variable `__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

160 \bool_new:N \__enumext_hyperref_bool
161 \bool_new:N \__enumext_footnotes_key_bool

```

(End of definition for `__enumext_hyperref_bool` and `__enumext_footnotes_key_bool`.)

Internal variables used by `save-ref` key (§13.30). The variables `__enumext_label_copy_X_tl` correspond to temporary copies of the *⟨labels⟩* defined by level on which operations will be performed.

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

```

162 \tl_new:N \__enumext_newlabel_arg_one_tl
163 \tl_new:N \__enumext_newlabel_arg_two_tl
164 \tl_new:N \__enumext_write_aux_file_tl
165 \cs_set_protected:Npn \__enumext_tmp:n #1
166 {
167   \tl_new:c { \__enumext_label_copy_#1_tl }
168 }
169 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

170 \int_new:N \g__enumext_footnote_standar_int
171 \int_new:N \g__enumext_footnote_starred_int
172 \seq_new:N \g__enumext_footnote_standar_arg_seq
173 \seq_new:N \g__enumext_footnote_starred_arg_seq
174 \seq_new:N \g__enumext_footnote_standar_int_seq
175 \seq_new:N \g__enumext_footnote_starred_int_seq

```

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


```

\l__enumext_item_starred_X_bool
\l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_tmpa_X_dim
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\l__enumext_miniright_code_X_box
\g__enumext_minipage_center_X_bool
\g__enumext_minipage_right_X_dim
\g__enumext_minipage_right_X_skip

```

Internal variables used by `enumext*` and `keyans*` environments.

```

176 \cs_set_protected:Npn \l__enumext_tmp:n #1
177 {
178   \bool_new:c { \l__enumext_item_starred_#1_bool }
179   \int_new:c { \l__enumext_item_column_pos_#1_int }
180   \int_new:c { \g__enumext_item_count_all_#1_int }
181   \int_new:c { \l__enumext_joined_item_#1_int }
182   \int_new:c { \l__enumext_joined_item_aux_#1_int }
183   \int_new:c { \l__enumext_tmpa_#1_int }
184   \dim_new:c { \l__enumext_tmpa_#1_dim }
185   \box_new:c { \l__enumext_item_text_#1_box }
186   \dim_new:c { \l__enumext_joined_width_#1_dim }
187   \dim_new:c { \l__enumext_item_width_#1_dim }
188   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
189   \str_new:c { \l__enumext_align_label_#1_str }
190   \bool_new:c { \g__enumext_minipage_active_#1_bool }
191   \box_new:c { \l__enumext_miniright_code_#1_box }
192   \bool_new:c { \g__enumext_minipage_center_#1_bool }
193   \dim_new:c { \g__enumext_minipage_right_#1_dim }
194   \skip_new:c { \g__enumext_minipage_right_#1_skip }
195 }
196 \clist_map_inline:nn { vii, viii } { \l__enumext_tmp:n {#1} }

```

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

```
\c__enumext_all_envs_clist
```

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

```

197 \clist_const:Nn \c__enumext_all_envs_clist
198 {
199   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
200   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
201 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

```

\keys_precompile:neN
\seq_use:NV

```

Non-standard kernel variants used by the `\printkeyans` command (§13.50) and `\foreachkeyans` command (§13.53).

```

202 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
203 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

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

```
\l__enumext_scan_tokens:n
```

The functions `\tl_rescan:n` and `\tl_set_rescan:Nnn` provided by `expl3` doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim stuff used inside one of `anskey*` environment will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in [How do use {<setup> in \tl_set_rescan:Nnn to replace \scantokens?](#).

```
204 \cs_new_protected:Npn \l__enumext_scan_tokens:n #1 { \tex_scantokens:D {#1} }
```

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

```
\l__enumext_at_begin_document:n
```

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

```

205 \cs_new_protected:Npn \l__enumext_at_begin_document:n #1
206 {
207   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
208 }

```

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

```

\__enumext_after_env:nn
\__enumext_before_env:nn

```

A internal “hook” functions for execute code `mini-right` and `mini-right*` keys outside the `enumext*` and `keyans*` environments and print check-ans outside the `enumext` and `enumext*` environments.

```

209 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
210 {
211   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
212 }
213 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
214 {
215   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
216 }

```

(End of definition for `__enumext_after_env:nn` and `__enumext_before_env:nn`.)

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

```
217 \cs_new:Nn \__enumext_level:
218 {
219   \int_to_roman:n { \__enumext_level_int }
220 }
```

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT` A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys.
`__enumext_if_is_int:nF` 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?](#)
`__enumext_if_is_int:nTF`

```
221 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
222 {
223   \regex_if_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
224   { \prg_return_true: }
225   { \prg_return_false: }
226 }
```

(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 use in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```
227 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
228 {
229   *~#2
230   \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
231   =~\use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
232 }
```

(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` according to the case.

```
233 \cs_new_protected:Nn \__enumext_unskip_unkern:
234 {
235   \int_case:nnT { \lastnodetype }
236   {
237     { 11 }{ \unskip }
238     { 12 }{ \unkern }
239   }
240 }
```

(End of definition for `__enumext_unskip_unkern:.`)

13.5.1 Utilities for environments and levels

`__enumext_is_not_nested:` The function `__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “true” only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.
`__enumext_is_on_first_level:`

```
241 \cs_new_protected:Nn \__enumext_is_not_nested:
242 {
243   \str_case:en { \@currenvir }
244   {
245     {enumext}
246     {
247       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
248       \bool_lazy_and:nnT
249       { \bool_not_p:n { \g__enumext_standar_bool } }
250       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
251       {
252         \bool_gset_true:N \g__enumext_standar_bool
253       }
254     }
255     {enumext*}
256     {
257       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
258       \bool_lazy_and:nnT
259       { \bool_not_p:n { \g__enumext_starred_bool } }
260       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
```

```

261         {
262             \bool_gset_true:N \g__enumext_starred_bool
263         }
264     }
265 }
266 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.29.1), `\l__enumext_starred_first_bool` (§13.29.1) to “true” only if the environment is not nested and we are in the “first level” of it. We will 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.

```

267 \cs_new_protected:Nn \__enumext_is_on_first_level:
268 {
269     \bool_lazy_all:nT
270     {
271         { \bool_if_p:N \g__enumext_standar_bool }
272         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
273         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
274     }
275     {
276         \bool_set_true:N \l__enumext_standar_first_bool
277         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
278         \tl_gset:Nn \g__enumext_start_line_tl
279             {
280                 on~line~\exp_not:V \inputlineno
281             }
282     }
283     \bool_lazy_all:nT
284     {
285         { \bool_if_p:N \g__enumext_starred_bool }
286         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
287         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
288     }
289     {
290         \bool_set_true:N \l__enumext_starred_first_bool
291         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
292         \tl_gset:Nn \g__enumext_start_line_tl
293             {
294                 on~line~\exp_not:V \inputlineno
295             }
296     }
297 }

```

(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.

```

298 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
299 {
300     \str_case:en { \@currentenvir }
301     {
302         {keyans}
303         {
304             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
305             \tl_set:Nn \l__enumext_check_start_line_env_tl
306                 {
307                     in~'keyans'~start~on~line~\exp_not:V \inputlineno
308                 }
309         }
310         {keyans*}
311         {
312             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
313             \tl_set:Nn \l__enumext_check_start_line_env_tl
314                 {
315                     in~'keyans*'~start~on~line~\exp_not:V \inputlineno
316                 }
317         }
318         {keyanspic}
319         {

```

```

320         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
321         \tl_set:Nc \l__enumext_check_start_line_env_tl
322             {
323             in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
324             }
325     }
326 }
327 }

```

(End of definition for `__enumext_keyans_name_and_start:`)

13.5.2 Utilities for log and terminal

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

```

\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
328 \cs_new_protected:Nn \__enumext_reset_global_vars:
329 {
330     \__enumext_reset_global_int:
331     \__enumext_reset_global_bool:
332     \__enumext_reset_global_tl:
333 }
334 \cs_new_protected:Nn \__enumext_reset_global_int:
335 {
336     \int_gzero:N \g__enumext_item_number_int
337     \int_gzero:N \g__enumext_item_anskey_int
338     \int_gzero:N \g__enumext_item_answer_diff_int
339 }
340 \cs_new_protected:Nn \__enumext_reset_global_bool:
341 {
342     \bool_gset_false:N \g__enumext_check_ans_key_bool
343     \bool_gset_false:N \g__enumext_standar_bool
344     \bool_gset_false:N \g__enumext_starred_bool
345 }
346 \cs_new_protected:Nn \__enumext_reset_global_tl:
347 {
348     \tl_gclear:N \g__enumext_store_name_tl
349     \tl_gclear:N \g__enumext_start_line_tl
350     \tl_gclear:N \g__enumext_envir_name_tl
351 }

```

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

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

```

352 \cs_new_protected:Nn \__enumext_log_global_vars:
353 {
354     \msg_log:nneeee { enumext } { prop-seq-int-hook }
355     { \g__enumext_store_name_tl }
356     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
357     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
358     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
359 }

```

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

```

360 \cs_new_protected:Nn \__enumext_log_answer_vars:
361 {
362     \msg_log:nneeee { enumext } { item-answer-hook }
363     { \int_use:N \g__enumext_item_number_int }
364     { \int_use:N \g__enumext_item_anskey_int }
365     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
366 }

```

(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:

```
\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 we copy them using `__enumext_at_begin_document:n` in case any package redefines the `list` environment or a related command.

🔍 For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `ltxcmd` (see `latex-lab-block`[19]).

```
\__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.

```
367 \__enumext_at_begin_document:n
368 {
369   \cs_new_eq:NN \__enumext_start_list:nn \list
370   \cs_new_eq:NN \__enumext_stop_list: \endlist
371   \NewCommandCopy \__enumext_item_std:w \item
372 }
```

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

```
373 \__enumext_at_begin_document:n
374 {
375   \cs_new_eq:NN \__enumext_minipage:w \minipage
376   \cs_new_eq:NN \__enumext_endminipage: \endminipage
377 }
```

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

13.7 Compatibility with hyperref and footnotehyper

```
\__enumext_after_hyperref:
  \__enumext_hypertarget:nn
  \__enumext_phantomsection:
```

First we define the necessary rules using “hooks” to determine if the `hyperref` package is loaded.

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

The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `\l__enumext_footnotes_key_bool` to “true”.

```
380 \cs_new_protected:Nn \__enumext_after_hyperref:
381 {
382   \IfPackageLoadedT { hyperref }
383   {
384     \msg_info:nnn { enumext } { package-load } { hyperref }
385     \bool_set_true:N \l__enumext_hyperref_bool
386     \IfHyperBoolean{hyperfootnotes}
387     {
388       \bool_set_true:N \l__enumext_footnotes_key_bool
389     }
390     { }
391   }
```

If the state of the variable `\l__enumext_footnotes_key_bool` is true we will check if the package `footnotehyper` is loaded, in case it is not present, we will set the value of `\l__enumext_footnotes_key_bool` to false and we will redefine `\footnote`.

```
392   \bool_if:NT \l__enumext_footnotes_key_bool
393   {
394     \IfPackageLoadedTF { footnotehyper }
395     {
396       \msg_info:nnn { enumext } { package-load } { footnotehyper }
397     }
398     {
399       \bool_set_false:N \l__enumext_footnotes_key_bool
```

```

400     }
401 }

```

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

```

402 \bool_if:NTF \l__enumext_hyperref_bool
403 {
404   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
405   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
406 }
407 {
408   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
409   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
410 }
411 }

```

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

`__enumext_newlabel:nn`

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

#1: `\l__enumext_newlabel_arg_one_tl`
 #2: `\l__enumext_newlabel_arg_two_tl`

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

```

412 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
413 {
414   \protected@write \@auxout { }
415   {
416     \token_to_str:N \newlabel {#1}
417     {
418       {#2}
419       \bool_if:NT \l__enumext_hyperref_bool
420       { { \thepage } {#2} {#1} }
421       { }
422     }
423   }
424   \__enumext_hypertarget:nn {#1} { }
425   \__enumext_phantomsection:
426 }

```

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

13.8 Internal redefining `\footnote` command

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

`__enumext_footnotetext:nn`
`__enumext_renew_footnote:`
`__enumext_print_footnote:`
`__enumext_renew_footnote_mini:`
`__enumext_print_footnote_mini:`

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

```

427 \cs_new_protected:Nn \__enumext_footnotetext:nn
428 {
429   \footnotetext[#1]{#2}
430 }
431 \cs_new_protected:Nn \__enumext_renew_footnote:
432 {
433   \RenewDocumentCommand \footnote { o +m }
434   {
435     \tl_if_novalue:nTF {##1}
436     {
437       \stepcounter{footnote}
438       \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
439     }
440     {
441       \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
442     }
443     \footnotemark [ \g__enumext_footnote_standar_int ]
444     \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
445     \seq_gput_right:NV
446     \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int

```



```

447     }
448   }
449   \cs_new_protected:Nn \__enumext_print_footnote:
450   {
451     \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
452     {
453       \seq_map_pairwise_function:NNN
454       \g__enumext_footnote_standar_int_seq
455       \g__enumext_footnote_standar_arg_seq
456       \__enumext_footnotetext:nn
457     }
458     \seq_gclear:N \g__enumext_footnote_standar_arg_seq
459     \seq_gclear:N \g__enumext_footnote_standar_int_seq
460   }

```

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

```

461   \cs_new_protected:Nn \__enumext_renew_footnote_mini:
462   {
463     \RenewDocumentCommand \footnote { o +m }
464     {
465       \tl_if_novalue:nTF {##1}
466       {
467         \stepcounter{footnote}
468         \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
469       }
470       {
471         \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
472       }
473       \footnotemark [ \g__enumext_footnote_starred_int ]
474       \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
475       \seq_gput_right:NV
476       \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
477     }
478   }
479   \cs_new_protected:Nn \__enumext_print_footnote_mini:
480   {
481     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
482     {
483       \seq_map_pairwise_function:NNN
484       \g__enumext_footnote_starred_int_seq
485       \g__enumext_footnote_starred_arg_seq
486       \__enumext_footnotetext:nn
487     }
488     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
489     \seq_gclear:N \g__enumext_footnote_starred_int_seq
490   }

```

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

`__enumext_renew_footnote_standar:`
`__enumext_print_footnote_standar:`
`__enumext_renew_footnote_starred:`
`__enumext_print_footnote_starred:`

We encapsulate the redefinition of `\footnote` to pass it to internal `__enumext_mini_page` environment used by the `mini-env` key in the `enumext` and `keyans` environments. We will run the redefinition when `tagged` PDF is active or when the `footnotehyper` package is not loaded.

```

491   \cs_new_protected:Nn \__enumext_renew_footnote_standar:
492   {
493     \bool_if:NT \g__enumext_standar_bool
494     {
495       \IfDocumentMetadataTF
496       {
497         \__enumext_renew_footnote:
498       }
499       {
500         \bool_if:NF \l__enumext_footnotes_key_bool
501         {
502           \__enumext_renew_footnote:
503         }
504       }
505     }
506   }
507   \cs_new_protected:Nn \__enumext_print_footnote_standar:
508   {

```

```

509 \bool_if:NT \g__enumext_standar_bool
510 {
511   \IfDocumentMetadataTF
512   {
513     \__enumext_print_footnote:
514   }
515   {
516     \bool_if:NF \l__enumext_footnotes_key_bool
517     {
518       \__enumext_print_footnote:
519     }
520   }
521 }
522 }

```

We encapsulate the redefinition of `\footnote` to pass it to the `enumext*` and `keyans*` environments. We will run the redefinition when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

523 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
524 {
525   \IfDocumentMetadataTF
526   {
527     \__enumext_renew_footnote_mini:
528   }
529   {
530     \bool_if:NF \l__enumext_footnotes_key_bool
531     {
532       \__enumext_renew_footnote_mini:
533     }
534   }
535 }
536 \cs_new_protected:Nn \__enumext_print_footnote_starred:
537 {
538   \IfDocumentMetadataTF
539   {
540     \__enumext_print_footnote_mini:
541   }
542   {
543     \bool_if:NF \l__enumext_footnotes_key_bool
544     {
545       \__enumext_print_footnote_mini:
546     }
547   }
548 }

```

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

```

549 \__enumext_after_env:nn { enumext* }
550 {
551   \__enumext_print_footnote_starred:
552 }
553 \__enumext_after_env:nn { keyans* }
554 {
555   \__enumext_print_footnote_starred:
556 }

```

(End of definition for `__enumext_renew_footnote_standar:` and others.)

13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a 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 we will add `\skip_vertical:N \c_zero_skip` 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 we redefine `\footnote` to make them look the same as if they were elsewhere in the document. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§13.42) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.47)

```

557 \cs_new_protected:Nn \__enumext_internal_mini_page:
558 {
559   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
560   {

```

```

561 \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
562 {
563     \__enumext_renew_footnote_standar:
564     \__enumext_minipage:w [ t ] { ##1 }
565     \legacy_if_gset_false:n { @minipage }
566     \skip_vertical:N \c_zero_skip
567 }
568 {
569     \skip_vertical:N \c_zero_skip
570     \__enumext_endminipage:
571     \__enumext_print_footnote_standar:
572 }
573 }
574 }

```

(End of definition for `__enumext_internal_mini_page:` and `__enumext_mini_env*`.)

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.

```

575 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

To create the necessary “counters” we must first make sure that they are not already defined by the user or a package such as `enumitem`, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

- #1:** A token list `__enumext_counter_X_tl` for “store” the counter’s name.
#2: The counter’s name.

```

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii
576 \cs_new_protected:Npn \__enumext_define_counter:Nn #1 #2
577 {
578     \cs_if_exist:cTF { c@ #2 }
579     { \msg_fatal:nnn { enumext } { counters }{ #2 } }
580     {
581         \tl_set:Nn #1 { #2 }
582         \newcounter { #2 }
583     }
584 }

```

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.

```

585 \__enumext_define_counter:Nn \__enumext_counter_i_tl { enumXi }
586 \__enumext_define_counter:Nn \__enumext_counter_ii_tl { enumXii }
587 \__enumext_define_counter:Nn \__enumext_counter_iii_tl { enumXiii }
588 \__enumext_define_counter:Nn \__enumext_counter_iv_tl { enumXiv }
589 \__enumext_define_counter:Nn \__enumext_counter_v_tl { enumXv }
590 \__enumext_define_counter:Nn \__enumext_counter_vi_tl { enumXvi }
591 \__enumext_define_counter:Nn \__enumext_counter_vii_tl { enumXvii }
592 \__enumext_define_counter:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

In version 1.6 the command `\resetenumext` (§13.27) was added which internally uses `\counterwithin*` so for its correct operation, we will create “real counters” instead of the “integer variables” for the keys `resume` and `resume*`.

```

\c@__enumext_resume_i_int
\c@__enumext_resume_ii_int
\c@__enumext_resume_iii_int
\c@__enumext_resume_iv_int
\c@__enumext_resume_vii_int
593 \cs_set_protected:Npn \__enumext_tmp:n #1
594 {
595     \cs_if_exist:cTF { c@ __enumext_resume_#1_int }
596     { \msg_fatal:nne { enumext } { counters }{ __enumext_resume_#1_int } }
597     {
598         \newcounter { __enumext_resume_#1_int }
599     }
600 }
601 \clist_map_inline:nn {i,ii,iii,iv,vii} { \__enumext_tmp:n {#1} }

```

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

13.12 Definition of labels

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

- Direct support for this is provided since L^AT_EX release 2025-06-01[13], but we will keep the original implementation so as not to hinder the internal “*label and ref*” system.

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.

```
602 \cs_new_protected:Npn \__enumext_register_default_label_wd:Nn #1 #2
603 {
604   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
605   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
606 }
607 \__enumext_register_default_label_wd:Nn \arabic { 0 }
608 \__enumext_register_default_label_wd:Nn \Alph { M }
609 \__enumext_register_default_label_wd:Nn \alph { m }
610 \__enumext_register_default_label_wd:Nn \Roman { VIII }
611 \__enumext_register_default_label_wd:Nn \roman { viii }
```

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

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

```
612 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
613 {
614   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
615   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
616 }
617 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

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

The function `__enumext_label_style:Nnn` is used by the `label` key to creates the variables containing the *⟨label style⟩* and will allow to use `\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.

```
618 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
619 {
620   \tl_clear_new:N #1
621   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
622   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
623   \tl_map_inline:Nn \g__enumext_counter_styles_tl
624   {
625     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
626     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
627     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
628   }
629   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
630   { \tl_use:N \g__enumext_widest_label_tl }
631   \tl_set_eq:cN { the #2 } #1
632 }
633 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

13.13 Setting keys associated with label

When *tagged* PDF is active `\makelabel` is redefined using `\makebox` to work correctly (§13.37). From the user side it is convenient to have a key that allows using this redefinition with `\makebox` without having `\IfDocumentMetadataTF` active.

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

```
634 \cs_set_protected:Npn \__enumext_tmp:n #1
635 {
636   \keys_define:nn { enumext / #1 }
637   {
638     mode-box .bool_set:N = \l__enumext_mode_box_bool,
639     mode-box .initial:n = false,
```

```

640     mode-box .value_forbidden:n = true,
641   }
642 }
643 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for mode-box.)

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

```

644 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
645 {
646   \keys_define:nn { enumext / #1 }
647   {
648     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
649     font .value_required:n = true,
650     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
651     labelsep .initial:n = {0.3333em},
652     labelsep .value_required:n = true,
653     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
654     labelwidth .value_required:n = true,
655     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
656     wrap-label .initial:n = {##1},
657     wrap-label .value_required:n = true,
658     wrap-label* .code:n = {
659       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
660       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
661     },
662     wrap-label* .value_required:n = true,
663   }
664 }
665 \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 we must set \l__enumext_align_label_pos_X_str.

```

666 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
667 {
668   \keys_define:nn { enumext / #1 }
669   {
670     align .choice:,
671     align / left .code:n =
672       {
673         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
674         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
675         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
676       },
677     align / right .code:n =
678       {
679         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
680         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
681         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
682       },
683     align / center .code:n =
684       {
685         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
686         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
687         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
688       },
689     align / unknown .code:n =
690       \msg_error:nneee { enumext } { unknown-choice }
691       { align } { left,~right,~ center } { \exp_not:n {##1} },
692     align .initial:n = left,
693     align .value_required:n = true,
694   }
695 }
696 \clist_map_inline:nn
697 {
698   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
699 }
700 { \__enumext_tmp:nn #1 }

```

```

701 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
702 {
703   \keys_define:nn { enumext / #1 }
704   {
705     align .choice:,
706     align / left .code:n = \str_set:cn { l__enumext_align_label#2_str } { l },
707     align / right .code:n = \str_set:cn { l__enumext_align_label#2_str } { r },
708     align / center .code:n = \str_set:cn { l__enumext_align_label#2_str } { c },
709     align / unknown .code:n =
710       \msg_error:nneee { enumext } { unknown-choice }
711       { align } { left,~right,~ center } { \exp_not:n {##1} },
712     align .initial:n = left,
713     align .value_required:n = true,
714   }
715 }
716 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

13.14 Setting label and ref keys

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

13.14.1 Define and set label and ref keys for enumext environment

Here we set the default `<labels>` of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
717 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
718 {
719   \keys_define:nn { enumext / #1 }
720   {
721     label .code:n = {
722       \__enumext_label_style:cvn { l__enumext_label#2_tl }
723       { l__enumext_counter#2_tl } {##1}
724       \dim_set_eq:cN { l__enumext_labelwidth#2_dim }
725       \l__enumext_current_widest_dim
726     },
727     label .initial:n = #3,
728     label .value_required:n = true,
729     ref .code:n = \__enumext_standar_ref:n {##1},
730     ref .value_required:n = true,
731   }
732 }
733 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
734 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
735 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
736 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

The `__enumext_standard_ref:n` 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 of the *current counter style* save 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`.

```

737 \cs_new_protected:Npn \__enumext_standar_ref:n #1
738 {
739   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
740   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
741   {
742     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
743   }
744   {
745     \tl_set_eq:Nc \l__enumext_ref_the_count_tl
746     {
747       l__enumext_the_counter_ \__enumext_level: _tl
748     }
749     \tl_set:ce { l__enumext_renew_counter_ \__enumext_level: _tl }
750     {
751       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
752       { \exp_not:V \l__enumext_ref_key_arg_tl }

```



```

753     }
754   }
755 }

```

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

```

756 \cs_new_protected:Nn \__enumext_standar_ref:
757 {
758   \tl_if_empty:cF { \__enumext_renew_counter_ \__enumext_level: _tl }
759   {
760     \tl_use:c { \__enumext_renew_counter_ \__enumext_level: _tl }
761   }
762 }

```

(End of definition for `__enumext_standar_ref:n` and `__enumext_standar_ref:`.)

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

Here we set the default `<labels>` for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key and `ref` key.

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
763 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
764 {
765   \keys_define:nn { enumext / #1 }
766   {
767     label .code:n = {
768       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
769       { \__enumext_counter_#2_tl } {##1}
770       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
771       \__enumext_current_widest_dim
772     },
773     label .initial:n = #3,
774     label .value_required:n = true,
775     ref .code:n = \__enumext_starred_ref:n {##1},
776     ref .value_required:n = true,
777   }
778 }
779 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
780 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*) }

```

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

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

```

\__enumext_starred_ref:n
\__enumext_starred_ref:n
781 \cs_new_protected:Npn \__enumext_starred_ref:n #1
782 {
783   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
784   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
785   {
786     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
787     {
788       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
789     }
790     {
791       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
792       \tl_set:Ne \l__enumext_renew_counter_vii_tl
793       {
794         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
795       }
796     }
797   }
798   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
799   {
800     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
801     {
802       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
803     }
804     {
805       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
806       \tl_set:Ne \l__enumext_renew_counter_viii_tl
807       {
808         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
809       }
810     }
811   }

```

```

811     }
812 }

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.

813 \cs_new_protected:Nn \__enumext_starred_ref:
814 {
815   \int_compare:nNt { \__enumext_level_h_int } = { 1 }
816   {
817     \tl_if_empty:NF \__enumext_renew_counter_vii_tl
818     {
819       \tl_use:N \__enumext_renew_counter_vii_tl
820     }
821   }
822   \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
823   {
824     \tl_if_empty:NF \__enumext_renew_counter_viii_tl
825     {
826       \tl_use:N \__enumext_renew_counter_viii_tl
827     }
828   }
829 }

```

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

13.14.3 Define and set label and ref keys for keyans and keyanspic environments

Here we set the default `<label>` for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` if it has not been established and `ref` key. The `keyanspic` environment use the same `<label>` as the `keyans` environment.

```

\__enumext_label_v_tl
\__enumext_label_vi_tl

830 \keys_define:nn { enumext / keyans }
831 {
832   label .code:n = {
833     \__enumext_label_style:cnv { \__enumext_label_v_tl }
834     { \__enumext_counter_v_tl } {#1}
835     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
836     { \__enumext_counter_vi_tl } {#1}
837     \dim_set_eq:NN
838     \__enumext_labelwidth_v_dim \__enumext_current_widest_dim
839   },
840   label .initial:n = \Alph*,
841   label .value_required:n = true,
842   ref .code:n = \__enumext_keyans_ref:n {#1},
843   ref .value_required:n = true,
844 }

```

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

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

```

845 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
846 {
847   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
848   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
849   {
850     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
851   }
852   {
853     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
854     \tl_put_right:Ne \__enumext_renew_counter_v_tl
855     {
856       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl } { \exp_not:V \l_
857     }
858   }
859 }

```

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

```

860 \cs_new_protected:Nn \__enumext_keyans_ref:
861 {
862   \tl_if_empty:NF \__enumext_renew_counter_v_tl
863   {
864     \tl_use:N \__enumext_renew_counter_v_tl
865   }

```

```
866 }
```

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

13.15 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 take three arguments:

```
#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: <integer or string>
```

The first argument of this function are the “counter style” set by `label` key, the second argument is returned by the function, the third argument 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 we must set the boolean variable `\l__enumext_resume_count_bool` set by the `resume` key *without value* to “false”. This is necessary to be able to “reset” the *start value* by means of the `start` or `start*` keys.

```
867 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
868 {
869   \bool_set_false:N \l__enumext_resume_count_bool
870   \__enumext_if_is_int:nTF { #3 }
871   {
872     \int_set:Nn #2 {#3}
873   }
874   {
875     \regex_if_match:nVT { \c{Alph} | \c{alph} } {#1}
876     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
877     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#1}
878     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
879   }
880 }
881 \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 take 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 second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an *<integer>* or *<string>* of the form `\Alph`, `\alph`, `\Roman` or `\roman`. The value of the four argument 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.

```
882 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
883 {
884   \__enumext_if_is_int:nTF {#4}
885   {
886     \setcounter{enumX#1} { #4 }
887   }
888   {
889     \regex_if_match:nVT { \c{Alph} | \c{alph} } {#2}
890     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
891     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#2}
892     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
893   }
894   \__enumext_label_width_by_box:cv
895   { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
896 }
897 \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.

```
898 \cs_set_protected:Npn \__enumext_tmp:n #1 #2
899 {
900   \keys_define:nn { enumext / #1 }
901   {
902     start* .code:n = {
903       \__enumext_start_from:ccn
```

```

904             { l__enumext_label_#2_tl }
905             { l__enumext_start_#2_int } {##1}
906         },
907         start* .value_required:n = true,
908         start .code:n = {
909             \__enumext_start_from:cce
910             { l__enumext_label_#2_tl }
911             { l__enumext_start_#2_int } { \int_eval:n {##1} }
912         },
913         start .initial:n = 1,
914         start .value_required:n = true,
915         widest .code:n = {
916             \__enumext_widest_from:nccn {#2}
917             { l__enumext_label_#2_tl }
918             { l__enumext_labelwidth_#2_dim } {##1}
919         },
920         widest .value_required:n = true,
921     }
922 }
923 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.16 Setting keys for penaltys

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 a *keys* to access these.

```

924 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
925 {
926     \keys_define:nn { enumext / #1 }
927     {
928         beginpenalty .int_set:c = { l__enumext_beginparpenalty_#2_int },
929         beginpenalty .initial:n = -51,
930         beginpenalty .value_required:n = true,
931         midpenalty .int_set:c = { l__enumext_itempenalty_#2_int },
932         midpenalty .initial:n = -51,
933         midpenalty .value_required:n = true,
934         endpenalty .int_set:c = { l__enumext_endparpenalty_#2_int },
935         endpenalty .initial:n = -51,
936         endpenalty .value_required:n = true,
937     }
938 }
939 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.17 Setting keys for vertical spaces

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

```

940 \cs_set_protected:Npn \__enumext_tmp:nnnnn #1 #2 #3 #4 #5 #6
941 {
942     \keys_define:nn { enumext / #1 }
943     {
944         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
945         topsep .initial:n = {#3},
946         topsep .value_required:n = true,
947         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
948         partopsep .initial:n = {#4},
949         partopsep .value_required:n = true,
950         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
951         parsep .initial:n = {#5},
952         parsep .value_required:n = true,
953         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
954         itemsep .initial:n = {#6},
955         itemsep .value_required:n = true,
956         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
957         noitemsep .value_forbidden:n = true,
958         nosep .meta:n = {
959             itemsep = 0pt, parsep = 0pt,
960             topsep = 0pt, partopsep = 0pt,

```

```

961         },
962         nosep      .value_forbidden:n = true,
963     }
964 }

```

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

```

965 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
966   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
967   { 4.0pt plus 2.0pt minus 1.0pt }
968 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
969   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
970   { 2.0pt plus 1.0pt minus 1.0pt }
971 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
972   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
973 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
974   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
975 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
976   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
977   { 2.0pt plus 1.0pt minus 1.0pt }
978 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
979   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
980   { 4.0pt plus 2.0pt minus 1.0pt }
981 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
982   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
983   { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.18 Setting base-fix key

When nesting starting right after `\item` (without material between them) there is a problem with the alignment of the *baseline* between the two environments. One way to get around this problem is to place `\mode_leave_vertical:` apply `\vspace{-\baselineskip}` and set `\topsep=0pt` for the “first level” of the nested `enumext` environment.

```

base-fix \__enumext_nested_base_line_fix:
984 \keys_define:nn { enumext / level-1 }
985   {
986     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
987     base-fix .initial:n = false,
988     base-fix .value_forbidden:n = true,
989   }

```

The function `__enumext_nested_base_line_fix:` passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.42) will be responsible for applying the *baseline correction* and adjusting the *keys* for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§13.50).

We will first implement the function code 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.

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

```

990 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
991   {
992     \bool_lazy_all:nT
993       {
994         { \bool_if_p:N \l__enumext_starred_first_bool }
995         { \bool_if_p:N \l__enumext_base_line_fix_bool }
996         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
997       }
998     {
999       \mode_leave_vertical:
1000       \vspace { -\dim_eval:n { \baselineskip + \parsep } }
1001       \keys_set:nn { enumext / level-1 }
1002         {
1003           topsep = 0pt, above = 0pt, above* = 0pt,
1004         }
1005     }

```

When we are running the `\printkeyans` command with the *starred argument* ‘*’ the variable `\l__enumext_print_keyans_star_bool` is true and we can run a simplified version of `\vspace` using `\skip_vertical:n`.

```

1006   \bool_lazy_and:nnT
1007   { \bool_if_p:N \l__enumext_starred_first_bool }
1008   { \bool_if_p:N \l__enumext_print_keyans_star_bool }
1009   {
1010     \mode_leave_vertical:
1011     \skip_vertical:n { -\baselineskip }
1012     \skip_vertical:N \c_zero_skip
1013     \keys_set:nn { enumext / level-1 }
1014     {
1015       topsep = 0pt, above = 0pt, above* = 0pt,
1016     }
1017   }
1018   \bool_set_false:N \l__enumext_base_line_fix_bool
1019 }

```

(End of definition for `base-fix` and `\l__enumext_nested_base_line_fix:`.)

13.19 Setting keys for horizontal spaces

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

```

1020 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1021 {
1022   \keys_define:nn { enumext / #1 }
1023   {
1024     itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
1025     itemindent .value_required:n = true,
1026     rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
1027     rightmargin .value_required:n = true,
1028     listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
1029     listparindent .value_required:n = true,
1030     list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
1031     list-offset .value_required:n = true,
1032     list-indent .code:n =
1033       \bool_set_true:c { \l__enumext_leftmargin_tmp_#2_bool }
1034       \dim_set:cn { \l__enumext_leftmargin_tmp_#2_dim } {##1},
1035     list-indent .value_required:n = true,
1036   }
1037 }
1038 \clist_map_inline:nn
1039 {
1040   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1041 }
1042 { \l__enumext_tmp:nn #1 }

```

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

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

```

1043 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1044 {
1045   \keys_define:nn { enumext / #1 }
1046   {
1047     itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
1048     itemindent .value_required:n = true,
1049     rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
1050     rightmargin .value_required:n = true,
1051     listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
1052     listparindent .value_required:n = true,
1053     list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
1054     list-offset .value_required:n = true,
1055     list-indent .meta:n = { list-offset = ##1 },
1056     list-indent .value_required:n = true,
1057   }
1058 }
1059 \clist_map_inline:nn
1060 {
1061   {enumext*}{vii}, {keyans*}{viii}
1062 }
1063 { \l__enumext_tmp:nn #1 }

```


13.19.1 Functions for setting the fake itemindent

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

```

1064 \cs_set_protected:Nn \__enumext_fake_item_indent:
1065 {
1066   \dim_compare:nNnT
1067   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1068   >
1069   { \c_zero_dim }
1070   {
1071     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1072     {
1073       \exp_not:N \mode_leave_vertical:
1074       \exp_not:n { \skip_horizontal:n }
1075       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1076       \exp_not:N \ignorespaces
1077     }
1078   }
1079 }
1080 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1081 {
1082   \dim_compare:nNnT
1083   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1084   {
1085     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1086     {
1087       \exp_not:N \mode_leave_vertical:
1088       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1089       \exp_not:N \ignorespaces
1090     }
1091   }
1092 }
1093 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1094 {
1095   \dim_compare:nNnT
1096   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1097   {
1098     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1099     {
1100       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1101       \exp_not:N \ignorespaces
1102     }
1103   }
1104 }
1105 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1106 {
1107   \dim_compare:nNnT
1108   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1109   {
1110     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1111     {
1112       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1113       \exp_not:N \ignorespaces
1114     }
1115   }
1116 }

```

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

13.20 Setting show-length key

show-length

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

```

1117 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1118 {
1119   \keys_define:nn { enumext / #1 }
1120   {

```

```

1121         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1122         show-length .initial:n = false,
1123     }
1124 }
1125 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for show-length.)

13.21 Setting before, after and first keys

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

```

before* 1126 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
after    1127 {
first    1128     \keys_define:nn { enumext / #1 }
          1129     {
          1130         before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
          1131         before .value_required:n = true,
          1132         before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
          1133         before* .value_required:n = true,
          1134         after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
          1135         after .value_required:n = true,
          1136         first .tl_set:c = { l__enumext_after_list_args_#2_tl },
          1137         first .value_required:n = true,
          1138     }
          1139 }
1140 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for before and others.)

13.21.1 Functions for before, after and first keys in enumext

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

```

__enumext_before_args_exec: 1141 \cs_new_protected:Nn \__enumext_before_args_exec:
                             1142 {
                             1143     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
                             1144 }

```

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 definition and values provides by `⟨keys⟩: \list{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}`

```

1145 \cs_new_protected:Nn \__enumext_before_keys_exec:
1146 {
1147     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1148 }

```

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

```

1149 \cs_new_protected:Nn \__enumext_after_stop_list:
1150 {
1151     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1152 }

```

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.`

```

1153 \cs_new_protected:Nn \__enumext_after_args_exec:
1154 {
1155     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1156 }

```

(End of definition for __enumext_before_args_exec: and others.)

13.21.2 Functions for before, after and first keys in keyans

Same implementation as the one used in the `enumext` environment.

```

__enumext_before_args_exec_v: 1157 \cs_new_protected:Nn \__enumext_before_args_exec_v:
__enumext_before_keys_exec_v: 1158 {
__enumext_after_stop_list_v: 1159     \tl_use:N \l__enumext_before_starred_key_v_tl
__enumext_after_args_exec_v: 1160 }
                             1161 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
                             1162 {

```

```

1163     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1164   }
1165   \cs_new_protected:Nn \__enumext_after_stop_list_v:
1166   {
1167     \tl_use:N \l__enumext_after_stop_list_v_tl
1168   }
1169   \cs_new_protected:Nn \__enumext_after_args_exec_v:
1170   {
1171     \tl_use:N \l__enumext_after_list_args_v_tl
1172   }

```

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

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

Same implementation as the one used in the `enumext` environment.

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1173 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1174 {
1175   \tl_use:N \l__enumext_before_starred_key_vii_tl
1176 }
1177 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1178 {
1179   \tl_use:N \l__enumext_before_starred_key_viii_tl
1180 }
1181 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1182 {
1183   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1184 }
1185 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1186 {
1187   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1188 }
1189 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1190 {
1191   \tl_use:N \l__enumext_after_stop_list_vii_tl
1192 }
1193 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1194 {
1195   \tl_use:N \l__enumext_after_stop_list_viii_tl
1196 }
1197 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1198 {
1199   \tl_use:N \l__enumext_after_list_args_vii_tl
1200 }
1201 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1202 {
1203   \tl_use:N \l__enumext_after_list_args_viii_tl
1204 }

```

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

13.22 Setting keys for multicols and minipage

The default value of the `columns-sep` key is handled by the state of the boolean variable `\l__enumext_columns_sep_X_bool` which is handled in the internal definition of the `enumext` and `keyans` environments. Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1205 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1206 {
1207   \keys_define:nn { enumext / #1 }
1208   {
1209     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1210     mini-env .value_required:n = true,
1211     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1212     mini-sep .initial:n = 0.3333em,
1213     mini-sep .value_required:n = true,
1214     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1215     columns-sep .value_required:n = true,
1216     columns .int_set:c = { l__enumext_columns_#2_int },
1217     columns .initial:n = 1,
1218     columns .value_required:n = true,
1219   }

```

```

1220 }
1221 \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 we will add the keys mini-right and mini-right* to implement support for minipage
environment.

1222 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1223 {
1224   \keys_define:nn { enumext / #1 }
1225   {
1226     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1227     mini-right .value_required:n = true,
1228     mini-right* .code:n = {
1229       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1230       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1231     },
1232     mini-right* .value_required:n = true,
1233   }
1234 }
1235 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.23 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 it can be seen like in the figure 7.

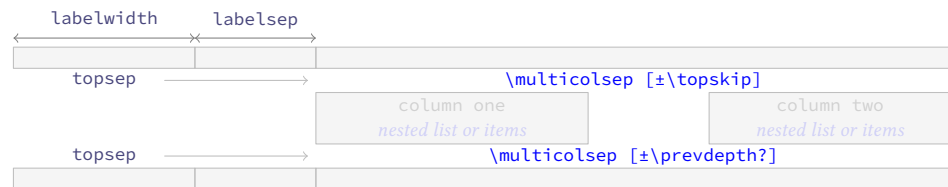


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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`.

🌱 I should make it 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.

13.23.1 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” that we will apply “above” and “below” the `multicols` environment in `enumext`.

We will set the default values taking into account that T_EX is in *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

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

```

1236 \cs_new_protected:Npn \__enumext_multi_set_vskip:
1237 {
1238   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1239   {
1240     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1241   }
1242   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1243   {
1244     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1245   }
1246   \__enumext_add_pre_parsep:
1247 }

```

(End of definition for `__enumext_multi_set_vskip:`.)

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

```

1248 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1249 {
1250   \int_case:nn { \l__enumext_level_int }
1251   {
1252     { 2 }{
1253       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1254       {
1255         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1256         {
1257           \l__enumext_parsep_i_skip
1258         }
1259       }
1260     }
1261     { 3 }{
1262       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1263       {
1264         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1265         {
1266           \l__enumext_parsep_ii_skip
1267         }
1268       }
1269     }
1270     { 4 }{
1271       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1272       {
1273         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1274         {
1275           \l__enumext_parsep_iii_skip
1276         }
1277       }
1278     }
1279   }
1280 }

```

(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*.

```

1281 \cs_new_protected:Nn \__enumext_multi_addvspace:
1282 {
1283   \__enumext_multi_set_vskip:
1284   \mode_if_vertical:T
1285   {
1286     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1287     {
1288       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1289     }
1290     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1291     {
1292       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1293     }
1294   }
1295   \par\nopagebreak
1296   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1297 }

```

(End of definition for `__enumext_multi_addvspace:`)

13.23.2 Adjustment of vertical spaces for multicols in keyans

`__enumext_keyans_multi_set_vskip:` The function `__enumext_keyans_multi_set_vskip:` will take care of determining the “adjusted spaces” that we will apply “above” and “below” the `multicols` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

`__enumext_keyans_multi_addvspace:`

```

1298 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1299 {
1300   \skip_set:Nn \l__enumext_multicols_above_v_skip
1301   {
1302     \l__enumext_topsep_v_skip
1303   }
1304   \skip_set:Nn \l__enumext_multicols_below_v_skip
1305   {

```

```

1306     \l__enumext_topsep_v_skip
1307   }
1308 }
1309 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1310 {
1311   \__enumext_keyans_multi_set_vskip:
1312   \mode_if_vertical:T
1313   {
1314     \skip_add:Nn \l__enumext_multicols_above_v_skip
1315     {
1316       \skip_use:N \l__enumext_partopsep_v_skip
1317     }
1318     \skip_add:Nn \l__enumext_multicols_below_v_skip
1319     {
1320       \skip_use:N \l__enumext_partopsep_v_skip
1321     }
1322   }
1323   \par\nopagebreak
1324   \addvspace{ \l__enumext_multicols_above_v_skip }
1325 }

```

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

13.24 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.

Since we want to keep the “left” and “right” environments “aligned on top”, preserving the `\baselineskip` and keep the desired “spaces” (`\topsep` + `[\partopsep]`) it is necessary to “adjust” the “vertical spaces” for `minipage` environments.

Here there are several complications that we must circumvent, 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, small adjustments must be made using `\vspace` and `\addvspace` to obtain the “desired vertical spacing”.

Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `minipage` environment is quite complicated, even more if `multicols` is nested. The setting of the values was more “trial and error” (approx to `\strutbox`), using the help of the `lua-visual-debug`[15] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

13.24.1 Adjustment of vertical spaces for minipage in enumext

```

\__enumext_minipage_set_skip:
\__enumext_minipage_add_space:

```

The function `__enumext_minipage_set_skip:` will take care of determining the “adjust” spaces that we will apply “above” and “below” the `__enumext_mini_page` environment in `enumext`.

First we will set the value of `\l__enumext_minipage_right_skip` equal to `\topsep`, then we will see if \TeX is in *vertical mode* and we will add `\partopsep`, followed by that we set the value of `\l__enumext_minipage_after_skip`.

```

1326 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1327 {
1328   \skip_set:Nn \l__enumext_minipage_right_skip
1329   {
1330     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1331   }
1332   \mode_if_vertical:T
1333   {
1334     \skip_add:Nn \l__enumext_minipage_right_skip
1335     {
1336       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1337     }
1338   }
1339   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip

```


We will adjust the values `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` and call the function `__enumext_pre_itemsep_skip:`.

```

1340 \skip_set_eq:cN
1341 { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1342 \skip_set_eq:cN
1343 { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1344 \__enumext_pre_itemsep_skip:

```

If the environment `multicols` is active, we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `\l__enumext_multicols_above_X_skip`.

```

1345 \int_compare:nNtT
1346 { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1347 {
1348   \skip_zero:N \topskip
1349   \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1350 }
1351 }

```

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*. Here we use the plain \TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1352 \cs_new_protected:Nn \__enumext_minipage_add_space:
1353 {
1354   \__enumext_minipage_set_skip:
1355   \__enumext_unskip_unkern:
1356   \mode_if_vertical:TF
1357   {
1358     \nopagebreak\nointerlineskip
1359   }
1360   {
1361     \par\nopagebreak\nointerlineskip
1362     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1363   }
1364   \int_compare:nNtTF
1365   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1366   {
1367     \addvspace{ 0.445\box_ht:N \strutbox }
1368   }
1369   {
1370     \addvspace{ 0.250\box_ht:N \strutbox }
1371   }
1372 }

```

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

`__enumext_pre_itemsep_skip:`

The function `__enumext_pre_itemsep_skip:` will adjust the spaces below the environment `minipage` and the environment `multicols` if it is nested in it, taking into account the value of `\itemsep` from the previous level.

```

1373 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1374 {
1375   \int_case:nn { \l__enumext_level_int }
1376   {
1377     { 2 }{
1378       \skip_if_eq:nnTF
1379       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1380       {
1381         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1382         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1383       }
1384       {
1385         \dim_compare:nNtT
1386         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1387         {
1388           \skip_sub:Nn
1389           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1390           \skip_sub:Nn
1391           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1392           \skip_add:Nn
1393           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1394         \skip_add:Nn
1395         \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1396     }
1397 \dim_compare:nNnT
1398 { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1399 {
1400     \skip_set:Nn \l__enumext_minipage_temp_skip
1401     {
1402         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1403     }
1404     \skip_sub:Nn
1405     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1406     \skip_sub:Nn
1407     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1408     \skip_add:Nn
1409     \l__enumext_minipage_after_skip
1410     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1411     \skip_add:Nn
1412     \l__enumext_multicols_below_ii_skip
1413     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1414 }
1415 }
1416 }
1417 { 3 }{
1418     \skip_if_eq:nnTF
1419     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1420     {
1421         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1422         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1423     }
1424     {
1425         \dim_compare:nNnT
1426         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1427         {
1428             \skip_sub:Nn
1429             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1430             \skip_sub:Nn
1431             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1432             \skip_add:Nn
1433             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1434             \skip_add:Nn
1435             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1436         }
1437         \dim_compare:nNnT
1438         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1439         {
1440             \skip_set:Nn \l__enumext_minipage_temp_skip
1441             {
1442                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1443             }
1444             \skip_sub:Nn
1445             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1446             \skip_sub:Nn
1447             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1448             \skip_add:Nn
1449             \l__enumext_minipage_after_skip
1450             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1451             \skip_add:Nn
1452             \l__enumext_multicols_below_iii_skip
1453             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1454         }
1455     }
1456 }
1457 { 4 }{
1458     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1459     {
1460         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1461         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1462     }
1463     {
1464         \dim_compare:nNnT

```

```

1465         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1466     {
1467         \skip_sub:Nn
1468         \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1469         \skip_sub:Nn
1470         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1471         \skip_add:Nn
1472         \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1473         \skip_add:Nn
1474         \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1475     }
1476 \dim_compare:nNnT
1477 { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1478 {
1479     \skip_set:Nn \l__enumext_minipage_temp_skip
1480     {
1481         \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1482     }
1483     \skip_sub:Nn
1484     \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1485     \skip_sub:Nn
1486     \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1487     \skip_add:Nn
1488     \l__enumext_minipage_after_skip
1489     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1490     \skip_add:Nn
1491     \l__enumext_multicols_below_iv_skip
1492     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1493 }
1494 }
1495 }
1496 }
1497 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

13.24.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

The function `__enumext_keyans_mini_set_vskip:` will take care of determining the “adjusted” spaces that we will apply “*above*” and “*below*” the `__enumext_mini_page` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

```

1498 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1499 {
1500     \skip_zero:N \l__enumext_minipage_after_skip
1501     \skip_zero:N \l__enumext_minipage_left_skip
1502     \skip_zero:N \l__enumext_minipage_right_skip
1503     \skip_set:Nn \l__enumext_minipage_right_skip
1504     {
1505         \l__enumext_topsep_v_skip
1506     }
1507     \mode_if_vertical:T
1508     {
1509         \skip_add:Nn \l__enumext_minipage_right_skip
1510         {
1511             \l__enumext_partopsep_v_skip
1512         }
1513     }
1514     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1515     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1516     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1517     \__enumext_keyans_pre_itemsep_skip:
1518     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1519     {
1520         \skip_zero:N \topskip
1521         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1522     }
1523 }
1524 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1525 {
1526     \__enumext_keyans_minipage_set_skip:
1527     \__enumext_unskip_unkern:
1528     \mode_if_vertical:TF

```

```

1529     {
1530         \nopagebreak\nointerlineskip
1531     }
1532     {
1533         \par\nopagebreak\nointerlineskip
1534         \skip_zero:N \l__enumext_partopsep_v_skip
1535     }
1536     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1537     {
1538         \addvspace{ 0.445\box_ht:N \strutbox }
1539     }
1540     {
1541         \addvspace{ 0.250\box_ht:N \strutbox }
1542     }
1543 }
1544 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1545 {
1546     \skip_if_eq:nnTF
1547     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1548     {
1549         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1550         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1551     }
1552     {
1553         \dim_compare:nNnT
1554         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1555         {
1556             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1557             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1558             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1559             \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1560         }
1561         \dim_compare:nNnT
1562         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1563         {
1564             \skip_set:Nn \l__enumext_minipage_temp_skip
1565             {
1566                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1567             }
1568             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1569             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1570             \skip_add:Nn \l__enumext_minipage_after_skip
1571             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1572             \skip_add:Nn \l__enumext_multicols_below_v_skip
1573             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1574         }
1575     }
1576 }

```

(End of definition for `__enumext_keyans_minipage_set_skip:`, `__enumext_keyans_minipage_add_space:`, and `__enumext_keyans_pre_itemsep_skip:`.)

13.24.3 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 “adjusted” spaces that we will apply “above” and “below” the `__enumext_mini_page` environment in `enumext*` and `keyans*`.

```

1577 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1578 {
1579     \skip_zero_new:N \l__enumext_minipage_left_skip
1580     \skip_gzero_new:N \g__enumext_minipage_right_skip
1581     \skip_gzero_new:N \g__enumext_minipage_after_skip
1582     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1583     {
1584         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1585         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1586     }
1587     {
1588         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1589         \skip_gset:Nn \g__enumext_minipage_right_skip
1590         {
1591             \l__enumext_topsep_vii_skip

```

```

1592     }
1593     \skip_gset:Nn \g__enumext_minipage_after_skip
1594     {
1595         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1596     }
1597 }
1598 }
1599 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1600 {
1601     \skip_zero_new:N \l__enumext_minipage_after_skip
1602     \skip_zero_new:N \l__enumext_minipage_left_skip
1603     \skip_zero_new:N \l__enumext_minipage_right_skip
1604     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1605     {
1606         \skip_set:Nn \l__enumext_minipage_left_skip
1607         {
1608             0.5\box_dp:N \strutbox
1609         }
1610         \skip_set:Nn \l__enumext_minipage_right_skip
1611         {
1612             \l__enumext_partopsep_viii_skip
1613         }
1614         \skip_set:Nn \l__enumext_minipage_after_skip
1615         {
1616             1.6\box_dp:N \strutbox
1617         }
1618     }
1619     {
1620         \skip_set:Nn \l__enumext_minipage_left_skip
1621         {
1622             0.5875\box_dp:N \strutbox
1623         }
1624         \skip_set:Nn \l__enumext_minipage_right_skip
1625         {
1626             \l__enumext_topsep_viii_skip
1627         }
1628         \skip_set:Nn \l__enumext_minipage_after_skip
1629         {
1630             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1631         }
1632     }
1633 }

```

(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.

Here we will NOT take into account whether TeX is in *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1634 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1635 {
1636     \__enumext_mini_set_vskip_vii:
1637     \par\nopagebreak
1638     \addvspace { \l__enumext_minipage_left_skip }
1639 }
1640 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1641 {
1642     \__enumext_mini_set_vskip_viii:
1643     \par\nopagebreak
1644     \addvspace { \l__enumext_minipage_left_skip }
1645 }

```

(End of definition for `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:`)

13.24.4 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 we will add `\centering` when starting the “right side” environment. The *starred argument* ‘***’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `__enumext_mini_page` on the “right side”.

`\miniright` First we will perform some checks to prevent the command from being executed outside the `enumext` environment or somewhere inappropriate then we will call the internal functions to execute it in the `enumext` and `keyans` environments.

```

1646 \NewDocumentCommand \miniright { s }
1647 {
1648   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1649   {
1650     \msg_error:nnn { enumext } { wrong-miniright-place }
1651   }
1652   % outside
1653   \bool_lazy_and:nnT
1654   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1655   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1656   {
1657     \msg_error:nnn { enumext } { wrong-miniright-place }
1658   }
1659   % starred env
1660   \bool_lazy_and:nnT
1661   { \bool_if_p:N \g__enumext_starred_bool }
1662   { \bool_not_p:n { \__enumext_standar_bool } }
1663   {
1664     \msg_error:nnn { enumext } { wrong-miniright-starred }
1665   }
1666   % exec
1667   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1668   {
1669     \__enumext_keyans_mini_right_cmd:n {#1}
1670   }
1671   { \__enumext_mini_right_cmd:n {#1} }
1672 }

```

(End of definition for `\miniright`. This function is documented on page 12.)

`__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. We check if the `mini-env` key is active via the variable `__enumext_minipage_right_X_dim`, if so we close the `multicols` environment with the `__enumext_mini_page` environment on the “left side”, then we open the `__enumext_mini_page` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the *starred argument* ‘`*`’ is not present and set zero `__enumext_minipage_stat_int`, otherwise we return an error.

```

1673 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1674 {
1675   \dim_compare:nNtF
1676   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1677   {
1678     \__enumext_multicols_stop:
1679     \int_compare:nNt
1680     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } = { 1 }
1681     {
1682       \par\addvspace{ \__enumext_minipage_after_skip }
1683     }
1684     \end__enumext_mini_page
1685     \hfill
1686     \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1687     \par\nointerlineskip
1688     \addvspace { \__enumext_minipage_right_skip }
1689     \bool_if:nF {#1}
1690     {
1691       \centering
1692     }
1693     \int_gzero:N \__enumext_minipage_stat_int
1694   }
1695   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1696   % paranoia
1697   \RenewDocumentCommand \miniright { s }
1698   {
1699     \msg_error:nn { enumext } { many-miniright-used }
1700   }
1701 }

```

(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.

```

1702 \cs_new_protected:Npn \\enumext_keyans_mini_right_cmd:n #1
1703 {
1704   \dim_compare:nNnTF { \\enumext_minipage_right_v_dim } > { \c_zero_dim }
1705   {
1706     \\enumext_keyans_multicols_stop:
1707     \int_compare:nNnT { \\enumext_columns_v_int } = { 1 }
1708     {
1709       \par\addvspace{ \\enumext_minipage_after_skip }
1710     }
1711     \end__enumext_mini_page
1712     \hfill
1713     \\enumext_mini_page{ \\enumext_minipage_right_v_dim }
1714     \par\nointerlineskip
1715     \addvspace { \\enumext_minipage_right_skip }
1716     \bool_if:nF {#1}
1717     {
1718       \centering
1719     }
1720     \int_gzero:N \g__enumext_minipage_stat_int
1721   }
1722   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1723   % paranoia
1724   \RenewDocumentCommand \miniright { s }
1725   {
1726     \msg_error:nn { enumext } { many-miniright-used }
1727   }
1728 }

```

(End of definition for \\enumext_keyans_mini_right_cmd:n.)

13.25 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. As I have not been able to correct these *glitches*, the best option is to leave a couple of *⟨keys⟩* dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

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

```

above* 1729 \cs_set_protected:Npn \\enumext_tmp:nn #1 #2
below 1730 {
below* 1731   \keys_define:nn { enumext / #1 }
1732   {
1733     above .skip_set:c = { \\enumext_vspace_above_#2_skip },
1734     above .value_required:n = true,
1735     above* .code:n = \bool_set_true:c { \\enumext_vspace_a_star_#2_bool }
1736               \keys_set:nn { enumext / #1 } { above = {##1} },
1737     above* .value_required:n = true,
1738     below .skip_set:c = { \\enumext_vspace_below_#2_skip },
1739     below .value_required:n = true,
1740     below* .code:n = \bool_set_true:c { \\enumext_vspace_b_star_#2_bool }
1741               \keys_set:nn { enumext / #1 } { below = {##1} },
1742     below* .value_required:n = true,
1743   }
1744 }
1745 \clist_map_inline:Nn \c__enumext_all_envs_clist { \\enumext_tmp:nn #1 }

```

(End of definition for above and others.)

13.25.1 Functions for above and below keys in enumext

\\enumext_vspace_above:

The function \\enumext_vspace_above: apply the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```

1746 \cs_new_protected:Nn \\enumext_vspace_above:
1747 {
1748   \skip_if_eq:nnF
1749   { \skip_use:c { \\enumext_vspace_above_ \\enumext_level: _skip } } { \c_zero_skip }
1750   {
1751     \bool_if:cTF { \\enumext_vspace_a_star_ \\enumext_level: _bool }

```

```

1752         {
1753             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1754         }
1755     {
1756         \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1757     }
1758 }
1759 }

```

(End of definition for __enumext_vspace_above:.)

__enumext_vspace_below: The function __enumext_vspace_below: apply the *vertical space below* the `enumext` environment set by the `below*` and `below` keys.

```

1760 \cs_new_protected:Nn \__enumext_vspace_below:
1761 {
1762     \skip_if_eq:nnF
1763     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1764     {
1765         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1766         {
1767             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1768         }
1769         {
1770             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1771         }
1772     }
1773 }

```

(End of definition for __enumext_vspace_below:.)

13.25.2 Functions for above and below keys in keyans

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

```

1774 \cs_new_protected:Nn \__enumext_vspace_above_v:
1775 {
1776     \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1777     {
1778         \bool_if:NTF \__enumext_vspace_a_star_v_bool
1779         {
1780             \vspace*{ \__enumext_vspace_above_v_skip }
1781         }
1782         { \vspace { \__enumext_vspace_above_v_skip } }
1783     }
1784 }

```

(End of definition for __enumext_vspace_above_v:.)

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

```

1785 \cs_new_protected:Nn \__enumext_vspace_below_v:
1786 {
1787     \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1788     {
1789         \bool_if:NTF \__enumext_vspace_b_star_v_bool
1790         {
1791             \vspace*{ \__enumext_vspace_below_v_skip }
1792         }
1793         { \vspace { \__enumext_vspace_below_v_skip } }
1794     }
1795 }

```

(End of definition for __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii: 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.

__enumext_vspace_above_viii:

```

1796 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1797 {
1798     \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1799     {
1800         \bool_if:NTF \__enumext_vspace_a_star_vii_bool

```

```

1801         {
1802             \vspace*{ \l__enumext_vspace_above_vii_skip }
1803         }
1804         { \vspace { \l__enumext_vspace_above_vii_skip } }
1805     }
1806 }
1807 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1808 {
1809     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1810     {
1811         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1812         {
1813             \vspace*{ \l__enumext_vspace_above_viii_skip }
1814         }
1815         { \vspace { \l__enumext_vspace_above_viii_skip } }
1816     }
1817 }

```

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

`__enumext_vspace_below_vii:` 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.

`__enumext_vspace_below_viii:`

```

1818 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1819 {
1820     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1821     {
1822         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1823         {
1824             \vspace*{ \l__enumext_vspace_below_vii_skip }
1825         }
1826         { \vspace { \l__enumext_vspace_below_vii_skip } }
1827     }
1828 }
1829 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1830 {
1831     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1832     {
1833         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1834         {
1835             \vspace*{ \l__enumext_vspace_below_viii_skip }
1836         }
1837         { \vspace { \l__enumext_vspace_below_viii_skip } }
1838     }
1839 }

```

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

13.26 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⟩*. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in `chat-Tex-SX`

`series`
`resume`
`resume*`

We define the keys `series`, `resume` and `resume*` for the “all levels” of `enumext` and `enumext*`. Here we do not need to make sure that `\printkeyans` is not running otherwise the startup value of the environments would be increased when using `resume` or `resume*` keys.

```

1840 \cs_set_protected:Npn \__enumext_tmp:n #1
1841 {
1842     \keys_define:nn { enumext / #1 }
1843     {
1844         series .str_set:N = \l__enumext_series_name_str,
1845         series .value_required:n = true,
1846         resume* .code:n = {
1847             \bool_if:NF \l__enumext_print_keyans_cmd_bool
1848             {
1849                 \__enumext_resume_star:
1850             }
1851         },
1852         resume* .value_forbidden:n = true,
1853     }

```

```

1854 }
1855 \clist_map_inline:nn {level-1, level-2, level-3, level-4, enumext*} { \__enumext_tmp:n {#1} }

```

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 we must set the boolean variable `\l__enumext_resume_count_bool` set by the key `resume without value` to “true” to be later processed by the function `__enumext_parse_series:n` in the definition of the environments `enumext` and `enumext*`.

```

1856 \cs_set_protected:Npn \__enumext_tmp:n #1
1857 {
1858   \keys_define:nn { enumext / #1 }
1859   {
1860     resume .code:n = {
1861       \bool_if:NF \l__enumext_print_keyans_cmd_bool
1862       {
1863         \tl_set:Nn \l__enumext_series_name_tl {##1}
1864         \tl_if_empty:NTF \l__enumext_series_name_tl
1865         {
1866           \typeout{resume=empty-for-series}
1867           \bool_set_true:N \l__enumext_resume_count_bool
1868         }
1869         {
1870           % bool true for resume=name
1871           \bool_set_true:c { l__enumext_resume_series_ \__enumext_level:
1872             \__enumext_resume:n {##1}
1873         }
1874       }
1875     },
1876   }
1877 }
1878 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
1879 \keys_define:nn { enumext / enumext* }
1880 {
1881   resume .code:n = {
1882     \bool_if:NF \l__enumext_print_keyans_cmd_bool
1883     {
1884       \tl_set:Nn \l__enumext_series_name_tl {##1}
1885       \tl_if_empty:NTF \l__enumext_series_name_tl
1886       {
1887         \typeout{resume=empty-for-series}
1888         \bool_set_true:N \l__enumext_resume_count_bool
1889       }
1890       {
1891         \bool_set_true:N \l__enumext_resume_series_vii_bool
1892         \__enumext_resume_vii:n {#1}
1893       }
1894     }
1895   },
1896 }

```

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

13.26.1 Internal functions for series key

`__enumext_filter_series:n` The function `__enumext_filter_series:n` will be in charge of filtering the `⟨keys⟩` we want to store where `#1` represents the *optional argument* passed to the environment.

```

1897 \cs_new:Npn \__enumext_filter_series:n #1
1898 {
1899   \use:e
1900   {
1901     \keyval_parse:NNn
1902     \__enumext_filter_series_key:n
1903     \__enumext_filter_series_pair:nn {#1}
1904   }
1905 }

```

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.

```

1906 \cs_new:Npn \__enumext_filter_series_key:n #1
1907 {
1908   \str_case:nnF {#1}
1909   {
1910     { resume } {} { resume* } {} { reset } {} { reset* } {} { base-fix } {}

```

```

1911     }
1912     { , { \exp_not:n {#1} } }
1913 }

```

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.

```

1914 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1915 {
1916   \str_case:nnF {#1}
1917   {
1918     { series } {} { resume } {} { start } {}
1919     { start* } {} { save-ans } {} { save-key } {}
1920   }
1921   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1922 }

```

(End of definition for `__enumext_filter_series:n`, `__enumext_filter_series_key:n`, and `__enumext_filter_series_pair:nn`.)

```

\__enumext_parse_series:n
\__enumext_save_last_keys:n

```

The function `__enumext_parse_series:n` will be responsible for storing the filtered *(keys)* in the global variable `\g__enumext_series_<series name>_tl` along with the creation of the integer variable `\g__enumext_series_<series name>_int` when the key is passed as an argument; otherwise, it will check the state of the boolean variable `\l__enumext_resume_X_bool` set by the keys `resume` and `resume*` and will call the function `__enumext_save_last_keys:n`.

- The value of boolean variable `\l__enumext_resume_X_bool` is set to true by the function `__enumext_resume_series:n` which is used by the keys `resume` and `resume*`, in this case we must make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.42) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.47).

```

1923 \cs_new_protected:Npn \__enumext_parse_series:n #1
1924 {
1925   \str_if_empty:NTF \l__enumext_series_name_str
1926   {
1927     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1928     {
1929       \bool_if:cF { l__enumext_resume_ \__enumext_level: _bool }
1930       {
1931         \__enumext_save_last_keys:n {#1}
1932       }
1933       % Aquí es necesario usar resume-counter...bool_lazy_and es la opción aquí
1934       \bool_if:NT \l__enumext_resume_count_bool
1935       {
1936         \tl_if_empty:NT \l__enumext_series_name_tl
1937         {
1938           \typeout{resume~~~~EMPTY~~~only~count}
1939           \__enumext_resume_counter:
1940         }
1941       }
1942     }
1943     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
1944     {
1945       \bool_if:NF \l__enumext_resume_vii_bool
1946       {
1947         \__enumext_save_last_keys:n {#1}
1948       }
1949       % Aquí es necesario usar resume-counter...bool_lazy_and es la opción aquí
1950       \bool_if:NT \l__enumext_resume_count_bool
1951       {
1952         \tl_if_empty:NT \l__enumext_series_name_tl
1953         {
1954           \typeout{resume~~~~EMPTY~~~only~count}
1955           \__enumext_resume_counter:
1956         }
1957       }
1958     }
1959   }
1960   {
1961     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1962     {
1963       \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level:
1964       \tl_gset:ce

```

```

1965         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
1966         { \__enumext_filter_series:n {#1} }
1967     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level:
1968         {
1969             \int_new:c { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _
1970         }
1971     }
1972     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
1973     {
1974         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
1975         \tl_gset:ce
1976             { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
1977             { \__enumext_filter_series:n {#1} }
1978         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
1979         {
1980             \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
1981         }
1982     }
1983 }
1984 }

```

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⟩}` are NOT active and will save them in the variable `\g__enumext_save_last_keys_X_tl` for the `enumext` environment and in the variable `\g__enumext_save_last_keys_vii_tl` for the `enumext*` environment.

```

1985 \cs_new_protected:Npn \__enumext_save_last_keys:n #1
1986 {
1987     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1988     {
1989         \tl_gclear:c { g__enumext_save_last_keys_ \__enumext_level: _tl }
1990         \tl_gset:ce
1991             { g__enumext_save_last_keys_ \__enumext_level: _tl } { \__enumext_filter_series:n {#1} }
1992     }
1993     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
1994     {
1995         \tl_gclear:N \g__enumext_save_last_keys_vii_tl
1996         \tl_gset:Ne \g__enumext_save_last_keys_vii_tl { \__enumext_filter_series:n {#1} }
1997     }
1998 }

```

(End of definition for `__enumext_parse_series:n` and `__enumext_save_last_keys:n`)

13.26.2 Internal function to save counter value

```

\__enumext_standar_save_counter:
\__enumext_standar_save_counter_aux:
\__enumext_starred_save_counter:
\__enumext_starred_save_counter_aux:

```

The `__enumext_standar_save_counter:` and `__enumext_starred_save_counter:` functions will save the last counter value to `\g__enumext_series_⟨series name⟩_int` if the `series={⟨series name⟩}` key has been passed, to `\c@__enumext_resume_X_int` if it has passed the key `resume without value` and the key `series` is not active, in `\g__enumext_series_⟨series name⟩_X_int` if the key `resume={⟨series name⟩}` has been passed and in `\g__enumext_series_⟨store name⟩_X_int` if the key has been passed `save-ans={⟨store name⟩}`.

🔗 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 execute `series={⟨series name⟩}` and the integer variable with `\l__enumext_series_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§13.42) and the `enumext*` environment definition (§13.47).

```

1999 \cs_new_protected:Nn \__enumext_standar_save_counter:
2000 {
2001     \bool_if:NTF \g__enumext_standar_bool
2002     {
2003         \__enumext_standar_save_counter_aux:
2004         \int_compare:nNnT { \l__enumext_level_int } = { 1 }
2005         {
2006             \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2007             {
2008                 \int_gset:eq:cN
2009                     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
2010             }
2011         }
2012     }
2013     {
2014         \__enumext_standar_save_counter_aux:

```



```

2015     }
2016 }
2017 \cs_new_protected:Nn \__enumext_standar_save_counter_aux:
2018 {
2019     \str_if_empty:NF \l__enumext_series_name_str
2020     {
2021         \int_gset_eq:cc
2022         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int } { c@enumX \
2023     }
2024     \tl_if_empty:NTF \l__enumext_series_name_tl
2025     {
2026         \str_if_empty:NT \l__enumext_series_name_str
2027         {
2028             \tl_if_empty:NT \l__enumext_store_name_tl
2029             {
2030                 \int_gset_eq:cc
2031                 { c@ __enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
2032             }
2033         }
2034     }
2035     {
2036         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
2037         {
2038             \int_gset_eq:cc
2039             { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int } { c@enumX \__enumext_level: }
2040         }
2041     }
2042 }
2043 \cs_new_protected:Nn \__enumext_starred_save_counter:
2044 {
2045     \bool_if:NTF \g__enumext_starred_bool
2046     {
2047         \__enumext_starred_save_counter_aux:
2048         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2049         {
2050             \int_gset_eq:cN
2051             { g__enumext_resume_ \l__enumext_store_name_tl _int } { \value{enumXvii} }
2052         }
2053     }
2054     {
2055         \__enumext_starred_save_counter_aux:
2056     }
2057 }
2058 \cs_new_protected:Nn \__enumext_starred_save_counter_aux:
2059 {
2060     \str_if_empty:NF \l__enumext_series_name_str
2061     {
2062         \int_gset_eq:cN
2063         { g__enumext_series_ \l__enumext_series_name_str _vii_int } { \value{enumXvii} }
2064     }
2065     \tl_if_empty:NTF \l__enumext_series_name_tl
2066     {
2067         \str_if_empty:NT \l__enumext_series_name_str
2068         {
2069             \tl_if_empty:NT \l__enumext_store_name_tl
2070             {
2071                 \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
2072             }
2073         }
2074     }
2075     {
2076         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
2077         {
2078             \int_gset_eq:cN
2079             { g__enumext_series_ \l__enumext_series_name_tl _vii_int } { \value{enumXvii} }
2080         }
2081     }
2082 }

```

(End of definition for __enumext_standar_save_counter: and others.)

13.26.3 Internal functions for resume key

`\\enumext_resume:n`
`\\enumext_resume_vii:n`

The functions `\\enumext_resume:n` and `\\enumext_resume_vii:n` will handle the `{⟨argument⟩}` passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function `\\enumext_resume_series:n` is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is NOT present.

```

2083 \cs_new_protected:Npn \\enumext_resume:n #1
2084 {
2085   \int_compare:nNtT { \\enumext_level_int } > { 0 }
2086   {
2087     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \\enumext_level: _tl }
2088     {
2089       \\enumext_resume_series:n {#1}
2090       \exp_args:Ne \keys_set:nv { enumext / level-\\int_use:N \\enumext_level_int }
2091       { g__enumext_series_ \tl_to_str:n {#1} _ \\enumext_level: _tl }
2092     }
2093     {
2094       \msg_error:nnn { enumext } { unknown-series-standar } {#1}
2095     }
2096   }
2097 }
2098 \cs_new_protected:Npn \\enumext_resume_vii:n #1
2099 {
2100   \int_compare:nNtT { \\enumext_level_h_int } = { 1 }
2101   {
2102     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2103     {
2104       \\enumext_resume_series:n {#1}
2105       \keys_set:nv { enumext / enumext* }
2106       { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2107     }
2108     {
2109       \msg_error:nnn { enumext } { unknown-series-starred } {#1}
2110     }
2111   }
2112 }

```

(End of definition for `\\enumext_resume:n` and `\\enumext_resume_vii:n`)

`\\enumext_resume_counter:`
`\\enumext_resume_series:n`
`\\enumext_resume_counter_series:`

The `\\enumext_resume_counter:` function is executed when the `resume` key is used *without value*, only the “counters” for the “levels” of the environments 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.

```

2113 \cs_new_protected:Nn \\enumext_resume_counter:
2114 {
2115   \cs_set:Npn \\enumext_tmp:n ##1
2116   {
2117     \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2118     {
2119       \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
2120     }
2121   }
2122   \int_compare:nNtT { \\enumext_level_int } > { 0 }
2123   {
2124     \bool_lazy_and:nnTF
2125     { \bool_if_p:N \\enumext_standar_first_bool }
2126     { \bool_if_p:N \\enumext_store_active_bool }
2127     {
2128       \int_set:Nn \\enumext_start_i_int
2129       {
2130         \int_use:c { g__enumext_resume_ \\enumext_store_name_tl _int } + 1
2131       }
2132       \int_step_function:nnN { 2 } { \\enumext_level_int } \\enumext_tmp:n
2133     }
2134     {
2135       \int_step_function:nnN { 1 } { \\enumext_level_int } \\enumext_tmp:n
2136     }
2137   }
2138   \int_compare:nNtT { \\enumext_level_h_int } = { 1 }
2139   {
2140     \bool_lazy_and:nnTF
2141     { \bool_if_p:N \\enumext_starred_first_bool }

```

```

2142         { \bool_if_p:N \l__enumext_store_active_bool }
2143     {
2144         \int_set:Nn \l__enumext_start_vii_int
2145         {
2146             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2147         }
2148     }
2149     {
2150         \int_set:Nn \l__enumext_start_vii_int
2151         {
2152             \int_use:c { c@ __enumext_resume_vii_int } + 1
2153         }
2154     }
2155 }
2156 }

```

The function `__enumext_resume_series:n` will set the variable `\l__enumext_resume_X_bool` to true and pass the value of the key `resume` to the variable `\l__enumext_series_name_tl` which will contain the `{⟨series name⟩}`. If the variable `\l__enumext_series_name_tl` is empty, that is, we are passing the key `resume` *without value*, we will execute the function `__enumext_resume_counter:` otherwise, when we pass `resume={⟨series name⟩}` we will execute the function `__enumext_resume_counter_series:`.

```

2157 \cs_new_protected:Npn \__enumext_resume_series:n #1
2158 {
2159     \int_compare:nNt { \l__enumext_level_int } > { 0 }
2160     {
2161         \tl_clear:N \l__enumext_series_name_tl
2162         \tl_set:Nn \l__enumext_series_name_tl {#1}
2163         \bool_set_true:c { l__enumext_resume_ \__enumext_level: _bool }
2164         \__enumext_resume_counter_series:
2165     }
2166     \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
2167     {
2168         \tl_clear:N \l__enumext_series_name_tl
2169         \tl_set:Nn \l__enumext_series_name_tl {#1}
2170         \bool_set_true:N \l__enumext_resume_vii_bool
2171         \__enumext_resume_counter_series:
2172     }
2173 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the counters for the “current level” of the environments according to the value of the integer variables created by the `series` key. If the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key.

```

2174 \cs_new_protected:Nn \__enumext_resume_counter_series:
2175 {
2176     \cs_set:Npn \__enumext_tmp:n ##1
2177     {
2178         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2179         {
2180             \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2181             {
2182                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2183             }
2184         }
2185     }
2186     \int_compare:nNt { \l__enumext_level_int } > { 0 }
2187     {
2188         \bool_lazy_and:nnTF
2189         { \bool_if_p:N \l__enumext_standar_first_bool }
2190         { \bool_if_p:N \l__enumext_store_active_bool }
2191         {
2192             \int_set:Nn \l__enumext_start_i_int
2193             {
2194                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2195             }
2196             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2197         }
2198         {
2199             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2200         }
2201     }

```

```

2202 \int_compare:nNtT { \l__enumext_level_h_int } = { 1 }
2203 {
2204   \bool_lazy_and:nnTF
2205     { \bool_if_p:N \l__enumext_starred_first_bool }
2206     { \bool_if_p:N \l__enumext_store_active_bool }
2207   {
2208     \int_set:Nn \l__enumext_start_vii_int
2209       {
2210         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2211       }
2212   }
2213   {
2214     \int_set:Nn \l__enumext_start_vii_int
2215       {
2216         \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
2217       }
2218   }
2219 }
2220 }

```

(End of definition for `__enumext_resume_counter:`, `__enumext_resume_series:n`, and `__enumext_resume_counter_series:`.)

13.26.4 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.

```

2221 \cs_new_protected:Nn \__enumext_resume_star:
2222 {
2223   \cs_set:Npn \__enumext_tmp:n ##1
2224   {
2225     \tl_if_empty:cF { g__enumext_save_last_keys_ \int_to_roman:n {##1} _tl }
2226     {
2227       \__enumext_resume_counter:
2228       \exp_args:Ne \keys_set:nv
2229         { enumext / level- \int_use:N \l__enumext_level_int }
2230         { g__enumext_save_last_keys_ \int_to_roman:n {##1} _tl }
2231     }
2232   }
2233   \int_compare:nNtT { \l__enumext_level_int } > { 0 }
2234   {
2235     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2236   }
2237   \int_compare:nNtT { \l__enumext_level_h_int } = { 1 }
2238   {
2239     \tl_if_empty:NF \g__enumext_save_last_keys_vii_tl
2240     {
2241       \__enumext_resume_counter:
2242       \keys_set:nV { enumext / enumext* } \g__enumext_save_last_keys_vii_tl
2243     }
2244   }
2245 }

```

(End of definition for `__enumext_resume_star:`.)

13.27 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 we use “internal counters” for the `resume` and `resume*` keys which set the *start value*, but are not accessible by the user, it is 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](#).

`\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>}`.

```

2246 \NewDocumentCommand \resetenumext { s o m }
2247 {
2248   \bool_if:nTF {#1}
2249   {

```

```

2250     \__enumext_reset_count_resume_all:n {#3}
2251   }
2252   {
2253     \tl_if_novalue:nTF {#2}
2254     {
2255       \__enumext_reset_count_resume_levels:n {#3}
2256     }
2257     {
2258       \str_if_eq:nnTF {#2} { * }
2259       { \__enumext_starred_reset:n {#3} }
2260       {
2261         \bool_lazy_and:nnTF
2262         { \int_compare_p:nNn {#2} > 0 }
2263         { \int_compare_p:nNn {#2} < 5 }
2264         { \__enumext_standard_reset:nn {#2} {#3} }
2265         {
2266           \msg_error:nne { enumext } { out-of-range } { \int_eval:n {#2} }
2267         }
2268       }
2269     }
2270   }
2271 }
2272 \cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
2273 {
2274   \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } {%#2}
2275 }
2276 \cs_new_protected:Npn \__enumext_starred_reset:n #1
2277 {
2278   \__enumext_reset_count_resume:nn { vii } {#1}
2279 }
2280 \cs_new_protected:Npn \__enumext_reset_count_resume:nn #1 #2
2281 {
2282   \counterwithin*{enumX#1}{#2}
2283   \counterwithin*{\__enumext_resume_#1_int}{#2}
2284 }
2285 \cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
2286 \cs_new_protected:Npn \__enumext_reset_count_resume_all:n #1
2287 {
2288   \clist_map_inline:nn { i,ii,iii,iv,vii }
2289   {
2290     \__enumext_reset_count_resume:nn { ##1 } { #1 }
2291   }
2292 }
2293 \cs_new_protected:Npn \__enumext_reset_count_resume_levels:n #1
2294 {
2295   \clist_map_inline:nn { i,ii,iii,iv }
2296   {
2297     \__enumext_reset_count_resume:nn { ##1 } { #1 }
2298   }
2299 }

```

(End of definition for `\resetenumext` and others. This function is documented on page 11.)

13.28 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.

We define the keys `reset` and `reset*` for the “all levels” of `enumext` and `enumext*`.

```

reset* 2300 \cs_set_protected:Npn \__enumext_tmp:n #1
2301 {
2302   \keys_define:nn { enumext / #1 }
2303   {
2304     reset .code:n = \__enumext_standard_reset_key:,
2305     reset .value_forbidden:n = true,
2306     reset* .code:n = \__enumext_standard_reset_key_star:,
2307     reset* .value_forbidden:n = true,
2308   }
2309 }
2310 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
2311 \keys_define:nn { enumext / enumext* }
2312 {

```

```

2313     reset .code:n = \__enumext_starred_reset_key:,
2314     reset .value_forbidden:n = true,
2315     reset* .code:n = \__enumext_starred_reset_key:,
2316     reset* .value_forbidden:n = true,
2317 }

```

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

13.28.1 Internal functions for `reset` and `reset*` keys

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.

```

2318 \cs_new_protected:Nn \__enumext_standard_reset_key:
2319 {
2320     \int_compare:nNt { \__enumext_level_int } > { 0 }
2321     {
2322         \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n { \__enumext_level_int } _int }
2323         {
2324             \int_gzero:c { c@__enumext_resume_ \int_to_roman:n { \__enumext_level_int } _int }
2325         }
2326     }
2327 }

```

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*.

```

2328 \cs_new_protected:Nn \__enumext_standard_reset_key_star:
2329 {
2330     \cs_set:Npn \__enumext_tmp:n ##1
2331     {
2332         \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n { ##1 } _int }
2333         {
2334             \int_gzero:c { c@__enumext_resume_ \int_to_roman:n { ##1 } _int }
2335         }
2336     }
2337     \int_compare:nNt { \__enumext_level_int } > { 0 }
2338     {
2339         \int_step_function:nn { \__enumext_level_int } { 4 } \__enumext_tmp:n
2340     }
2341 }

```

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.

```

2342 \cs_new_protected:Nn \__enumext_starred_reset_key:
2343 {
2344     \int_gzero:c { c@__enumext_resume_vii_int }
2345 }

```

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

13.29 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.

13.29.1 Setting `save-ans` key

`save-ans` We define the keys `save-ans` only for the “first level” of `enumext` and `enumext*`.

```

2346 \cs_set_protected:Npn \__enumext_tmp:n #1
2347 {
2348     \keys_define:nn { enumext / #1 }
2349     {
2350         save-ans .code:n = \__enumext_storing_set:n { ##1 },
2351         save-ans .value_required:n = true,
2352     }
2353 }
2354 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n { #1 } } }

```

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

13.29.2 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 display in the terminal and `.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:`.

```

2355 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2356 {
2357     \msg_term:nnVV { enumext } { save-ans-log }
2358     \g__enumext_envir_name_tl \l__enumext_store_name_tl
2359 }
2360 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2361 {
2362     \msg_term:nnVV { enumext } { save-ans-log-hook }
2363     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2364 }

```

(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 pass the value of the `save-ans` key to the variable `\l__enumext_store_name_tl` which will contain the `{⟨store name⟩}` of the *sequence* and *prop list* we will use. If `\l__enumext_store_name_tl` is *empty* we return an error message, otherwise will return the appropriate message `__enumext_start_save_ans_msg:` and proceed to execute the function `__enumext_storing_exec:` for `enumext` and `enumext*` environments.

```

2365 \cs_new_protected:Npn \__enumext_storing_set:n #1
2366 {
2367     \tl_set:Nx \l__enumext_store_name_tl {#1}
2368     \tl_if_empty:NTF \l__enumext_store_name_tl
2369     {
2370         \bool_lazy_or:nnT
2371         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2372         {
2373             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2374         }
2375     }
2376     {
2377         \bool_lazy_or:nnT
2378         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2379         {
2380             \__enumext_start_save_ans_msg:
2381             \__enumext_storing_exec:
2382         }
2383     }
2384 }

```

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, copy `{⟨store name⟩}` into the variable `\g__enumext_store_name_tl`.

```

2385 \cs_new_protected:Nn \__enumext_storing_exec:
2386 {
2387     \bool_set_true:N \l__enumext_store_active_bool
2388     \bool_set_true:N \l__enumext_check_answers_bool
2389     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl

```

The *prop list* `\g__enumext_series_⟨store name⟩_prop` and the *sequence* `\g__enumext_series_⟨store name⟩_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_⟨store name⟩_int` used by the keys `resume` and `resume*`.

```

2390     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2391     {
2392         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2393         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2394     }
2395     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2396     {
2397         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2398         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2399     }
2400     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }

```

```

2401     {
2402         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2403         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2404     }
2405 }

```

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

13.29.3 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 decrementing by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

- 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 we need to decrementing 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 we will use a “hook” function after closing the *first level* of the environment.

13.29.4 Setting check-ans and no-store keys

Now we define the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```

check-ans no-store
2406 \cs_set_protected:Npn \__enumext_tmp:n #1
2407 {
2408     \keys_define:nn { enumext / #1 }
2409     {
2410         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2411         check-ans .initial:n = false,
2412         check-ans .value_required:n = true,
2413         no-store .code:n = {
2414             \bool_set_false:N \l__enumext_check_answers_bool
2415             \bool_set_false:N \l__enumext_check_ans_key_bool
2416         },
2417         no-store .value_forbidden:n = true,
2418     }
2419 }
2420 \clist_map_inline:nn
2421 {
2422     level-1, level-2, level-3, level-4, enumext*
2423 }
2424 { \__enumext_tmp:n {#1} }

```

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

13.29.5 Set-up check answer mechanism

The function `__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `__enumext_check_ans_level:` only if “true”, i.e. the key `no-store` is not active.

```

2425 \cs_new_protected:Nn \__enumext_check_ans_active:
2426 {

```

```

2427     \tl_if_empty:NF \l__enumext_store_name_tl
2428     {
2429         \bool_if:NT \l__enumext_check_answers_bool
2430         {
2431             \__enumext_check_ans_level:
2432         }
2433     }
2434 }

```

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”.

```

2435 \cs_new_protected:Nn \__enumext_check_ans_level:
2436 {
2437     \int_case:nn { \l__enumext_level_int }
2438     {
2439         { 1 }{
2440             \bool_lazy_all:nT
2441             {
2442                 { \bool_if_p:N \g__enumext_starred_bool }
2443                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2444             }
2445             {
2446                 \int_gdecr:N \g__enumext_item_number_int
2447                 \bool_set_false:N \l__enumext_item_number_bool
2448             }
2449         }
2450         { 2 }{
2451             \int_gdecr:N \g__enumext_item_number_int
2452             \bool_set_false:N \l__enumext_item_number_bool
2453         }
2454         { 3 }{
2455             \int_gdecr:N \g__enumext_item_number_int
2456             \bool_set_false:N \l__enumext_item_number_bool
2457         }
2458         { 4 }{
2459             \int_gdecr:N \g__enumext_item_number_int
2460             \bool_set_false:N \l__enumext_item_number_bool
2461         }
2462     }

```

We should only execute this 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.

```

2463     \int_case:nn { \l__enumext_level_h_int }
2464     {
2465         { 1 }{
2466             \bool_lazy_all:nT
2467             {
2468                 { \bool_if_p:N \g__enumext_standar_bool }
2469                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2470             }
2471             {
2472                 \int_gdecr:N \g__enumext_item_number_int
2473                 \bool_set_false:N \l__enumext_item_number_bool
2474             }
2475         }
2476     }
2477 }

```

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

`__enumext_check_ans_key_hook:`

The function `__enumext_check_ans_key_hook:` will *export* the status of the local variable `\l__enumext_check_ans_key_bool` to the global variable `\g__enumext_check_ans_key_bool` only if the key `check-ans` is active.

```

2478 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2479 {
2480     \bool_lazy_and:nnT
2481     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2482     { \bool_if_p:N \g__enumext_standar_bool }
2483     {

```

```

2484         \bool_gset_true:N \g__enumext_check_ans_key_bool
2485     }
2486 \bool_lazy_and:nnT
2487 { \bool_if_p:N \l__enumext_check_ans_key_bool }
2488 { \bool_if_p:N \g__enumext_starred_bool }
2489 {
2490     \bool_gset_true:N \g__enumext_check_ans_key_bool
2491 }
2492 }

```

(End of definition for __enumext_check_ans_key_hook:.)

`__enumext_item_answer_diff:` The function `__enumext_item_answer_diff:` will set the value of the variable `\g__enumext_item_answer_diff_int` which is used by the functions `__enumext_check_ans_show:` for the key `save-ans` and by the function `__enumext_check_ans_log:` by the internal “*check answer*” mechanism. This function will be passed to the function `__enumext_execute_after_env:`.

```

2493 \cs_new_protected:Nn \__enumext_item_answer_diff:
2494 {
2495     \int_gset:Nn \g__enumext_item_answer_diff_int
2496     {
2497         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2498     }
2499 }

```

(End of definition for __enumext_item_answer_diff:.)

`__enumext_check_ans_show:` The function `__enumext_check_ans_show:` will be executed within the function `__enumext_execute_after_env:` when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is “*true*” and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `__enumext_item_answer_diff:`.

```

2500 \cs_new_protected:Nn \__enumext_check_ans_show:
2501 {
2502     \int_case:nn { \g__enumext_item_answer_diff_int }
2503     {
2504         { -1 } { \__enumext_check_ans_msg_less: }
2505         { 0 } { \__enumext_check_ans_msg_same_ok: }
2506         { 1 } { \__enumext_check_ans_msg_greater: }
2507     }
2508 }
2509 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2510 {
2511     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2512     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2513 }
2514 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2515 {
2516     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2517     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2518 }
2519 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2520 {
2521     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2522     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2523 }

```

(End of definition for __enumext_check_ans_show: and others.)

`__enumext_check_ans_log:` The function `__enumext_check_ans_log:` will be executed within the function `__enumext_execute_after_env:` when the key `check-ans` is not active, that is, when `\g__enumext_check_ans_key_bool` is “*false*” and write in the log the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `__enumext_item_answer_diff:`.

```

2524 \cs_new_protected:Nn \__enumext_check_ans_log:
2525 {
2526     \int_case:nn { \g__enumext_item_answer_diff_int }
2527     {
2528         { -1 } { \__enumext_check_ans_log_msg_less: }
2529         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2530         { 1 } { \__enumext_check_ans_log_msg_greater: }
2531     }
2532 }

```

```

2533 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2534 {
2535   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2536   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2537 }
2538 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2539 {
2540   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2541   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2542 }
2543 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2544 {
2545   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2546   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2547 }

```

(End of definition for __enumext_check_ans_log: and others.)

13.29.6 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.

```

2548 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2549 {
2550   \int_compare:nNnT
2551   { \g__enumext_check_starred_cmd_int } = { 0 }
2552   {
2553     \msg_warning:nnnV
2554     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2555   }
2556   \int_compare:nNnT
2557   { \g__enumext_check_starred_cmd_int } > { 1 }
2558   {
2559     \msg_warning:nnnV
2560     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2561   }
2562   \int_gzero:N \g__enumext_check_starred_cmd_int
2563   \tl_clear:N \l__enumext_check_start_line_env_tl
2564 }

```

(End of definition for __enumext_check_starred_cmd:n.)

13.30 Keys and functions associated with storage

13.30.1 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 we will only add the keys `mark-ans*`, `mark-pos*`, `mark-sep*`, `wrap-ans*`, `wrap-opt`, `save-sep`, `show-ans` and `show-pos`.

```

2565 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2566 {
2567   \keys_define:nn { enumext / #1 }
2568   {
2569     mark-ans* .tl_set:c = { \l__enumext_mark_answer_sym_#2_tl },
2570     mark-ans* .initial:n = \textasteriskcentered,
2571     mark-ans* .value_required:n = true,
2572     mark-pos* .choice:,
2573     mark-pos* / left .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { l },
2574     mark-pos* / right .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { r },
2575     mark-pos* / center .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { c },
2576     mark-pos* / unknown .code:n =
2577       \msg_error:nneee { enumext } { unknown-choice }
2578       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2579     mark-pos* .initial:n = right,
2580     mark-pos* .value_required:n = true,
2581     mark-sep* .dim_set:c = { \l__enumext_mark_sym_sep_#2_dim },
2582     mark-sep* .value_required:n = true,
2583     wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2584     wrap-ans* .value_required:n = true,
2585     wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,

```

```

2586     wrap-opt .initial:n = [##1],
2587     wrap-opt .value_required:n = true,
2588     save-sep .tl_set:c = { l__enumext_store_keyans_item_opt_sep_#2_tl },
2589     save-sep .initial:n = {,~},
2590     save-sep .value_required:n = true,
2591     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2592     show-ans .initial:n = false,
2593     show-ans .value_required:n = true,
2594     show-pos .bool_set:N = \l__enumext_show_position_bool,
2595     show-pos .initial:n = false,
2596     show-pos .value_required:n = true,
2597   }
2598 }
2599 \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:n #1 }

```

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

We add the $\langle\text{keys}\rangle$ mark-ref and save-ref related to the “storage system” and internal mechanism of “label and ref” along with the $\langle\text{keys}\rangle$ show-ans, show-pos and the $\langle\text{keys}\rangle$ mark-ans, mark-pos, mark-sep and wrap-ans for the command `\anskey`, the environment `anskey*` and the the $\langle\text{keys}\rangle$ for environments `keyans` and `keyans*` only at the *first level* of `enumext` and `enumext*`.

```

2600 \cs_set_protected:Npn \__enumext_tmp:n #1
2601 {
2602   \keys_define:nn { enumext / #1 }
2603   {
2604     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2605     mark-ref .initial:n = \textreferencemark,
2606     mark-ref .value_required:n = true,
2607     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2608     save-ref .initial:n = false,
2609     save-ref .value_required:n = true,
2610     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2611     show-ans .initial:n = false,
2612     show-ans .value_required:n = true,
2613     show-pos .bool_set:N = \l__enumext_show_position_bool,
2614     show-pos .initial:n = false,
2615     show-pos .value_required:n = true,
2616     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2617     mark-ans .initial:n = \textasteriskcentered,
2618     mark-ans .value_required:n = true,
2619     mark-sep .dim_set:N = \l__enumext_mark_sym_sep_dim,
2620     mark-sep .value_required:n = true,
2621     mark-pos .choice:,
2622     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2623     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2624     mark-pos / center .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2625     mark-pos / unknown .code:n =
2626       \msg_error:nnee { enumext } { unknown-choice }
2627       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2628     mark-pos .initial:n = right,
2629     mark-pos .value_required:n = true,
2630
2631     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2632     wrap-ans .initial:n =
2633       {
2634         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2635       },
2636     wrap-ans .value_required:n = true,
2637     mark-ans* .code:n = {
2638       \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2639       \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2640     },
2641     mark-ans* .value_required:n = true,
2642     mark-pos* .code:n = {
2643       \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2644       \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2645     },
2646     mark-pos* .value_required:n = true,
2647     mark-sep* .code:n = {
2648       \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2649       \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }

```

```

2650         },
2651         mark-sep* .value_required:n = true,
2652         wrap-ans* .code:n = {
2653             \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2654             \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2655         },
2656         wrap-ans* .value_required:n = true,
2657         wrap-opt .code:n = {
2658             \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2659             \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2660         },
2661         wrap-opt .value_required:n = true,
2662         save-sep .code:n = {
2663             \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2664             \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2665         },
2666         save-sep .value_required:n = true,
2667     }
2668 }
2669 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.30.2 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 we must capture the *optional argument* passed to the levels of the environment in which it is executed and “*storing*” this 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. We will execute this function only if the variable `__enumext_store_save_key_X_bool` is false, that is, the key *store-key* is not active, establishing the variable `__enumext_store_save_key_X_tl` with the filtered *keys*.

```

2670 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2671 {
2672     \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2673     {
2674         \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }
2675         \tl_set:ce
2676         { \__enumext_store_save_key_ \__enumext_level: _tl }
2677         { \__enumext_filter_save_key:n {#1} }
2678     }
2679 }
2680 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2681 {
2682     \bool_if:NF \__enumext_store_save_key_vii_bool
2683     {
2684         \tl_clear:N \__enumext_store_save_key_vii_tl
2685         \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2686     }
2687 }

```

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

13.30.3 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the *save-ans* key when executing `\anskey` or *anskey**, we will not be able to modify it. The best thing here is to have a key that allows you to modify the *optional argument* of the “*storing structure*” in the *sequence*.

save-key

The values set by this key passed in the *optional argument* of the *enumext* and *enumext** environments will override the values of the `__enumext_store_save_key_X_tl` variable set by the functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n`. Now define the key *save-key* for all levels of *enumext* and *enumext** environments.

```

2688 \cs_set_protected:Npn \__enumext_tmp:n #1
2689 {
2690     \keys_define:nn { enumext / enumext* }
2691     {
2692         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2693         save-key .value_required:n = true,

```



```

2694     }
2695     \keys_define:nn { enumext / #1 }
2696     {
2697         save-key .code:n = \__enumext_parse_save_key:n {##1},
2698         save-key .value_required:n = true,
2699     }
2700 }
2701 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for “*storing keys*” in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2702 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2703 {
2704     \bool_set_true:c { \__enumext_store_save_key_ \__enumext_level: _bool }
2705     \tl_clear:c { \__enumext_save_key_ \__enumext_level: _tl }
2706     \tl_set:ce
2707     { \__enumext_store_save_key_ \__enumext_level: _tl }
2708     { \__enumext_filter_save_key:n {#1} }
2709 }
2710 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2711 {
2712     \bool_set_true:N \__enumext_store_save_key_vii_bool
2713     \tl_clear:N \__enumext_store_save_key_vii_tl
2714     \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2715 }

```

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

13.30.4 Internal functions to store optional arguments

The function `__enumext_filter_save_key:n` will be in charge of “*filtering keys*” we want to *stored in sequence* where `{#1}` represents the *optional argument* passed to the environment.

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn
2716 \cs_new:Npn \__enumext_filter_save_key:n #1
2717 {
2718     \use:e
2719     {
2720         \keyval_parse:NNn
2721         \__enumext_filter_save_key_key:n
2722         \__enumext_filter_save_key_pair:nn {#1}
2723     }
2724 }

```

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.

```

2725 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2726 {
2727     \str_case:nnF {#1}
2728     {
2729         { resume } {} { resume* } {} { reset } {} { reset* } {} { no-store } {} { base-fix } {}
2730     }
2731     { , { \exp_not:n {#1} } }
2732 }

```

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`, `save-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.

```

2733 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2734 {
2735     \str_case:nnF {#1}
2736     {
2737         { series } {} { resume } {} { save-ans } {} { save-ref } {}
2738         { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2739         { mark-ans } {} { mark-pos } {} { mark-sep } {} { wrap-ans } {}
2740         { mark-ans* } {} { mark-pos* } {} { mark-sep* } {} { wrap-ans* } {}
2741         { wrap-opt } {} { save-sep } {} { mark-ref } {} { mini-env } {}
2742         { mini-sep } {} { mini-right } {} { mini-right* } {}
2743     }
2744     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2745 }

```

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

13.30.5 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 $\{\langle content \rangle\}$ in *prop list* defined by `save-ans` key. The “*stored content*” is retrieved by means of the `\getkeyans` command.

The form in which the $\{\langle content \rangle\}$ is “*stored*” in the *prop list* is $\{\langle position \rangle\}\{\langle content \rangle\}$. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
2746 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2747 {
2748   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2749   {
2750     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2751   }
2752   { #1 }
2753 }
2754 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

13.30.6 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 $\{\langle content \rangle\}$ in *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

The form in which the $\{\langle content \rangle\}$ is stored in *sequence* is in a 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.

```
2755 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2756 {
2757   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2758 }
2759 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.30.7 Functions for storing structure in the sequence

`__enumext_store_level_open:`
`__enumext_store_level_close:`

The “*storing structure*” is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```
2760 \cs_new_protected:Nn \__enumext_store_level_open:
2761 {
2762   \bool_if:NT \l__enumext_check_answers_bool
2763   {
2764     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2765     {
2766       \__enumext_store_addto_seq:n
2767       {
2768         \item \begin{enumext}
2769       }
2770     }
2771     {
2772       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2773       {
2774         \item \begin{enumext} [
2775         ]
2776       }
2777       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2778       {
2779         ]
2780       }
2781       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2782     }
2783   }
2784 \cs_new_protected:Nn \__enumext_store_level_close:
2785 {
2786   \bool_if:NT \l__enumext_check_answers_bool
2787   {
2788     \__enumext_store_addto_seq:n { \end{enumext} }
2789   }
2790 }
```

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

```
\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:
```

The “*storing structure*” is handled by the functions `__enumext_store_level_open_vii:` and `__enumext_store_level_close_vii:` which are executed in the `enumext*` environment.

```
2791 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2792 {
2793   \bool_if:NT \l__enumext_check_answers_bool
2794   {
2795     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2796     {
2797       \__enumext_store_addto_seq:n
2798       {
2799         \item \begin{enumext*}
2800       }
2801     }
2802     {
2803       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2804       {
2805         \item \begin{enumext*}[
2806       }
2807       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2808       {
2809         ]
2810       }
2811       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2812     }
2813   }
2814 }
2815 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2816 {
2817   \bool_if:NT \l__enumext_check_answers_bool
2818   {
2819     \__enumext_store_addto_seq:n { \end{enumext*} }
2820   }
2821 }
```

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

13.30.8 Function for show marks and position

```
\__enumext_print_keyans_box:NN
\__enumext_print_keyans_box:cc
```

The function `__enumext_print_keyans_box:NN` print a box in the left margin with `\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`

```
2822 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2823 {
2824   \mode_leave_vertical:
2825   \skip_horizontal:n { -\dim_use:N #2 }
2826   \hbox_overlap_left:n
2827   {
2828     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2829     {
2830       \tl_use:N \l__enumext_mark_answer_sym_tl
2831     }
2832   }
2833   \skip_horizontal:n { \dim_use:N #2 }
2834 }
2835 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

13.31 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` will allow to execute `\ref{⟨store name : position⟩}` and will return `1.(a).i.A`.

```
\__enumext_store_internal_ref:
```

First we will remove the dots “.” from the current `⟨labels⟩`, we do not want to get double dots in our references, then we will place this in the variable `\l__enumext_newlabel_arg_two_tl`.

```
2836 \cs_new_protected:Nn \__enumext_store_internal_ref:
2837 {
```

```

2838 \cs_set_protected:Npn \__enumext_tmp:n ##1
2839 {
2840   \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2841   \tl_reverse:c { l__enumext_label_copy_##1_tl }
2842   \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2843   \tl_reverse:c { l__enumext_label_copy_##1_tl }
2844 }
2845 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2846 \cs_set:Npn \__enumext_tmp:n ##1
2847 { . \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }

```

Here we need to analyse 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.

```

2848 \bool_lazy_all:nT
2849 {
2850   { \bool_if_p:N \g__enumext_starred_bool }
2851   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2852 }
2853 {
2854   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2855   { \tl_use:N \l__enumext_label_copy_vii_tl }
2856 }
2857 \bool_lazy_all:nT
2858 {
2859   { \bool_not_p:n { \g__enumext_standar_bool } }
2860   { \bool_if_p:N \l__enumext_standar_bool }
2861   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2862 }
2863 {
2864   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2865   {
2866     \tl_use:N \l__enumext_label_copy_vii_tl
2867     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2868   }
2869 }

```

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.

```

2870 \bool_lazy_all:nT
2871 {
2872   { \bool_if_p:N \g__enumext_standar_bool }
2873   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2874   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2875 }
2876 {
2877   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2878   {
2879     \tl_use:N \l__enumext_label_copy_i_tl
2880     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2881   }
2882 }
2883 \cs_set:Npn \__enumext_tmp:n ##1
2884 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2885 \bool_lazy_all:nT
2886 {
2887   { \bool_if_p:N \g__enumext_standar_bool }
2888   { \bool_if_p:N \l__enumext_starred_bool }
2889   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2890 }
2891 {
2892   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2893   {
2894     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2895     \tl_use:N \l__enumext_label_copy_vii_tl
2896   }
2897 }

```

Now we set the variable `\l__enumext_newlabel_arg_one_tl` which will contain $\langle \textit{store name} : \textit{position} \rangle$.

```

2898 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2899 {
2900   \l__enumext_store_name_tl \c_colon_str
2901   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2902 }

```

Now execute the function `__enumext_newlabel:nn` and save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

2903 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2904 {
2905   \__enumext_newlabel:nn
2906   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2907   { \l__enumext_newlabel_arg_two_tl }
2908 }
2909 \l__enumext_write_aux_file_tl
2910 }

```

(End of definition for `__enumext_store_internal_ref:`)

13.32 Common functions for `\anskey` and `anskey*` environment

`__enumext_store_anskey_arg:n`

The internal function `__enumext_store_anskey_arg:n` first we pass the $\langle argument \rangle$ to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the *save-ref* key and will call the function `__enumext_store_internal_ref:` for the “*internal label and ref*” system. Followed by this if the *show-ans* or *show-pos* keys are active we will show the “*wrapped*” $\langle argument \rangle$.

```

2911 \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2912 {
2913   \int_gincr:N \g__enumext_item_anskey_int
2914   \__enumext_store_addto_prop:n {#1}
2915   \bool_if:NT \l__enumext_store_ref_key_bool
2916   {
2917     \__enumext_store_internal_ref:
2918   }
2919   \__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the $[key = val]$ passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “*store*” in the *sequence*. First we clear the variable `\l__enumext_store_anskey_arg_tl` and process the $\langle keys \rangle$, if the *break-col* key is present and the command is running under *enumext* (not in *enumext**) we will add `\columnbreak` and then `\item`.

```

2920 \tl_clear:N \l__enumext_store_anskey_arg_tl
2921 \bool_lazy_and:nnT
2922 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2923 { \bool_not_p:n { \l__enumext_starred_bool } }
2924 {
2925   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2926 }
2927 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the *item-join* key is present and the command is running under *enumext** we will add $\langle number \rangle$ to `\l__enumext_store_anskey_arg_tl`.

```

2928 \bool_lazy_and:nnT
2929 { \bool_not_p:n { \l__enumext_starred_bool } }
2930 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2931 {
2932   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2933   {
2934     ( \exp_not:V \l__enumext_store_item_join_int )
2935   }
2936 }

```

And now we will review the keys *item-star*, *item-sym** and *item-pos** and pass them to `\l__enumext_store_anskey_arg_tl` along with the $\langle argument \rangle$ for `\anskey` or $\langle body \rangle$ for *anskey**.

```

2937 \bool_if:NTF \l__enumext_store_item_star_bool
2938 {
2939   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2940   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2941   {
2942     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2943     {
2944       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2945     }
2946   }
2947   \dim_compare:nT
2948   {
2949     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2950   }
2951   {
2952     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl

```

```

2953         {
2954             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2955         }
2956     }
2957     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2958 }
2959 {
2960     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2961 }

```

Finally we check if the `save-ref` key are 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*.

```

2962 \bool_lazy_and:nnT
2963 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2964 { \bool_if_p:N \l__enumext_hyperref_bool }
2965 {
2966     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2967     {
2968         \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2969         { \exp_not:V \l__enumext_mark_ref_sym_tl }
2970     }
2971 }
2972 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2973 }

```

(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 $\langle argument \rangle$ passed to `\anskey` and the $\langle body \rangle$ for `anskey*` when using the `wrap-ans` and `wrap-sep` keys.

```

2974 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2975 {
2976     \par
2977     \bool_if:NTF \l__enumext_starred_bool
2978     {
2979         \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2980         {
2981             \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2982         }
2983         \__enumext_print_keyans_box:NN
2984         \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2985     }
2986     {
2987         \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2988         {
2989             \dim_set:Nn \l__enumext_mark_sym_sep_dim
2990             {
2991                 \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2992             }
2993         }
2994         \__enumext_print_keyans_box:cc
2995         { \l__enumext_labelwidth_ \__enumext_level: _dim } { \l__enumext_mark_sym_sep_dim }
2996     }
2997     \__enumext_anskey_wrapper:n { #1 }
2998 }

```

(End of definition for `__enumext_anskey_show_wrap_arg:n`)

`__enumext_anskey_show_wrap_left:n`

The function `__enumext_anskey_show_wrap_left:n` will show the “*mark*” defined by the `mark-ans` key or the “*position*” of the $\langle content \rangle$ stored in the *prop list* when using the `show-pos` key on the left margin next to the “wraps” $\langle argument \rangle$ passed to `\anskey` and the $\langle body \rangle$ in `anskey*` on the right side when using the `show-ans` key.

```

2999 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
3000 {
3001     \bool_if:NT \l__enumext_show_answer_bool
3002     {
3003         \__enumext_anskey_show_wrap_arg:n { #1 }
3004     }
3005     \bool_if:NT \l__enumext_show_position_bool
3006     {
3007         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3008         {

```

```

3009         \group_begin:
3010         \exp_not:N \normalfont
3011         \exp_not:N \footnotesize [ \int_eval:n
3012         {
3013             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3014         }
3015         ]
3016         \group_end:
3017     }
3018     \__enumext_anskey_show_wrap_arg:n { #1 }
3019 }
3020 }

```

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

13.33 The command `\anskey`

Since we will be “*storing content*” in a `list` environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing.

The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows `\anskey[⟨key = val⟩]{⟨content⟩}`.

First we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```

break-col 3021 \keys_define:nn { enumext / anskey }
item-join 3022 {
item-star 3023     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
item-sym* 3024     break-col .default:n = true,
            3025     break-col .value_forbidden:n = true,
            3026     item-join .int_set:N = \l__enumext_store_item_join_int,
            3027     item-join .value_required:n = true,
            3028     item-star .bool_set:N = \l__enumext_store_item_star_bool,
            3029     item-star .default:n = true,
            3030     item-star .value_forbidden:n = true,
            3031     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
            3032     item-sym* .value_required:n = true,
            3033     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
            3034     item-pos* .value_required:n = true,
            3035     unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
            3036 }

```

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`.

```

3037 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
3038 {
3039     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
3040 }
3041 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
3042 {
3043     \tl_if_blank:nTF {#2}
3044     {
3045         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
3046     }
3047     {
3048         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
3049     }
3050 }

```

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

- 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 We will first call the function `__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and execute the function `__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[⟨key = val⟩]` and call the function `__enumext_store_anskey_arg:n`.

```

3051 \NewDocumentCommand \anskey { o +m }
3052 {
3053     \__enumext_anskey_safe_outer:
3054     \group_begin:

```



```

3055 \bool_if:NT \l__enumext_check_answers_bool
3056 {
3057   \tl_if_novalue:nF {#1}
3058   {
3059     \keys_set:nn { enumext / anskey } {#1}
3060   }
3061   \tl_if_blank:nTF {#2}
3062   {
3063     \msg_error:nn { enumext } { anskey-empty-arg }
3064   }
3065   {
3066     \__enumext_anskey_safe_inner:
3067     \__enumext_store_anskey_arg:n {#2}
3068   }
3069 }
3070 \group_end:
3071 }

```

(End of definition for `\anskey`. This function is documented on page 14.)

13.33.1 Internal functions for the command

`__enumext_anskey_safe_outer:`
`__enumext_anskey_safe_inner:`

The `__enumext_store_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.

```

3072 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
3073 {
3074   \bool_if:NF \l__enumext_store_active_bool
3075   {
3076     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
3077   }
3078   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3079   {
3080     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
3081   }
3082   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3083   {
3084     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
3085   }
3086   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
3087   {
3088     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
3089   }
3090 }

```

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.

```

3091 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3092 {
3093   \int_incr:N \l__enumext_anskey_level_int
3094   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
3095   {
3096     \msg_error:nn { enumext } { anskey-nested }
3097   }
3098   \bool_if:NF \l__enumext_item_number_bool
3099   {
3100     \msg_error:nn { enumext } { anskey-unnumber-item }
3101   }
3102   \mode_if_math:T
3103   {
3104     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
3105   }
3106 }

```

(End of definition for `__enumext_anskey_safe_outer:` and `__enumext_anskey_safe_inner:`.)

13.34 The environment `anskey*`

The original implementation of the `anskey*` environment used non-public functions from the `scontents`[4] package, which was not the best approach. Fortunately L^AT_EX release 2025-06-01 implemented the new `c`-type argument in the `\ltxcmd`[13], with which we can record the *(body)* of the environment in *verbatim mode* and, together with `\scantokens` do the work as the original implementation.

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

First we add the same keys from the `\anskey` command along with the `force-eol`, `write-env` and `overwrite` keys that were in the original implementation that used the `scontents` support package for these.

```

3107 \keys_define:nn { enumext / anskey* }
3108 {
3109     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3110     break-col .default:n = true,
3111     break-col .value_forbidden:n = true,
3112     item-join .int_set:N = \l__enumext_store_item_join_int,
3113     item-join .value_required:n = true,
3114     item-star .bool_set:N = \l__enumext_store_item_star_bool,
3115     item-star .default:n = true,
3116     item-star .value_forbidden:n = true,
3117     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
3118     item-sym* .value_required:n = true,
3119     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
3120     item-pos* .value_required:n = true,
3121     force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
3122     force-eol .initial:n = false,
3123     force-eol .default:n = true,
3124     write-env .code:n = {
3125         \bool_set_true:N \l__enumext_write_anskey_env_bool
3126         \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
3127     },
3128     write-env .value_required:n = true,
3129     overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
3130     overwrite .initial:n = false,
3131     overwrite .default:n = true,
3132     unknown .code:n = { \l__enumext_anskey_env_unknown:n {#1} },
3133 }

```

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

__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn

The `<keys>` are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `__enumext_anskey_env_unknown:n`.

```

3134 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
3135 {
3136     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
3137 }
3138 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
3139 {
3140     \tl_if_blank:nTF {#2}
3141     {
3142         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
3143     }
3144     {
3145         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3146     }
3147 }

```

(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

The conditional function `__enumext_anskey_env_file_if_writable:n` used by the `write-env` and `overwrite` keys in the `anskey*` environment to determine whether the output file is written or overwritten.

```

3148 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
3149 {
3150     \bool_if:nTF \l__enumext_write_anskey_env_bool
3151     {
3152         \file_if_exist:nTF {#1}
3153         {
3154             \bool_if:nTF \l__enumext_anskey_env_overwrite_bool
3155             {
3156                 \msg_warning:nne { enumext } { overwrite-file } {#1}
3157                 \prg_return_true:
3158             }
3159             {
3160                 \msg_warning:nne { enumext } { not-writing } {#1}
3161                 \prg_return_false:
3162             }
3163         }
3164     }

```

```

3165         \msg_warning:nne { enumext } { writing-file } {#1}
3166         \prg_return_true:
3167     }
3168 }
3169 { \prg_return_false: }
3170 }

```

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.

```

3171 \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2
3172 {
3173     \__enumext_anskey_env_file_if_writable:nT {#1}
3174     {
3175         \iow_open:Nn \__enumext_write_anskey_env_file_iow {#1}
3176         \iow_now:Nn \__enumext_write_anskey_env_file_iow {#2}
3177         \iow_close:N \__enumext_write_anskey_env_file_iow
3178     }
3179 }
3180 \cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }

```

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

anskey* First, we'll call the function `__enumext_anskey_env_safe_outer:` to make sure where we're running the environment, then, we'll check the state of the variable `__enumext_check_answers_bool` set by the key `no-store`. If it's true, we'll look for `[⟨key = val⟩]` and verify that the *argument* `c` (`⟨body⟩`) is not empty. Finally, we'll run the internal check function `__enumext_anskey_env_safe_inner:n` and call the function `__enumext_store_anskey_arg:n`.

```

3181 \NewDocumentEnvironment{anskey*}{ o c }
3182 {
3183     \__enumext_anskey_env_safe_outer:
3184     \bool_if:NT \__enumext_check_answers_bool
3185     {
3186         \tl_if_no_value:nF {#1}
3187         {
3188             \keys_set:nn { enumext / anskey* } {#1}
3189         }
3190         \tl_if_blank:nTF {#2}
3191         {
3192             \msg_error:nn { enumext } { anskey-empty-arg }
3193         }
3194         {
3195             \__enumext_anskey_env_safe_inner:
3196             \__enumext_store_anskey_env:n {#2}
3197         }
3198     }
3199 } { }

```

(End of definition for `anskey*`. This function is documented on page 15.)

13.34.1 Internal functions for the environment

`__enumext_anskey_env_safe_outer:` The function `__enumext_store_anskey_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.

```

3200 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3201 {
3202     \bool_if:NF \__enumext_store_active_bool
3203     {
3204         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3205     }
3206     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3207     {
3208         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
3209     }
3210     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
3211     {
3212         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3213     }
3214     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3215     {
3216         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3217     }
3218 }

```

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.

```

3219 \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
3220 {
3221   \bool_if:NF \l__enumext_item_number_bool
3222   {
3223     \msg_error:nn { enumext } { anskey-unnumber-item }
3224   }
3225   \mode_if_math:T
3226   {
3227     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
3228   }
3229 }

```

The `__enumext_store_anskey_env:n` function will first pass the argument `c` (*body*) to the variable `\l__enumext_store_anskey_env_tl` and replace the macro `\obeyedline` with `^^J` and then execute the `write-env` and `overwrite` keys, check the state of the variable `\l__enumext_anskey_env_force_eol_bool` managed by the `force-eol` key and we will add `\c__enumext_anskey_env_hidden_space_str` if necessary. Finally we will use `\exp_args:Ne` on the `__enumext_store_anskey_arg:n` to expand the `__enumext_scan_tokens:n` function which rescans the `\l__enumext_store_anskey_env_tl` variable before processing it.

```

3230 \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
3231 {
3232   \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
3233   \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^J }
3234   \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N ^^J }
3235   \__enumext_anskey_env_file_write:VV
3236   \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
3237   \bool_if:NF \l__enumext_anskey_env_force_eol_bool
3238   {
3239     \tl_put_right:Ne \l__enumext_store_anskey_env_tl
3240     {
3241       \c__enumext_anskey_env_hidden_space_str
3242     }
3243   }
3244   \exp_args:Ne
3245   \__enumext_store_anskey_arg:n
3246   {
3247     \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
3248   }
3249 }

```

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`.)

13.35 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 call the `__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `__enumext_check_ans_log:` and write the results in the `.log` file and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

3250 \cs_new_protected:Nn \__enumext_execute_after_env:
3251 {
3252   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3253   {
3254     \tl_if_empty:NF \g__enumext_store_name_tl
3255     {
3256       \__enumext_stop_save_ans_msg:
3257       \__enumext_item_answer_diff:
3258       \__enumext_log_global_vars:
3259       \__enumext_log_answer_vars:
3260       \bool_if:NTF \g__enumext_check_ans_key_bool
3261       {
3262         \__enumext_check_ans_show:
3263       }
3264     }
3265   }

```

```

3264         { \__enumext_check_ans_log: }
3265     }
3266     \__enumext_reset_global_vars:
3267 }
3268 }

```

• This function is passed to the function `__enumext_after_env:n` for the environments `enumext` (§13.42) and `enumext*` (§13.47) 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:.`)

13.36 Common functions for keyans, keyans* and keyanspic

13.36.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the the current $\langle label \rangle$ for $\langle item^* \rangle$ in `keyans` environment and the current $\langle label \rangle$ for $\langle anspic^* \rangle$ in `keyanspic` environment followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `__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`.

```

3269 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3270 {
3271     \tl_clear:N \__enumext_store_current_label_tl
3272     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
3273     {
3274         \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
3275     }
3276     {
3277         \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
3278     }

```

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

```

3279     \tl_if_novalue:nF { #1 }
3280     {
3281         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_v_tl
3282         {
3283             \tl_put_right:NV \__enumext_store_current_label_tl \__enumext_store_keyans_item_opt_
3284         }
3285         \tl_put_right:Nn \__enumext_store_current_label_tl { #1 }
3286     }
3287     \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
3288 }

```

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

13.36.2 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 $\langle label \rangle$ for $\langle item^* \rangle$ and $\langle anspic^* \rangle$ with the $\langle contents \rangle$ of the *optional argument*. The mechanism defined here will allow to execute `\ref{⟨store name: position⟩}` and will return `1. (A)`.

The function `__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the `save-ref` key for $\langle item^* \rangle$ and $\langle anspic^* \rangle$ commands. First we will create copies of the current $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in references.

```

3289 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3290 {
3291     \bool_if:NT \__enumext_store_ref_key_bool
3292     {
3293         \cs_set_protected:Npn \__enumext_tmp:n #1
3294         {
3295             \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3296             \tl_reverse:c { \__enumext_label_copy_##1_tl }
3297             \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3298             \tl_reverse:c { \__enumext_label_copy_##1_tl }
3299         }
3300         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3301         \__enumext_keyans_store_ref_aux_i:
3302     }
3303 }

```

The auxiliary function `__enumext_keyans_store_ref_aux_i`: set the variable `\l__enumext_newlabel_arg_one_tl` which will contain $\langle store\ name : position \rangle$ analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

3304 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3305 {
3306   \bool_if:NT \g__enumext_starred_bool
3307   {
3308     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3309   }
3310   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3311   {
3312     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3313     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3314   }
3315   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3316   {
3317     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3318     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3319   }
3320   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3321   {
3322     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3323     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3324   }
3325   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3326   {
3327     \l__enumext_store_name_tl \c_colon_str
3328     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3329   }
3330   \__enumext_keyans_store_ref_aux_ii:
3331 }

```

Now auxiliary function `__enumext_keyans_store_ref_aux_ii`: save the result in the variable `\l__enumext_write_aux_file_tl` and finally we write in the `.aux` file.

```

3332 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3333 {
3334   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3335   {
3336     \__enumext_newlabel:nn
3337     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3338     { \l__enumext_newlabel_arg_two_tl }
3339   }
3340   \l__enumext_write_aux_file_tl
3341 }

```

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

13.36.3 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 $\langle label \rangle$ `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3342 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3343 {
3344   \tl_clear:N \l__enumext_store_current_label_tl
3345   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3346   {
3347     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3348   }
3349   {
3350     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3351   }
3352   \tl_if_novalue:nF { #1 }
3353   {
3354     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3355     {
3356       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
3357     }
3358     \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }

```

```

3359     }
3360     \__enumext_keyans_addto_seq_link:
3361 }

3362 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3363 {
3364     \bool_lazy_and:nnT
3365     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3366     { \bool_if_p:N \l__enumext_hyperref_bool }
3367     {
3368         \tl_put_right:Nx \l__enumext_store_current_label_tl
3369         {
3370             \hfill \exp_not:N \hyperlink
3371             {
3372                 \exp_not:V \l__enumext_newlabel_arg_one_tl
3373             }
3374             { \exp_not:V \l__enumext_mark_ref_sym_tl }
3375         }
3376     }
3377     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3378     \bool_if:NT \l__enumext_check_answers_bool
3379     {
3380         \int_gincr:N \g__enumext_item_anskey_int
3381     }
3382 }

```

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

13.36.4 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`.

```

3383 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3384 {
3385     \tl_if_novalue:nF { #1 }
3386     {
3387         \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3388     }
3389 }

```

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.

```

3390 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3391 {
3392     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3393     {
3394         \bool_lazy_or:nnT
3395         { \bool_if_p:N \l__enumext_show_answer_bool }
3396         { \bool_if_p:N \l__enumext_show_position_bool }
3397         {
3398             \__enumext_keyans_wrapper_opt_v:n
3399             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3400         }
3401     }
3402 }

```

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.

```

3403 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3404 {
3405     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3406     {
3407         \bool_lazy_or:nnT
3408         { \bool_if_p:N \l__enumext_show_answer_bool }
3409         { \bool_if_p:N \l__enumext_show_position_bool }
3410         {

```



```

3411         \__enumext_keyans_wrapper_opt_viii:n
3412         { \__enumext_store_current_opt_arg_tl } \c_space_tl
3413     }
3414 }
3415 }

```

(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`.

```

3416 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3417 {
3418     \__enumext_label_width_by_box:Nn
3419     \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3420     \str_case:Vn \l__enumext_align_label_pos_v_str
3421     {
3422         { l }
3423         {
3424             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3425         }
3426         { r }
3427         {
3428             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3429             { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3430         }
3431         { c }
3432         {
3433             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3434             { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3435         }
3436     }

```

Here we set the default values for the key `mark-ans*`, `mark-sep*` and `mark-pos*`.

```

3437     \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3438     {
3439         \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3440     }
3441     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3442     \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3443     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3444 }

```

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

```

3445 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3446 {
3447     \bool_lazy_all:nT
3448     {
3449         { \bool_if_p:N \l__enumext_show_answer_bool }
3450         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3451     }
3452     {
3453         \__enumext_keyans_pos_mark_set:
3454         \__enumext_print_keyans_box:NN
3455         \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3456     }
3457 }

```

The function `__enumext_keyans_show_pos:` will print the `\position` of the stored content in *prop list*. Need add `1` to `\g__enumext_<store name>_prop` for `keyans` environment.

```

3458 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3459 {
3460     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3461     {
3462         \int_incr:N \l__enumext_show_pos_tmp_int
3463     }
3464     {
3465         \int_zero:N \l__enumext_show_pos_tmp_int
3466     }
3467     \bool_lazy_all:nT
3468     {

```

```

3469     { \bool_if_p:N \l__enumext_show_position_bool }
3470     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3471   }
3472   {
3473     \tl_set:Nx \l__enumext_mark_answer_sym_v_tl
3474     {
3475       \group_begin:
3476       \exp_not:N \normalfont
3477       \exp_not:N \footnotesize [ \int_eval:n
3478         {
3479           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3480           + \l__enumext_show_pos_tmp_int
3481         }
3482       ]
3483       \group_end:
3484     }
3485     \__enumext_keyans_pos_mark_set:
3486     \__enumext_print_keyans_box:NN
3487     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3488   }
3489 }

```

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

13.37 Redefining `\item` and `\makelabel` in `enumext`

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makelabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

When *labeling* 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 we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

`__enumext_default_item:n`

First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```

3490 \cs_new_protected:Npn \__enumext_default_item:n #1
3491 {
3492   \tl_if_novalue:nTF {#1}
3493   {
3494     \bool_if:NT \l__enumext_check_answers_bool
3495     {
3496       \int_gincr:N \g__enumext_item_number_int
3497       \bool_set_true:N \l__enumext_item_number_bool
3498     }
3499     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3500     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3501   }
3502   {
3503     \bool_set_eq:cc
3504     { l__enumext_wrap_label_ \__enumext_level: _bool }
3505     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3506     \__enumext_item_std:w [ #1 ] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3507   }
3508 }

```

(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⟩]` works like the *numbered* `\item`, but placing a `⟨symbol⟩` to the “left” of the `⟨label⟩` separated from it by the value the second *optional argument* `⟨offset⟩`.

`#1:` `\l__enumext_item_symbol_X_tl`

`#2:` `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` 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 we will see the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`.

```

3509 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3510 {
3511   \tl_if_novalue:nTF {#1}
3512   {
3513     \tl_gset_eq:Nc
3514     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3515   }
3516   {
3517     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3518   }
3519   \tl_if_novalue:nTF {#2}
3520   {
3521     \dim_set_eq:cc
3522     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3523     { \l__enumext_labelsep_ \__enumext_level: _dim }
3524   }
3525   {
3526     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3527   }
3528   \bool_if:NT \l__enumext_check_answers_bool
3529   {
3530     \int_gincr:N \g__enumext_item_number_int
3531     \bool_set_true:N \l__enumext_item_number_bool
3532   }
3533   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3534   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3535 }

```

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

```

3536 \cs_new_protected:Nn \__enumext_item_starred_exec:
3537 {
3538   \tl_if_empty:cF { \l__enumext_item_symbol_ \__enumext_level: _tl }
3539   {
3540     \mode_leave_vertical:
3541     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3542     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3543     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3544   }
3545 }

```

(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 are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.42).

```

3546 \cs_new_protected:Nn \__enumext_redefine_item:
3547 {
3548   \RenewDocumentCommand \item { s o o }
3549   {
3550     \bool_if:nTF {##1}
3551     {
3552       \__enumext_item_starred_exec:nn {##2} {##3}
3553     }
3554     { \__enumext_default_item:n {##2} }
3555   }
3556 }

```

(End of definition for `__enumext_redefine_item:.`)

`__enumext_make_label:`

`__enumext_make_label_std:`

`__enumext_make_label_box:`

The function `__enumext_make_label:` redefine `\make_label` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.42).

```

3557 \cs_new_protected:Nn \__enumext_make_label:

```

```

3558 {
3559   \IfDocumentMetadataTF
3560   {
3561     \__enumext_make_label_box:
3562   }
3563   {
3564     \bool_if:NTF \__enumext_mode_box_bool
3565     {
3566       \__enumext_make_label_box:
3567     }
3568     {
3569       \__enumext_make_label_std:
3570     }
3571   }
3572 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3573 \cs_new_protected:Nn \__enumext_make_label_std:
3574 {
3575   \RenewDocumentCommand \makeLabel { m }
3576   {
3577     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3578     \__enumext_item_starred_exec:
3579     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3580     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3581     {
3582       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3583     }
3584     { ##1 }
3585     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3586     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3587   }
3588 }

```

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

- Here it is necessary to use `\strut\smash` to maintain text *alignment* in case the user wants to use `\labelbx` for example. In my experiments with *mimicking* the `description` environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic `list` environment package than `enumext`.

```

3589 \cs_new_protected:Nn \__enumext_make_label_box:
3590 {
3591   \RenewDocumentCommand \makeLabel { m }
3592   {
3593     \strut\smash
3594     {
3595       \makebox
3596       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3597       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3598       {
3599         \__enumext_item_starred_exec:
3600         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3601         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3602         {
3603           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3604         }
3605         { ##1 }
3606         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3607       }
3608     } % close smash
3609   }
3610 }

```

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

13.38 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 us to control and set by default the *symbol* and its *offset*.

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

```

item-pos*
3611 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3612 {
3613   \keys_define:nn { enumext / #1 }

```

```

3614     {
3615         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3616         item-sym* .value_required:n = true,
3617         item-sym* .initial:n = {\textborn},
3618         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3619         item-pos* .value_required:n = true,
3620     }
3621 }
3622 \clist_map_inline:nn
3623 {
3624     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3625 }
3626 { \__enumext_tmp:nn #1 }

```

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

13.39 Handling unknown keys

At this point in the code I already know that I will NOT add more *⟨keys⟩* for and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the *⟨keys⟩* (you have to be consistent in life).

- Well, the paragraph above is not so real, after all I had to add more *⟨keys⟩* than I had planned, not everything turns out the way one thinks in life.

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

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

```

3627 \cs_set_protected:Npn \__enumext_tmp:n #1
3628 {
3629     \keys_define:nn { enumext / #1 }
3630     {
3631         unknown .code:n = {
3632             \tl_set:Nn \l__enumext_envir_name_tl {#1}
3633             \__enumext_keyans_unknown_keys:n {##1}
3634         },
3635     }
3636 }
3637 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3638 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3639 {
3640     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3641 }
3642 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3643 {
3644     \tl_if_blank:nTF {#2}
3645     {
3646         \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3647     }
3648     {
3649         \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3650     }
3651 }

```

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

13.39.2 Handling unknown keys for `enumext*`

Define and set `unknown` key for `enumext*` environment.

```

3652 \keys_define:nn { enumext / enumext* }
3653 {
3654     unknown .code:n = {
3655         \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
3656         \__enumext_starred_unknown_keys:n {#1}
3657     },
3658 }

```

Internal functions for handling `unknown` key.

```

3659 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3660 {
3661     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3662 }

```

```

3663 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3664 {
3665   \tl_if_blank:nTF {#2}
3666   {
3667     \msg_error:nne { enumext } { starred-unknown-key } {#1}
3668   }
3669   {
3670     \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}
3671   }
3672 }

```

(End of definition for `unknown`, `__enumext_starred_unknown_keys:n`, and `__enumext_starred_unknown_keys:nn`.)

13.39.3 Handling unknown keys for enumext

`unknown` Defines and set the key `unknown` for `enumext` environment.

```

\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
3673 \cs_set_protected:Npn \__enumext_tmp:n #1
3674 {
3675   \keys_define:nn { enumext / level-#1 }
3676   {
3677     unknown .code:n = {
3678       \int_set:Nn \__enumext_level_int { #1 }
3679       \tl_set:Nn \__enumext_envir_name_tl { enumext }
3680       \__enumext_standar_unknown_keys:n {##1}
3681     },
3682   }
3683 }
3684 \clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3685 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3686 {
3687   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3688 }
3689 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3690 {
3691   \tl_if_blank:nTF {#2}
3692   {
3693     \msg_error:nne { enumext } { standar-unknown-key } {#1}
3694   }
3695   {
3696     \msg_error:nnee { enumext } { standar-unknown-key-value } {#1} {#2}
3697   }
3698 }

```

(End of definition for `unknown`, `__enumext_standar_unknown_keys:n`, and `__enumext_standar_unknown_keys:nn`.)

13.40 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`.

```

3699 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3700 {
3701   \tl_if_novalue:nTF { #1 }
3702   {
3703     \bool_set_true:N \__enumext_wrap_label_v_bool
3704     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3705   }
3706   {
3707     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3708     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3709   }
3710 }

```

(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.

```

3711 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3712 {
3713   \__enumext_keyans_save_item_opt:n { #1 }
3714   \bool_set_true:N \l__enumext_wrap_label_v_bool
3715   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3716   \__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 increments \g__enumext_check_starred_cmd_int for internal check system.

```

3717   \__enumext_keyans_addto_prop:n { #1 }
3718   \__enumext_keyans_store_ref:
3719   \__enumext_keyans_addto_seq:n { #1 }
3720   \int_gincr:N \g__enumext_check_starred_cmd_int
3721 }

```

(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 we will set to true the variable \l__enumext_item_wrap_key_bool used by the wrap-ans* key only when \item* is executed and additionally we need to use \peek_remove_spaces:n to avoid an unwanted space when using \item* together with the itemindent key. This function are passed to __enumext_list_arg_two_v: used in the definition of the **keyans** environment (§13.41).

```

3722 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3723 {
3724   \RenewDocumentCommand \item { s o }
3725   {
3726     \bool_if:nTF {##1}
3727     {
3728       \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3729       \peek_remove_spaces:n
3730       {
3731         \__enumext_keyans_starred_item:n {##2}
3732       }
3733     }
3734     {
3735       \bool_set_false:N \l__enumext_item_wrap_key_bool
3736       \__enumext_keyans_default_item:n {##2}
3737     }
3738   }
3739 }

```

(End of definition for \item* and __enumext_keyans_redefine_item:. This function is documented on page 17.)

__enumext_keyans_make_label:
__enumext_keyans_wrapper_label:n
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makeLabel for the keys mode-box, align, font, wrap-label, wrap-label*, wrap-ans* and \item* for **keyans** environment. This function are passed to __enumext_list_arg_two_v: used in the definition of the **keyans** environment (§13.41).

```

3740 \cs_new_protected:Nn \__enumext_keyans_make_label:
3741 {
3742   \IfDocumentMetadataTF
3743   {
3744     \__enumext_keyans_make_label_box:
3745   }
3746   {
3747     \bool_if:NTF \l__enumext_mode_box_bool
3748     {
3749       \__enumext_keyans_make_label_box:
3750     }
3751     {
3752       \__enumext_keyans_make_label_std:
3753     }
3754   }
3755 }

```


We added conditionals to the `__enumext_keyans_wrapper_label:n` function to handle the keys `wrap-ans*`, `wrap-label` and `wrap-label*`.

```

3756 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3757 {
3758   \bool_lazy_all:nT
3759   {
3760     { \bool_if_p:N \__enumext_wrap_label_v_bool }
3761     { \bool_if_p:N \__enumext_show_answer_bool }
3762     { \bool_if_p:N \__enumext_item_wrap_key_bool }
3763     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3764   }
3765   {
3766     \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3767   }
3768   \bool_if:NTF \__enumext_wrap_label_v_bool
3769   {
3770     \__enumext_wrapper_label_v:n { #1 }
3771   }
3772   { #1 }
3773 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3774 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3775 {
3776   \RenewDocumentCommand \makeLabel { m }
3777   {
3778     \tl_use:N \__enumext_label_fill_left_v_tl
3779     \__enumext_keyans_show_ans:
3780     \__enumext_keyans_show_pos:
3781     \tl_use:N \__enumext_label_font_style_v_tl
3782     \__enumext_keyans_wrapper_label:n { ##1 }
3783     \tl_use:N \__enumext_label_fill_right_v_tl
3784   }
3785 }

```

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

```

3786 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3787 {
3788   \RenewDocumentCommand \makeLabel { m }
3789   {
3790     \strut\smash
3791     {
3792       \makebox[ \__enumext_labelwidth_v_dim ][ \__enumext_align_label_pos_v_str ]
3793       {
3794         \__enumext_keyans_show_ans:
3795         \__enumext_keyans_show_pos:
3796         \tl_use:N \__enumext_label_font_style_v_tl
3797         \__enumext_keyans_wrapper_label:n { ##1 }
3798       }
3799     }
3800   }
3801 }


```

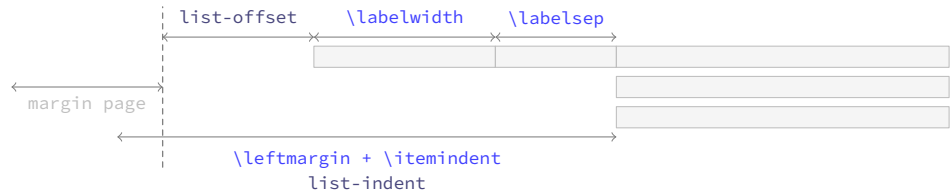
(End of definition for `__enumext_keyans_make_label:` and others.)

13.41 Second argument of the lists

At this point in the code we have already programmed most of the tools needed to create a *custom list* environment, remember that the `__enumext_start_list:nn` function takes two arguments, we have the “first” one ready, the “second” one we will define for all levels of the `enumext` environment, the `keyans` environment and the `enumext*` and `keyans*` environments.

Here we will implement the `__enumext_list_arg_two_X:` function, which will be responsible for setting all the list parameters, the counter, the redefinition of `\item`, `\makeLabel` along with the keys `ref`, `itemindent` and `show-length`.

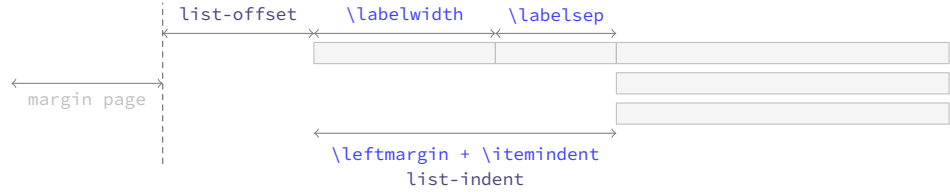
 In the functions `__enumext_list_arg_two_X:` we will implement the “counter” for the environments, but we do NOT set the “start value” for it to be compatible with *tagged* PDF that should be done later.

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

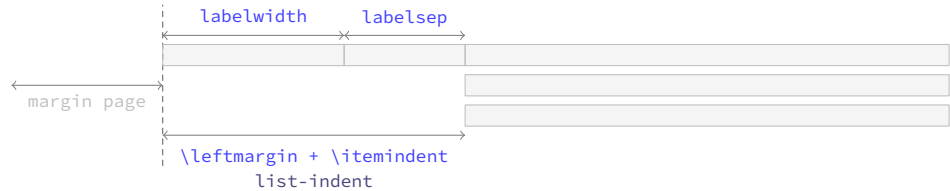
13.41.1 Calculation of `\leftmargin` and `\itemindent`

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

The idea is to have control over these margins so that our 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: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__enumext_leftmargin_tmp_X_bool
```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

```
3802 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3803 {
3804   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3805   {
3806     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3807     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3808   }
3809   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3810   {
3811     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3812     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3813   }
3814 }
```

If no value has been passed to the `labelwidth` and `labelsep` keys we set the default values for `__enumext_leftmargin_tmp_X_dim`.

```
3814 \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

We now analyze the cases and set the values for `\leftmargin` and `\itemindent`.

```
3815 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3816 {
3817   \dim_set:Nn #6 { #1 + #2 - #4 }
3818   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3819 }
3820 {
3821   \dim_compare:nNnT { #4 } = { #1 + #2 }
3822   { \dim_set:Nn #6 { \c_zero_dim } }
```

```

3823 \dim_compare:nNnT { #4 } < { #1 + #2 }
3824 { \dim_set:Nn #6 { #1 + #2 - #4 } }
3825 \dim_compare:nNnT { #4 } > { #1 + #2 }
3826 {
3827     \dim_set:Nn #6 { -#1 - #2 + #4 }
3828     \dim_set:Nn #6 { #6*-1 }
3829 }
3830 \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3831 }
3832 }
3833 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

13.4.1.2 Setting second argument of the lists

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

3834 \cs_set_protected:Npn \__enumext_tmp:n #1
3835 {
3836     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3837     {
3838         \__enumext_calc_hspace:ccccc
3839         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3840         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3841         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3842         { \__enumext_leftmargin_tmp_#1_bool }
3843         \clist_map_inline:nn
3844         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3845         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3846         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3847         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3848         \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3849         { \int_set_eq:cc {@####1} { \__enumext_####1_#1_int } }
3850         \usecounter { enumX#1 }
3851         \str_if_eq:nnTF {#1} { v }
3852         {
3853             \__enumext_keyans_redefine_item:
3854             \__enumext_keyans_make_label:
3855             \__enumext_keyans_ref:
3856             \__enumext_keyans_fake_item_indent:
3857             \bool_if:cT { \__enumext_show_length_#1_bool }
3858             {
3859                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3860             }
3861         }
3862         {
3863             \__enumext_redefine_item:
3864             \__enumext_make_label:
3865             \__enumext_standar_ref:
3866             \__enumext_fake_item_indent:
3867             \bool_if:cT { \__enumext_show_length_#1_bool }
3868             {
3869                 \msg_term:nnne { enumext } { list-lengths } {#1}
3870                 { \int_use:N \__enumext_level_int }
3871             }
3872         }
3873     }
3874 }
3875 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `\opt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `\parskip` locally.

```

3876 \cs_set_protected:Npn \__enumext_tmp:n #1
3877 {
3878     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3879     {
3880         \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }

```

```

3881 \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3882 \__enumext_calc_hspace:ccccc
3883 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3884 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3885 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3886 { \__enumext_leftmargin_tmp_#1_bool }
3887 \clist_map_inline:nn
3888 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3889 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3890 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3891 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3892 \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3893 { \int_set_eq:cc {@####1} { \__enumext_####1_#1_int } }
3894 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3895 \skip_zero:N \partopsep
3896 \usecounter { enumX#1 }
3897 \__enumext_starred_ref:
3898 \str_if_eq:nnTF {#1} { vii }
3899 {
3900   \__enumext_fake_item_indent_vii:
3901   \bool_if:cT { \__enumext_show_length_vii_bool }
3902     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3903 }
3904 {
3905   \__enumext_fake_item_indent_viii:
3906   \bool_if:cT { \__enumext_show_length_#1_bool }
3907     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3908 }
3909 }
3910 }
3911 \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:.)

13.42 The environment enumext

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to “true” if we are NOT nested within enumext*, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to “true” and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to “true” only if the environment is NOT nested and we are at the “first level”.

```

3912 \cs_new_protected:Nn \__enumext_safe_exec:
3913 {
3914   \__enumext_is_not_nested:
3915   \__enumext_internal_mini_page:
3916   \int_incr:N \l__enumext_level_int
3917   \int_compare:nNtT { \l__enumext_level_int } > { 4 }
3918     { \msg_fatal:nn { enumext } { list-too-deep } }
3919   \bool_set_true:N \l__enumext_standar_bool
3920   \bool_set_false:N \l__enumext_starred_bool
3921   \__enumext_is_on_first_level:
3922 }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_name_str used by the key series and then we check if we are at the “first level”, if so we process the ⟨keys⟩ and then execute the function __enumext_parse_series:n used by the key series and call the function __enumext_nested_base_line_fix: used by the key base-fix, otherwise we will pass the ⟨keys⟩ to the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the ⟨keys⟩ to pass them to the sequence if the key save-key is not active.

```

3923 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3924 {
3925   \tl_if_novalue:nF {#1}
3926   {
3927     \str_clear:N \l__enumext_series_name_str
3928     \int_compare:nNtTF { \l__enumext_level_int } = { 1 }
3929     {
3930       \keys_set:nn { enumext / level-1 } {#1}
3931       \bool_if:NF \l__enumext_print_keyans_cmd_bool

```

```

3932         {
3933             \__enumext_parse_series:n {#1}
3934         }
3935         \__enumext_nested_base_line_fix:
3936     }
3937     {
3938         \exp_args:Ne \keys_set:nn
3939         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3940         \bool_if:NF \l__enumext_print_keyans_cmd_bool
3941         {
3942             \__enumext_parse_series:n {#1}
3943         }
3944     }
3945     \__enumext_store_active_keys:n {#1}
3946 }
3947 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level: The __enumext_start_store_level: function activate the “*storing structure*” mechanism in the *sequence* for the command \anskey and the environment anskey*.

```

3948 \cs_new_protected:Nn \__enumext_start_store_level:
3949 {
3950     \bool_lazy_all:nT
3951     {
3952         { \bool_if_p:N \l__enumext_store_active_bool }
3953         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3954         { \bool_if_p:N \g__enumext_standar_bool }
3955     }
3956     {
3957         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3958         {
3959             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3960             \__enumext_store_level_open:
3961         }
3962     }

```

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

```

3963     \bool_lazy_all:nT
3964     {
3965         { \bool_if_p:N \l__enumext_store_active_bool }
3966         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3967         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3968     }
3969     {
3970         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3971         {
3972             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3973             \__enumext_store_level_open:
3974         }
3975     }
3976 }

```

(End of definition for __enumext_start_store_level:.)

__enumext_stop_store_level: The __enumext_stop_store_level: function stop the “*storing structure*” mechanism in the *sequence* for the command \anskey and the environment anskey*.

```

3977 \cs_new_protected:Nn \__enumext_stop_store_level:
3978 {
3979     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3980     {
3981         \__enumext_store_level_close:
3982     }
3983 }

```

(End of definition for __enumext_stop_store_level:.)

__enumext_multicols_start: The function __enumext_multicols_start: will start the multicols environment according to the value passed by the columns key, then set the default value for \columnsep when columns-sep=opt and set the value of \multicolsep equal to zero and leave \columnseprule equal to zero for inner levels.

```

3984 \cs_new_protected:Nn \__enumext_multicols_start:

```

```

3985 {
3986   \int_compare:nNt
3987   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3988   {
3989     \dim_compare:nNt
3990     { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3991     {
3992       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
3993       {
3994         ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
3995           + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3996         ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3997         - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3998       }
3999     }
4000     \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
4001     \int_compare:nNt { \__enumext_level_int } > { 1 }
4002     {
4003       \dim_zero:N \columnseprule
4004     }

```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `__enumext_multi_addvspace:`, apply our “*vertical adjust spacing*”, then start the `multicols` environment.

```

4005   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
4006   {
4007     \skip_zero:N \multicolsep
4008     \__enumext_multi_addvspace:
4009   }
4010   \raggedcolumns
4011   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
4012 }
4013 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment and apply our “*vertical adjust*” spacing. For compatibility with *tagged* PDF, the closing of the `list` environment is executed here along with `__enumext_stop_store_level:`.

```

4014 \cs_new_protected:Nn \__enumext_multicols_stop:
4015 {
4016   \int_compare:nNtF
4017   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
4018   {
4019     \__enumext_stop_list:
4020     \__enumext_stop_store_level:
4021     \end{multicols}
4022     \__enumext_unskip_unkern:
4023     \__enumext_unskip_unkern:
4024     \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
4025   }
4026   {
4027     \__enumext_stop_list:
4028     \__enumext_stop_store_level:
4029   }
4030 }

```

(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.

```

4031 \cs_new_protected:Nn \__enumext_before_list:
4032 {
4033   \__enumext_vspace_above:
4034   \__enumext_before_args_exec:
4035   \__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.

```

4036 \dim_compare:nNtT
4037 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
4038 {
4039   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
4040   {
4041     \linewidth
4042     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
4043     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
4044   }

```

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.

```

4045   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
4046   \int_gincr:N \g__enumext_minipage_stat_int
4047   \__enumext_minipage_add_space:
4048   \noindent
4049   \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
4050 }
4051 \__enumext_multicols_start:
4052 }

```

(End of definition for `__enumext_before_list:`)

`__enumext_second_part:` The function `__enumext_second_part:` first check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_page` environment has not been closed), then close `__enumext_mini_page` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

4053 \cs_new_protected:Nn \__enumext_second_part:
4054 {
4055   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
4056   {
4057     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
4058     {
4059       \msg_warning:nn { enumext } { missing-miniright }
4060       \miniright
4061     }
4062     \int_gzero:N \g__enumext_minipage_stat_int
4063     \__enumext_unskip_unkern: % remove topsep + [partopsep]
4064     \end__enumext_mini_page
4065   }
4066   {
4067     \__enumext_multicols_stop:
4068   }

```

Now we will execute the functions `__enumext_after_stop_list:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below:` used by the keys `below` and `below*`. Finally set `\l__enumext_standar_bool` to false and call the function `__enumext_resume_save_counter:` used by the `series`, `resume` and `resume*` keys.

```

4069   \__enumext_after_stop_list:
4070   \__enumext_check_ans_key_hook:
4071   \__enumext_vspace_below:
4072   \bool_set_false:N \l__enumext_standar_bool
4073   \bool_if:NF \l__enumext_print_keyans_cmd_bool
4074   {
4075     \__enumext_standar_save_counter:
4076   }
4077 }

```

(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.

```

4078 \cs_new_protected:Nn \__enumext_set_item_width:
4079 {
4080   \dim_set:Nn \itemwidth { \linewidth }
4081   \dim_compare:nT

```



```

4082     {
4083         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
4084     }
4085     {
4086         \dim_sub:Nn \itemwidth
4087         {
4088             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
4089         }
4090     }
4091 }

```

(End of definition for __enumext_set_item_width:.)

__enumext_start_counter: For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of `\item`, i.e. `\begin{list}{\langle arg one \rangle}{\langle arg two \rangle}\setcounter{enumX}`.

- This is described in [processing order of legacysetupcode in the block templates](#) and we will apply the workaround provided by Frank Mittelbach.

```

4092 \cs_new_protected:Nn \__enumext_start_counter:
4093 {
4094     \setcounter { enumX \__enumext_level: }
4095     {
4096         \int_eval:n { \int_use:c { l__enumext_start_ \__enumext_level: _int } - 1 }
4097     }
4098 }

```

(End of definition for __enumext_start_counter:.)

enumext Now create the `enumext` environment based on `list` environment by levels.

```

4099 \NewDocumentEnvironment{enumext}{0}{ }
4100 {
4101     \__enumext_safe_exec:
4102     \__enumext_parse_keys:n {#1}
4103     \__enumext_before_list:
4104     \__enumext_start_store_level:
4105     \__enumext_start_list:nn
4106     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
4107     {
4108         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
4109         \__enumext_before_keys_exec:
4110     }
4111     \__enumext_start_counter:
4112     \__enumext_set_item_width:
4113     \__enumext_after_args_exec:
4114 }
4115 {
4116     \__enumext_second_part:
4117 }

```

(End of definition for `enumext`. This function is documented on page 5.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext` environment using the “hook” function `__enumext_after_env:nn`.

```

4118 \__enumext_after_env:nn {enumext}
4119 {
4120     \__enumext_execute_after_env:
4121 }

```

13.43 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”.

__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. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```

4122 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
4123 {
4124     \bool_if:NF \l__enumext_store_active_bool
4125     {

```

```

4126         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
4127     }
4128     \int_incr:N \l__enumext_keyans_level_int
4129     \bool_set_true:N \l__enumext_keyans_env_bool
4130     \__enumext_keyans_name_and_start:
4131     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
4132     \bool_set_false:N \l__enumext_store_active_bool
4133     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
4134     {
4135         \msg_error:nn { enumext } { keyans-nested }
4136     }
4137     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4138     {
4139         \msg_error:nn { enumext } { keyans-wrong-level }
4140     }
4141 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

`__enumext_keyans_parse_keys:n` Parse [`key = val`] for `keyans` environment.

```

4142 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
4143 {
4144     \keys_set:nn { enumext / keyans } {#1}
4145 }

```

(End of definition for `__enumext_keyans_parse_keys:n`.)

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

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_second_part_v:
4146 \cs_new_protected:Nn \__enumext_before_list_v:
4147 {
4148     \__enumext_vspace_above_v:
4149     \__enumext_before_args_exec_v:
4150     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
4151     {
4152         \dim_set:Nn \l__enumext_minipage_left_v_dim
4153         {
4154             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
4155         }
4156         \bool_set_true:N \l__enumext_minipage_active_v_bool
4157         \int_gincr:N \g__enumext_minipage_stat_int
4158         \__enumext_keyans_minipage_add_space:
4159         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
4160     }
4161     \__enumext_keyans_multicols_start:
4162 }
4163 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
4164 {
4165     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
4166     {
4167         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
4168         {
4169             \dim_set:Nn \l__enumext_columns_sep_v_dim
4170             {
4171                 (
4172                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
4173                 ) / \l__enumext_columns_v_int
4174                 - \l__enumext_listoffset_v_dim
4175             }
4176         }
4177         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
4178         \dim_zero:N \columnseprule % no rule here
4179         \bool_if:NF \l__enumext_minipage_active_v_bool
4180         {
4181             \skip_zero:N \multicolsep
4182             \__enumext_keyans_multi_addvspace:
4183         }
4184         \raggedcolumns
4185         \begin{multicols}{\l__enumext_columns_v_int}
4186     }
4187 }
4188 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:

```

```

4189 {
4190   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
4191   {
4192     \__enumext_stop_list:
4193     \end{multicols}
4194     \__enumext_unskip_unkern:
4195     \__enumext_unskip_unkern:
4196     \par\addvspace{ \l__enumext_multicols_below_v_skip }
4197   }
4198   {
4199     \__enumext_stop_list:
4200   }
4201 }
4202 \cs_new_protected:Nn \__enumext_second_part_v:
4203 {
4204   \bool_if:NTF \l__enumext_minipage_active_v_bool
4205   {
4206     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4207     {
4208       \msg_warning:nn { enumext } { missing-miniright }
4209       \miniright
4210     }
4211     \int_gzero:N \g__enumext_minipage_stat_int
4212     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4213     \end__enumext_mini_page
4214     \par\addvspace{ \l__enumext_minipage_after_skip }
4215   }
4216   {
4217     \__enumext_keyans_multicols_stop:
4218   }
4219   \bool_set_false:N \l__enumext_keyans_env_bool
4220   \__enumext_after_stop_list_v:
4221   \__enumext_vspace_below_v:
4222 }

```

(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.

```

4223 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4224 {
4225   \dim_set:Nn \itemwidth { \linewidth }
4226   \dim_compare:nT
4227   {
4228     \l__enumext_listoffset_v_dim != \c_zero_dim
4229   }
4230   {
4231     \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4232   }
4233 }

```

(End of definition for __enumext_keyans_set_item_width:.)

__enumext_keyans_start_counter:

For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of \item, i.e. \begin{list}{\langle arg one \rangle}{\langle arg two \rangle}\setcounter{enumX}.

```

4234 \cs_new_protected:Nn \__enumext_keyans_start_counter:
4235 {
4236   \setcounter { enumXv } { \int_eval:n { \int_use:c { \l__enumext_start_v_int } - 1 } }
4237 }

```

(End of definition for __enumext_keyans_start_counter:.)

keyans Now we define the environment **keyans** also based on lists.

```

4238 \NewDocumentEnvironment{keyans}{0}{}
4239 {
4240   \__enumext_keyans_safe_exec:
4241   \__enumext_keyans_parse_keys:n {#1}
4242   \__enumext_before_list_v:
4243   \__enumext_start_list:nn
4244   { \tl_use:N \l__enumext_label_v_tl }

```

```

4245     {
4246         \__enumext_list_arg_two_v:
4247         \__enumext_before_keys_exec_v:
4248     }
4249     \__enumext_keyans_start_counter:
4250     \__enumext_keyans_set_item_width:
4251     \__enumext_after_args_exec_v:
4252 }
4253 {
4254     \__enumext_check_starred_cmd:n { item }
4255     \__enumext_second_part_v:
4256 }

```

(End of definition for `keyans`. This function is documented on page 16.)

13.44 Tagging PDF support for non-standart list environments

The \TeX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the expected 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`[20]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my \item redefinition to be compatible with tagging-pdf](#).

13.44.1 Socket for tagging support in `enumext*` and `keyans*`

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```

start-list-tags
stop-start-tags
stop-list-tags
\__enumext_start_list_tag:n
\__enumext_stop_start_list_tag:
\__enumext_stop_list_tag:n
4257 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4258 \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
4259 {
4260     \tag_resume:n {#1}
4261     \tag_mc_end_push:
4262     \tag_struct_begin:n {tag=LI}
4263     \tag_struct_begin:n {tag=Lbl}
4264     \tag_mc_begin:n {tag=Lbl}
4265 }
4266 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4267 {
4268     \tag_mc_end:
4269     \tag_struct_end:n {tag=Lbl}
4270     \tag_struct_begin:n {tag=LBody}
4271     \tag_struct_begin:n {tag=text-unit}
4272     \tag_struct_begin:n {tag=text}
4273 }
4274 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4275 {
4276     \tag_struct_end:n {tag=text}
4277     \tag_struct_end:n {tag=text-unit}
4278     \tag_struct_end:n {tag=LBody}
4279     \tag_struct_end:n {tag=LI}
4280     \tag_mc_begin_pop:n {}
4281     \tag_suspend:n {#1}
4282 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

4283 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4284 {
4285     \IfDocumentMetadataT
4286     {
4287         \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4288         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4289     }
4290 }
4291 \cs_new_protected_nopar:Npn \__enumext_stop_start_list_tag:
4292 {
4293     \IfDocumentMetadataT
4294     {
4295         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4296         \socket_use:nn {tagsupport/__enumext/starred} {}
4297     }
4298 }

```

```

4299 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4300 {
4301   \IfDocumentMetadataT
4302   {
4303     \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4304     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4305   }
4306 }

```

(End of definition for start-list-tags and others.)

13.44.2 Socket for tagging support in keyanspic

We will first define the necessary sockets and their behavior for `keyanspic` environment.

```

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:
4307 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4308 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4309 {
4310   \tag_resume:n {keyanspic}
4311   \tag_mc_end_push:
4312     \tag_struct_begin:n {tag=LI}
4313     \tag_struct_begin:n {tag=Lbl}
4314     \tag_mc_begin:n {tag=Lbl}
4315 }
4316 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4317 {
4318   \tag_mc_end:
4319   \tag_struct_end:n {tag=Lbl}
4320   \tag_struct_begin:n {tag=LBody}
4321   \tag_struct_begin:n {tag=text-unit}
4322   \tag_struct_begin:n {tag=text}
4323   \tag_mc_begin:n {tag=text}
4324 }
4325 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4326 {
4327   \tag_mc_end:
4328   \tag_struct_end:n {tag=text}
4329   \tag_struct_end:n {tag=text-unit}
4330   \tag_struct_end:n {tag=LBody}
4331   \tag_struct_end:n {tag=LI}
4332   \tag_mc_begin_pop:n {}
4333   \tag_suspend:n {keyanspic}
4334 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

4335 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4336 {
4337   \IfDocumentMetadataT
4338   {
4339     \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4340     \socket_use:n {tagsupport/__enumext/keyanspic}
4341   }
4342 }
4343 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4344 {
4345   \IfDocumentMetadataT
4346   {
4347     \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4348     \socket_use:n {tagsupport/__enumext/keyanspic}
4349   }
4350 }
4351 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4352 {
4353   \IfDocumentMetadataT
4354   {
4355     \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4356     \socket_use:n {tagsupport/__enumext/keyanspic}
4357   }
4358 }

```

(End of definition for start-list-tags and others.)

13.45 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.

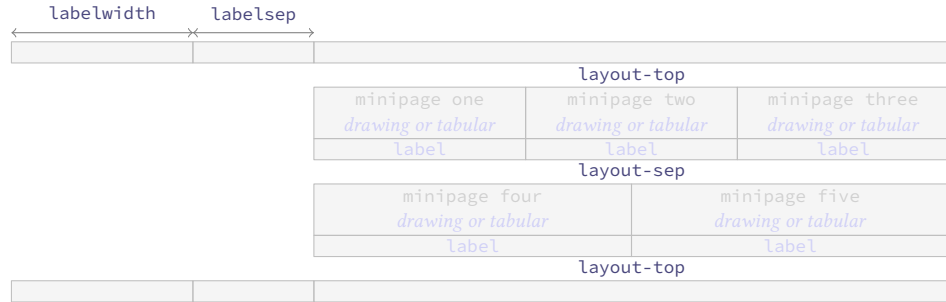


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

13.45.1 The environment `keyanspic`

First we define the key that allows us to process the position of the `\label` centered “*above*” or “*below*” which will be `label-pos`, the vertical separation of these from *drawing or tabular* will be handled with the key `label-sep`. The “*layout style*” will be handled with the key `layout-sty` will take two values separated by comma `{(n° upper, n° lower)}` and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “*upper*” and “*lower*” within the environments separated by the value of the key `layout-sep`. The vertical space “*top*” and “*bottom*” of the environment will be handled with the key `layout-top`.

```

label-pos  First we define the key that allows us to process the position of the \label centered “above” or “below” which
label-sep  will be label-pos, the vertical separation of these from drawing or tabular will be handled with the key
layout-sty label-sep. The “layout style” will be handled with the key layout-sty will take two values separated
layout-sep by comma {(n° upper, n° lower)} and will determine the number of minipage environments in which all
layout-top arguments of \anspic will be printed at the “upper” and “lower” within the environments separated by the
mark-ans   value of the key layout-sep. The vertical space “top” and “bottom” of the environment will be handled with
mark-pos   the key layout-top.
mark-sep
save-sep
wrap-opt
wrap-ans*
show-ans
show-pos

4359 \keys_define:nn { enumext / keyanspic }
4360 {
4361   label-pos .choice:,
4362   label-pos / above .code:n =
4363     \bool_set_true:N \l__enumext_anspic_label_above_bool
4364     \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4365   label-pos / below .code:n =
4366     \bool_set_false:N \l__enumext_anspic_label_above_bool
4367     \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4368   label-pos / unknown .code:n =
4369     \msg_error:nnee { enumext } { unknown-choice }
4370     { label-pos } { above,~ below } { \exp_not:n {#1} },
4371   label-pos .initial:n = below,
4372   label-pos .value_required:n = true,
4373   label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4374   label-sep .value_required:n = true,
4375   layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4376   layout-sty .value_required:n = true,
4377   layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4378   layout-sep .value_required:n = true,
4379   layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4380   layout-top .value_required:n = true,
4381   mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4382   mark-ans .value_required:n = true,
4383   mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4384   mark-pos .value_required:n = true,
4385   mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4386   mark-sep .value_required:n = true,
4387   save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4388   save-sep .value_required:n = true,
4389   wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4390   wrap-opt .value_required:n = true,
4391   wrap-ans* .code:n = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4392   wrap-ans* .value_required:n = true,
4393   show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4394   show-ans .value_required:n = true,
4395   show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4396   show-pos .value_required:n = true,

```

```

4397     unknown      .code:n      = {
4398                                     \tl_set:Nn \__enumext_envir_name_tl { keyanspic }
4399                                     \__enumext_keyans_unknown_keys:n {#1}
4400                                 },
4401     }

```

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

The function `__enumext_keyans_pic_safe_exec:` check the nested level position inside the `enumext` environment.

```

4402 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4403 {
4404     \int_incr:N \__enumext_keyans_pic_level_int
4405     \int_compare:nNtT { \__enumext_keyans_pic_level_int } > { 1 }
4406     {
4407         \msg_error:nn { enumext } { keyanspic-nested }
4408     }
4409     \__enumext_keyans_name_and_start:
4410 }

```

Parse [`key = val`] for `keyanspic` environment.

```

4411 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4412 {
4413     \tl_if_novalue:nF {#1}
4414     {
4415         \keys_set:nn { enumext / keyanspic } {#1}
4416     }
4417 }

```

The function `__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```

4418 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4419 {
4420     \dim_compare:nNtT { #1 } < { \c_zero_dim }
4421     {
4422         \skip_set:Nn #1 { -#1 }
4423     }
4424 }

```

The `__enumext_keyans_pic_arg_two:` function will be used in the *second argument* of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the “*spaces*” and the keys `label`, `wrap-label`, `parsep` and `topsep` from the `keyans` environment. The first thing we need to do is set the boolean variable `__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.

```

4425 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4426 {
4427     \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
4428     \__enumext_list_arg_two_v:
4429     \__enumext_keyans_pic_skip_abs:N \parsep

```

Now we increment the counter `enumXv` of the `keyans` environment and save the *total height* of the `(label)` in `__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the key `label-pos` is set to *below*.

```

4430     \bool_if:nF \__enumext_anspic_label_above_bool
4431     {
4432         \stepcounter { enumXv }
4433         \hbox_set:Nn \__enumext_anspic_label_box { \__enumext_label_v_tl }
4434         \dim_set:Nn \__enumext_anspic_label_htdp_dim
4435         {
4436             \box_ht_plus_dp:N \__enumext_anspic_label_box
4437         }
4438         \skip_add:Nn \parsep
4439         {
4440             \__enumext_anspic_label_htdp_dim
4441             + \box_dp:N \strutbox
4442             + \__enumext_anspic_label_sep_skip
4443         }
4444     }

```


Finally we *adjust* the value of `\leftmargin` and `\topsep` then set `\listparindent`, `\partopsep` and `\itemsep` to zero so that the *horizontal* and *vertical* space is not affected.

```

4445 \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4446 \ignorespaces
4447 \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4448 \dim_zero:N \listparindent
4449 \skip_zero:N \partopsep
4450 \skip_zero:N \itemsep
4451 }

```

(End of definition for `__enumext_keyans_pic_safe_exec`: and others.)

keyanspic Now we define the environment `keyanspic`. For compatibility with *tagged* PDF we must use the `\begin{list}` form and a lot of conditional code using `\IfDocumentMetadataTF`. We will first stop the code for automatic *tagged* PDF for `list` environments, redefine `\item` so that it cannot be used, and stop the code for automatic *tagged* PDF for the `keyanspic` environment.

```

4452 \NewDocumentEnvironment{keyanspic}{ o }
4453 {
4454   \__enumext_keyans_pic_safe_exec:
4455   \__enumext_keyans_pic_parse_keys:n {#1}
4456   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4457   \IfDocumentMetadataT
4458   {
4459     \tag_suspend:n {list}
4460   }
4461   \item[] \scan_stop:
4462   \RenewDocumentCommand \item {}
4463   {
4464     \msg_error:nn { enumext } { keyanspic-item-cmd }
4465   }
4466   \IfDocumentMetadataT
4467   {
4468     \tag_resume:n {keyanspic}
4469     \tag_tool:n {para/tagging=false}
4470     \tag_suspend:n {keyanspic}
4471   }
4472 }

```

In the second part of the environment definition we will manually place our code for *tagged* PDF and execute the command `\anspic` using the `__enumext_anspic_exec`: function.

```

4473 {
4474   \IfDocumentMetadataT
4475   {
4476     \tag_resume:n {keyanspic}
4477     \tag_mc_end_push:
4478     \tag_struct_begin:n {tag=L,attribute=enumerate}
4479   }
4480   \__enumext_anspic_exec:
4481   \IfDocumentMetadataT
4482   {
4483     \tag_suspend:n {keyanspic}
4484   }
4485   \end{list}
4486   \IfDocumentMetadataT
4487   {
4488     \tag_struct_end:n {tag=L}
4489     \tag_mc_begin_pop:n {}
4490     \tag_struct_end:n {tag=L}
4491     \tag_mc_begin_pop:n {}
4492   }

```

Finally we check if `\anspic*` has been used, set the counter `enumXvi` to zero and apply our “adjusted” vertical space bottom.

```

4493 \__enumext_check_starred_cmd:n { anspic }
4494 \setcounter { enumXvi } { 0 }
4495 \bool_if:NTF \l__enumext_anspic_label_above_bool
4496 {
4497   \par\addvspace{ 0.5\box_dp:N \strutbox }
4498 }
4499 {
4500   \par

```

```

4501         \addvspace
4502         {
4503             \dim_eval:n
4504             {
4505                 \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4506                 + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4507             }
4508         }
4509     }
4510 }

```

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

13.45.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*[\langle content \rangle]` store the current `\label` next to the *optional argument* `[\langle content \rangle]` in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* `{\langle drawing or tabular \rangle}` 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.

`\anspic`

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. 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.

```

4511 \NewDocumentCommand \anspic { s o +m }
4512 {
4513     \bool_if:NF \l__enumext_store_active_bool
4514     {
4515         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4516     }
4517     \int_compare:nNt { \l__enumext_level_int } > { 1 }
4518     {
4519         \msg_error:nn { enumext } { keyanspic-wrong-level }
4520     }
4521     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4522     {
4523         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4524     }
4525     \seq_put_right:Nn \l__enumext_anspic_args_seq
4526     {
4527         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4528     }
4529 }

```

The `__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4530 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4531 {
4532     \bool_if:NF \l__enumext_anspic_label_above_bool
4533     {
4534         \IfDocumentMetadataT
4535         {
4536             \tag_suspend:n {keyanspic}
4537         }
4538         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4539         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4540         {
4541             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4542         }
4543         \IfDocumentMetadataT
4544         {
4545             \tag_resume:n {keyanspic}
4546         }
4547     }
4548 }

```

The `__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* ‘*’ and *optional argument* passed to the command. Here we will store the `\label` and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label`, `wrap-ans*` and `wrap-opt` keys.

```

4549 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4550 {
4551   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4552   {
4553     \bool_if:NTF { #1 }
4554     {
4555       \bool_set_true:N \l__enumext_item_wrap_key_bool
4556       \bool_set_true:N \l__enumext_wrap_label_v_bool
4557       \__enumext_keyans_save_item_opt:n { #2 }
4558       \__enumext_keyans_addto_prop:n { #2 }
4559       \__enumext_keyans_store_ref:
4560       \__enumext_keyans_addto_seq:n { #2 }
4561       \int_gincr:N \g__enumext_check_starred_cmd_int
4562       \__enumext_keyans_show_ans:
4563       \__enumext_keyans_show_pos:
4564       \makebox[ \l__enumext_labelwidth_v_dim ][ c ]
4565       {
4566         \tl_use:N \l__enumext_label_font_style_v_tl
4567         \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4568       }
4569       \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4570       \__enumext_keyans_show_item_opt:
4571     }
4572     {
4573       \bool_set_false:N \l__enumext_item_wrap_key_bool
4574       \tl_use:N \l__enumext_label_font_style_v_tl
4575       \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4576     }
4577   }
4578 }

```

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.

```

4579 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4580 {
4581   \stepcounter { enumXvi }
4582   \__enumext_anspic_body_dim:n { #3 }
4583   \bool_if:NTF \l__enumext_anspic_label_above_bool
4584   {
4585     \__enumext_anspic_label:nn { #1 } { #2 }
4586   }
4587   {
4588     \raisebox
4589     {
4590       -\dim_eval:n
4591       {
4592         \l__enumext_anspic_label_htdp_dim
4593         + \l__enumext_anspic_body_htdp_dim
4594         + \box_dp:N \strutbox
4595         + \l__enumext_anspic_label_sep_skip
4596       }
4597     }
4598     [ opt ] [ opt ]
4599     {
4600       \__enumext_anspic_label:nn { #1 } { #2 }
4601     }
4602   }
4603 }
4604 %

```

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.

```

4605 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4606 {
4607   \__enumext_anspic_start_list_tag:
4608   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4609   \__enumext_anspic_stop_start_list_tag:
4610   \bool_if:NTF \l__enumext_anspic_label_above_bool
4611   {
4612     \[\l__enumext_anspic_label_sep_skip] #3
4613   }

```

```

4614     {
4615         \\ #3
4616     }
4617     \__enumext_anspic_stop_list_tag:
4618 }

```

The value $\langle n^{\circ} upper, n^{\circ} lower \rangle$ passed to the `layout-sty` key is split by comma and is handled directly by the function `__enumext_anspic_print:n` and passed to the function `__enumext_anspic_row:n`.

```

4619 \cs_new_protected:Nn \__enumext_anspic_print:n
4620 {
4621     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4622 }
4623 \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.

```

4624 \cs_new_protected:Nn \__enumext_anspic_row:n
4625 {
4626     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4627     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4628     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4629     \int_step_inline:nnn
4630     { \l__enumext_anspic_above_int + 1 }
4631     { \l__enumext_anspic_below_int }
4632     {
4633         \IfDocumentMetadataT
4634         {
4635             \tag_suspend:n {minipage}
4636         }
4637         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4638             \centering
4639             \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4640             \end{minipage}
4641         \IfDocumentMetadataT
4642         {
4643             \tag_resume:n {minipage}
4644         }
4645     }
4646     \par
4647 }

```

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*.

```

4648 \cs_new_protected:Nn \__enumext_anspic_exec:
4649 {
4650     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4651     {
4652         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4653     }
4654     {
4655         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4656     }
4657 }

```

(End of definition for `\anspic` and others. This function is documented on page 18.)

13.46 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 `shortlist` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makeLabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the $\langle content \rangle$ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`, we will also add the *optional argument* $\langle number \rangle$ to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* $\langle number \rangle$.

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by \LaTeX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`,

`center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.46.1 Functions for item box width

We set the default value for the *width of the box* containing the *content* of the items for `enumext*` environment.

```
\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
4658 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4659 {
4660   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4661   {
4662     \dim_set:Nn \__enumext_columns_sep_vii_dim
4663     {
4664       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4665       / \__enumext_columns_vii_int
4666     }
4667   }
4668   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4669   \dim_set:Nn \__enumext_item_width_vii_dim
4670   {
4671     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4672     / \__enumext_columns_vii_int
4673     - \__enumext_labelwidth_vii_dim
4674     - \__enumext_labelsep_vii_dim
4675   }
```

When the key `rightmargin` is active we must adjust the values.

```
4676   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4677   {
4678     \dim_sub:Nn \__enumext_item_width_vii_dim
4679     {
4680       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4681       / \__enumext_columns_vii_int
4682     }
4683     \dim_add:Nn \__enumext_columns_sep_vii_dim
4684     {
4685       \__enumext_rightmargin_vii_dim
4686     }
4687   }
4688 }
```

Same implementation for the `keyans*` environment.

```
4689 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4690 {
4691   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4692   {
4693     \dim_set:Nn \__enumext_columns_sep_viii_dim
4694     {
4695       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
4696       / \__enumext_columns_viii_int
4697     }
4698   }
4699   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
4700   \dim_set:Nn \__enumext_item_width_viii_dim
4701   {
4702     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
4703     / \__enumext_columns_viii_int
4704     - \__enumext_labelwidth_viii_dim
4705     - \__enumext_labelsep_viii_dim
4706   }
4707   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4708   {
4709     \dim_sub:Nn \__enumext_item_width_viii_dim
4710     {
4711       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_viii_int )
4712       / \__enumext_columns_viii_int
4713     }
4714   }
```

```

4714         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4715         {
4716             \l__enumext_rightmargin_viii_dim
4717         }
4718     }
4719 }

```

(End of definition for `__enumext_starred_columns_set_vii:` and `__enumext_starred_columns_set_viii:`)

13.46.2 Functions for join item columns

```

\__enumext_starred_joined_item_vii:n
\__enumext_starred_joined_item_viii:n

```

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the `\item`(`\columns`) will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4720 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4721 {
4722     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4723     \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4724     {
4725         \msg_warning:nnee { enumext } { item-joined }
4726         { \int_use:N \l__enumext_joined_item_vii_int }
4727         { \int_use:N \l__enumext_columns_vii_int }
4728         \int_set:Nn \l__enumext_joined_item_vii_int
4729         {
4730             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4731         }
4732     }
4733     \int_compare:nNt
4734     { \l__enumext_joined_item_vii_int }
4735     >
4736     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4737     {
4738         \msg_warning:nnee { enumext } { item-joined-columns }
4739         { \int_use:N \l__enumext_joined_item_vii_int }
4740         {
4741             \int_eval:n
4742             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4743         }
4744         \int_set:Nn \l__enumext_joined_item_vii_int
4745         {
4746             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4747         }
4748     }
4749     \int_compare:nNtF { \l__enumext_joined_item_vii_int } > { 1 }
4750     {
4751         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4752         \int_decr:N \l__enumext_joined_item_aux_vii_int
4753         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4754         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4755         \dim_set:Nn \l__enumext_joined_width_vii_dim
4756         {
4757             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4758             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4759                 + \l__enumext_columns_sep_vii_dim
4760                 ) * \l__enumext_joined_item_aux_vii_int
4761         }
4762         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4763     }
4764     {
4765         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4766         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4767     }
4768 }

```

Same implementation for the `keyans*` environment.

```

4769 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4770 {
4771     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4772     \int_compare:nNt { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4773     {
4774         \msg_warning:nnee { enumext } { item-joined }
4775         { \int_use:N \l__enumext_joined_item_viii_int }

```

```

4776         { \int_use:N \l__enumext_columns_viii_int }
4777     \int_set:Nn \l__enumext_joined_item_viii_int
4778     {
4779         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4780     }
4781 }
4782 \int_compare:nNnT
4783 { \l__enumext_joined_item_viii_int }
4784 >
4785 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4786 {
4787     \msg_warning:nnee { enumext } { item-joined-columns }
4788     { \int_use:N \l__enumext_joined_item_viii_int }
4789     {
4790         \int_eval:n
4791         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4792     }
4793     \int_set:Nn \l__enumext_joined_item_viii_int
4794     {
4795         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4796     }
4797 }
4798 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4799 {
4800     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4801     \int_decr:N \l__enumext_joined_item_aux_viii_int
4802     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4803     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4804     \dim_set:Nn \l__enumext_joined_width_viii_dim
4805     {
4806         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4807         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4808             + \l__enumext_columns_sep_viii_dim
4809             ) * \l__enumext_joined_item_aux_viii_int
4810     }
4811     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4812 }
4813 {
4814     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4815     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4816 }
4817 }

```

(End of definition for `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n`)

13.46.3 Functions for mini-env, mini-right and mini-right* keys

`__enumext_start_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`.

```

4818 \cs_new_protected:Nn \__enumext_start_mini_vii:
4819 {
4820     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4821     {
4822         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4823         {
4824             \linewidth
4825             - \l__enumext_minipage_right_vii_dim
4826             - \l__enumext_minipage_hsep_vii_dim
4827         }
4828         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4829         \dim_gset_eq:NN
4830         \g__enumext_minipage_right_vii_dim
4831         \l__enumext_minipage_right_vii_dim
4832         \__enumext_mini_addvspace_vii:
4833         \nointerlineskip\noindent
4834         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4835     }
4836 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will

be used in the function `__enumxt_after_env:n` to execute the `minipage` on the “*right side*”. At this point we will execute the `__enumxt_stop_list:` and `__enumxt_stop_store_level_vii:` functions 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 `__enumxt_after_list_vii:` function in the second part of the `enumxt*` environment definition (§13.47).

```

4837 \cs_new_protected:Nn \__enumxt_stop_mini_vii:
4838 {
4839   \bool_if:NTF \l__enumxt_minipage_active_vii_bool
4840   {
4841     \__enumxt_stop_list:
4842     \__enumxt_stop_store_level_vii:
4843     \IfDocumentMetadataT { \tag_resume:n {enumxt*} }
4844     \end__enumxt_mini_page
4845     \hfill
4846     \bool_gset_true:N \g__enumxt_minipage_active_vii_bool
4847   }
4848   {
4849     \__enumxt_stop_list:
4850     \__enumxt_stop_store_level_vii:
4851   }
4852 }

```

(End of definition for `__enumxt_start_mini_vii:` and `__enumxt_stop_mini_vii:`.)

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumxt_miniright_code_vii_tl` in the `minipage` environment on the “*right side*”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

4853 \__enumxt_after_env:n {enumxt*}
4854 {
4855   \bool_if:NT \g__enumxt_minipage_active_vii_bool
4856   {
4857     \__enumxt_minipage:w [ t ] { \g__enumxt_minipage_right_vii_dim }
4858     \legacy_if_gset_false:n { @minipage }
4859     \skip_vertical:N \c_zero_skip
4860     \par\addvspace { \g__enumxt_minipage_right_skip }
4861     \bool_if:NF \g__enumxt_minipage_center_vii_bool
4862     {
4863       \tl_put_left:Nn \g__enumxt_miniright_code_vii_tl
4864       {
4865         \centering
4866       }
4867     }
4868     \vbox_set_top:Nn \l__enumxt_miniright_code_vii_box
4869     {
4870       \tl_use:N \g__enumxt_miniright_code_vii_tl
4871     }
4872     \box_use_drop:N \l__enumxt_miniright_code_vii_box
4873     \skip_vertical:N \c_zero_skip
4874     \__enumxt_endminipage:
4875     \par\addvspace{ \g__enumxt_minipage_after_skip }
4876   }
4877   \bool_gset_false:N \g__enumxt_minipage_active_vii_bool
4878   \bool_gset_true:N \g__enumxt_minipage_center_vii_bool
4879   \tl_gclear:N \g__enumxt_miniright_code_vii_tl
4880   \dim_gzero:N \g__enumxt_minipage_right_vii_dim
4881   \bool_gset_false:N \g__enumxt_starred_bool
4882 }

```

`__enumxt_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumxt*` environment.

```

4883 \cs_new_protected:Nn \__enumxt_start_mini_viii:
4884 {
4885   \dim_compare:nNnT { \l__enumxt_minipage_right_viii_dim } > { \c_zero_dim }
4886   {
4887     \dim_set:Nn \l__enumxt_minipage_left_viii_dim
4888     {
4889       \linewidth
4890       - \l__enumxt_minipage_right_viii_dim
4891       - \l__enumxt_minipage_hsep_viii_dim
4892     }

```

```

4893     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4894     \dim_gset_eq:NN
4895         \g__enumext_minipage_right_viii_dim
4896         \l__enumext_minipage_right_viii_dim
4897     \__enumext_mini_addvspace_viii:
4898     \nointerlineskip\noindent
4899     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4900 }
4901 }
4902 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4903 {
4904     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4905     {
4906         \__enumext_stop_list:
4907         \IfDocumentMetadataT { \tag_resume:n {keyans*} }
4908         \end__enumext_mini_page
4909         \hfill
4910         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4911     }
4912     {
4913         \__enumext_stop_list:
4914     }
4915 }
4916 \__enumext_after_env:nn {keyans*}
4917 {
4918     \bool_if:NT \g__enumext_minipage_active_viii_bool
4919     {
4920         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4921         \par\addvspace { \g__enumext_minipage_right_skip }
4922         \bool_if:NF \g__enumext_minipage_center_viii_bool
4923         {
4924             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4925             {
4926                 \centering
4927             }
4928         }
4929         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4930         {
4931             \tl_use:N \g__enumext_miniright_code_viii_tl
4932         }
4933         \box_use_drop:N \l__enumext_miniright_code_viii_box
4934         \end__enumext_mini_page
4935         \par\addvspace{ \g__enumext_minipage_after_skip }
4936     }
4937     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4938     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4939     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4940     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4941 }

```

(End of definition for __enumext_start_mini_viii: and __enumext_stop_mini_viii:.)

13.47 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the **shortlst** package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```

4942 \NewDocumentEnvironment{enumext*}{o }
4943 {
4944     \__enumext_safe_exec_vii:
4945     \__enumext_parse_keys_vii:n {#1}
4946     \__enumext_before_list_vii:
4947     \__enumext_start_store_level_vii:
4948     \__enumext_start_list:nn { }
4949     {
4950         \__enumext_list_arg_two_vii:
4951         \__enumext_before_keys_exec_vii:
4952     }

```

```

4953     \setcounter { enumXvii } { \int_eval:n { \int_use:c { l__enumext_start_vii_int } - 1 } }
4954     \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
4955     \__enumext_starred_columns_set_vii:
4956     \item[] \scan_stop:
4957     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4958     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4959     \ignorespaces
4960   }
4961   {
4962     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4963     \__enumext_stop_item_tmp_vii:
4964     \__enumext_remove_extra_parsep_vii:
4965     \__enumext_after_list_vii:
4966   }

```

(End of definition for enumext*. This function is documented on page 5.)

`__enumext_safe_exec_vii:` We will first call the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are NOT nested within `enumext`, then call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

4967 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4968 {
4969   \__enumext_is_not_nested:
4970   \__enumext_internal_mini_page:
4971   \int_incr:N \l__enumext_level_h_int
4972   \int_compare:nNtT { \l__enumext_level_h_int } > { 1 }
4973   {
4974     \msg_error:nn { enumext } { nested }
4975   }
4976   \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
4977   {
4978     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4979   }
4980   \bool_set_true:N \l__enumext_starred_bool
4981   \bool_set_false:N \l__enumext_standar_bool
4982   \__enumext_is_on_first_level:
4983 }

```

(End of definition for __enumext_safe_exec_vii:.)

`__enumext_parse_keys_vii:n` First we will clear the variable `\l__enumext_series_name_str` used by the key `series`, process the environment `[⟨key = val⟩]` and execute the function `__enumext_parse_series:n` and used by the key `series`, then we execute the function `__enumext_store_active_keys_vii:n` and reprocess the `⟨keys⟩` to pass them to the storage `sequence` if the key `save-key` is not active.

```

4984 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4985 {
4986   \tl_if_novalue:nF {#1}
4987   {
4988     \str_clear:N \l__enumext_series_name_str
4989     \keys_set:nn { enumext / enumext* } {#1}
4990     \bool_if:NF \l__enumext_print_keyans_cmd_bool
4991     {
4992       \__enumext_parse_series:n {#1}
4993     }
4994     \__enumext_store_active_keys_vii:n {#1}
4995   }
4996 }

```

(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:` and `__enumext_start_mini_vii:` used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```

4997 \cs_new_protected:Nn \__enumext_before_list_vii:
4998 {
4999   \__enumext_vspace_above_vii:

```

```

5000     \__enumext_check_ans_active:
5001     \__enumext_before_args_exec_vii:
5002     \__enumext_start_mini_vii:
5003 }

```

(End of definition for __enumext_before_list_vii:.)

__enumext_after_list_vii: The function __enumext_after_list_vii: first calls the function __enumext_stop_mini_vii: which internally calls __enumext_stop_list: and __enumext_stop_store_level_vii: (§13.46.3) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions __enumext_after_stop_list_vii: used by the key `after`, __enumext_check_ans_key_hook: used by the key `check-ans`, __enumext_vspace_below_vii: used by the keys `below` and `below*`. Finally set \l__enumext_starred_bool to false and call the __enumext_resume_save_counter: function used by the `series`, `resume` and `resume*` keys.

```

5004 \cs_new_protected:Nn \__enumext_after_list_vii:
5005 {
5006     \__enumext_stop_mini_vii:
5007     \__enumext_after_stop_list_vii:
5008     \__enumext_check_ans_key_hook:
5009     \__enumext_vspace_below_vii:
5010     \bool_set_false:N \l__enumext_starred_bool
5011     \bool_if:NF \l__enumext_print_keyans_cmd_bool
5012     {
5013         \__enumext_starred_save_counter:
5014     }
5015 }

```

(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`.

```

5016 \cs_new_protected:Nn \__enumext_start_store_level_vii:
5017 {
5018     \bool_if:NT \l__enumext_store_active_bool
5019     {
5020         \int_compare:nNt { \l__enumext_level_int } > { 0 }
5021         {
5022             \__enumext_store_level_open_vii:
5023         }
5024     }
5025 }
5026 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
5027 {
5028     \bool_if:NT \l__enumext_store_active_bool
5029     {
5030         \int_compare:nNt { \l__enumext_level_int } > { 0 }
5031         {
5032             \__enumext_store_level_close_vii:
5033         }
5034     }
5035 }

```

(End of definition for __enumext_start_store_level_vii: and __enumext_stop_store_level_vii:.)

13.47.1 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.

```

5036 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
5037 {
5038     \skip_horizontal:n
5039     {
5040         -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
5041     }
5042     \ignorespaces
5043 }

```

(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_standar_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 we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```

5044 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
5045 {
5046   \__enumext_stop_item_tmp_vii:
5047   \int_incr:N \l__enumext_item_column_pos_vii_int
5048   \int_gincr:N \g__enumext_item_count_all_vii_int
5049   \__enumext_item_peek_args_vii:
5050 }

```

The function `__enumext_item_peek_args_vii:` will handle the `\item(<number>)`. Look for the argument “(”, if it is present we will call the function `__enumext_joined_item_vii:w (<number>)`, which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```

5051 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
5052 {
5053   \peek_meaning:NTF (
5054     { \__enumext_joined_item_vii:w }
5055     { \__enumext_joined_item_vii:w (1) }
5056   }

```

The function `__enumext_joined_item_vii:w` will first call the function `__enumext_starred_joined_item_vii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “*”, if it is present we will call the function `__enumext_starred_item_vii:w` otherwise we will call the function `__enumext_standar_item_vii:w`.

```

5057 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
5058 {
5059   \__enumext_starred_joined_item_vii:n {#1}
5060   \peek_meaning_remove:NTF *
5061     { \__enumext_starred_item_vii:w }
5062     { \__enumext_standar_item_vii:w }
5063 }

```

The function `__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_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 we will set 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]`.

```

5064 \cs_new_protected:Npn \__enumext_standar_item_vii:w
5065 {
5066   \bool_set_false:N \l__enumext_item_starred_vii_bool
5067   \peek_meaning:NTF [
5068     {
5069       \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
5070       \__enumext_start_item_vii:w
5071     }
5072     {
5073       \bool_set_true:N \l__enumext_wrap_label_vii_bool
5074       \legacy_if_set_true:n { @noitemarg }
5075       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5076     }
5077   }

```

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>]`.

```

5078 \cs_new_protected:Npn \__enumext_starred_item_vii:w
5079 {
5080   \bool_set_true:N \l__enumext_item_starred_vii_bool
5081   \bool_set_true:N \l__enumext_wrap_label_vii_bool
5082   \peek_meaning:NTF [
5083     { \__enumext_starred_item_vii_aux_i:w }
5084     { \__enumext_starred_item_vii_aux_ii:w }
5085   }
5086 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
5087 {
5088   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
5089   \__enumext_starred_item_vii_aux_ii:w

```

```

5090 }
5091 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
5092 {
5093   \peek_meaning:NTF [
5094     { \__enumext_starred_item_vii_aux_iii:w }
5095     {
5096       \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
5097       \legacy_if_set_true:n { @noitemarg }
5098       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5099     }
5100   }
5101   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
5102   {
5103     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
5104     \legacy_if_set_true:n { @noitemarg }
5105     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5106   }

```

(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 our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[\langle symbol \rangle][\langle offset \rangle]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute 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 pdf_latex and not if you are running lua_latex

```

5107 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
5108 {
5109   \legacy_if:nT { @noitemarg }
5110   {
5111     \legacy_if_set_false:n { @noitemarg }
5112     \legacy_if:nT { @nmbrrlist }
5113     {
5114       \IfDocumentMetadataT
5115       {
5116         \bool_if:NT \l__enumext_hyperref_bool
5117         {
5118           \legacy_if_set_true:n { @hyper@item }
5119         }
5120       }
5121       \refstepcounter{enumXvii}
5122       \bool_if:NT \l__enumext_check_answers_bool
5123       {
5124         \int_gincr:N \g__enumext_item_number_int
5125         \bool_set_true:N \l__enumext_item_number_bool
5126       }
5127     }
5128   }
5129   \bool_if:NT \l__enumext_item_starred_vii_bool
5130   {
5131     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
5132     {
5133       \tl_gset_eq:NN
5134       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
5135     }
5136     \mode_leave_vertical:
5137     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
5138     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5139     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
5140     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5141   }
5142   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
5143   {
5144     \tl_use:N \l__enumext_label_font_style_vii_tl
5145     \bool_if:NTF \l__enumext_wrap_label_vii_bool

```

```

5146         {
5147             \__enumext_wrapper_label_vii:n {#1}
5148         }
5149         { #1 }
5150     }
5151     \skip_horizontal:N \__enumext_labelsep_vii_dim \ignorespaces
5152 }

```

(End of definition for __enumext_fake_make_label_vii:n.)

13.47.2 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 we will NOT use an extra group and the plain form of the lrbox environment.

__enumext_start_item_vii:w The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_stop_item_vii: which we will define later, after that we will start capturing \item and “item content” in a horizontal box where the width will be \itemwidth plus \labelwidth plus \labelsep.

```

5153 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5154 {
5155     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5156     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
5157     {
5158         \l__enumext_joined_width_vii_dim
5159         + \l__enumext_labelwidth_vii_dim
5160         + \l__enumext_labelsep_vii_dim
5161     }

```

Redefine the \footnote command.

```
5162 \__enumext_renew_footnote_starred:
```

Now we insert our sockets for tagging PDF support and run \item.

```

5163 \__enumext_start_list_tag:n {enumext*}
5164 \__enumext_fake_make_label_vii:n {#1}
5165 \__enumext_stop_start_list_tag:

```

Finally we open the minipage environment, capture the “item content”, make \parindent take the value of the key listparindent and \parskip take the value of the key parsep, then execute the keys itemindent and first.

Here the use of \unskip 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.

```

5166 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
5167 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5168 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5169 \__enumext_unskip_unkern:
5170 \__enumext_unskip_unkern:
5171 \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5172 \tl_use:N \l__enumext_fake_item_indent_vii_tl
5173 \tl_use:N \l__enumext_after_list_args_vii_tl
5174 }

```

The __enumext_stop_item_vii: function will finish the fetching \item and “item content” by closing the minipage environment, the sockets for tagging PDF and the horizontal box.

```

5175 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5176 {
5177     \__enumext_endminipage:
5178     \__enumext_stop_list_tag:n {enumext*}
5179     \hbox_set_end:

```

Here we will reduce the warnings a bit by setting the value of \hbadness to 10000, print \item and “item content” from the horizontal box.

```

5180 \int_set:Nn \hbadness { 10000 }
5181 \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.

```

5182 \int_compare:nNnTF
5183 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
5184 {
5185     \par\noindent
5186     \int_zero:N \l__enumext_item_column_pos_vii_int
5187 }

```



```

5188     {
5189         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
5190     }
5191 }

```

(End of definition for \l__enumext_start_item_vii:w and \l__enumext_stop_item_vii:.)

\l__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 use of `\unskip` or `\removeatlastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in *vertical mode*.

```

5192 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
5193 {
5194     \int_compare:nNnT
5195     {
5196         \int_mod:nn
5197         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
5198     }
5199     =
5200     { 0 }
5201     {
5202         \para_end:
5203         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
5204         \skip_vertical:N \c_zero_skip
5205         \int_gzero:N \g__enumext_item_count_all_vii_int
5206     }
5207 }

```

(End of definition for \l__enumext_remove_extra_parsep_vii:.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “hook” function `\l__enumext_after_env:nn`.

```

5208 \l__enumext_after_env:nn {enumext*}
5209 {
5210     \l__enumext_execute_after_env:
5211 }

```

13.48 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `\l__enumext_check_starred_cmd:n` function added in the second part.

```

5212 \NewDocumentEnvironment{keyans*}{ o }
5213 {
5214     \l__enumext_safe_exec_viii:
5215     \l__enumext_parse_keys_viii:n {#1}
5216     \l__enumext_before_list_viii:
5217     \l__enumext_start_list:nn { }
5218     {
5219         \l__enumext_list_arg_two_viii:
5220         \l__enumext_before_keys_exec_viii:
5221     }
5222     \setcounter { enumXviii } { \int_eval:n { \int_use:c { \l__enumext_start_viii_int } - 1 } }
5223     \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
5224     \l__enumext_starred_columns_set_viii:
5225     \item[] \scan_stop:
5226     \cs_set_eq:NN \l__enumext_stop_item_tmp_viii: \l__enumext_first_item_tmp_viii:
5227     \cs_set_eq:NN \item \l__enumext_start_item_tmp_viii:
5228     \ignorespaces
5229 }
5230 {
5231     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
5232     \l__enumext_stop_item_tmp_viii:
5233     \l__enumext_remove_extra_parsep_viii:
5234     \l__enumext_check_starred_cmd:n { item }
5235     \l__enumext_after_list_viii:
5236 }

```

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

`__enumext_safe_exec_viii:` The `__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `\l__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```

5237 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5238 {
5239   \bool_if:NF \l__enumext_store_active_bool
5240   {
5241     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
5242   }
5243   \int_incr:N \l__enumext_keyans_level_h_int
5244   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5245   {
5246     \msg_error:nn { enumext } { nested }
5247   }
5248   \__enumext_keyans_name_and_start:
5249   \bool_if:NT \l__enumext_starred_bool
5250   {
5251     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5252   }
5253   \bool_set_true:N \l__enumext_starred_bool
5254   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5255   \bool_set_false:N \l__enumext_store_active_bool
5256   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5257   {
5258     \msg_error:nn { enumext } { keyans-wrong-level }
5259   }
5260 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

`__enumext_parse_keys_viii:n` Parse [`<key = val>`] for `keyans*`.

```

5261 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5262 {
5263   \tl_if_novalue:NF {#1}
5264   {
5265     \keys_set:nn { enumext / keyans* } {#1}
5266   }
5267 }

```

(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, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

5268 \cs_new_protected:Nn \__enumext_before_list_viii:
5269 {
5270   \__enumext_vspace_above_viii:
5271   \__enumext_before_args_exec_viii:
5272   \__enumext_start_mini_viii:
5273 }

```

(End of definition for `__enumext_before_list_viii:`.)

`__enumext_after_list_viii:` The function `__enumext_after_list_viii:` first call 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.

```

5274 \cs_new_protected:Nn \__enumext_after_list_viii:
5275 {
5276   \__enumext_stop_mini_viii:
5277   \__enumext_after_stop_list_viii:
5278   \__enumext_vspace_below_viii:
5279 }

```

(End of definition for `__enumext_after_list_viii:`.)

13.48.1 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* (`\langle number \rangle`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\langle label \rangle` next to the `\langle content \rangle` if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*\langle content \rangle`, `\item(\langle number \rangle)*` and `\item(\langle number \rangle)*\langle content \rangle` 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.

```
5280 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5281 {
5282   \skip_horizontal:n
5283   {
5284     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
5285   }
5286   \ignorespaces
5287 }
```

(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_standar_item_viii:w`

First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
5288 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5289 {
5290   \__enumext_stop_item_tmp_viii:
5291   \int_incr:N \l__enumext_item_column_pos_viii_int
5292   \int_gincr:N \g__enumext_item_count_all_viii_int
5293   \__enumext_item_peek_args_viii:
5294 }
```

The function `__enumext_item_peek_args_viii:` will handle the `\item(\langle number \rangle)`. Look for the argument “`(`”, if it is present we will call the function `__enumext_joined_item_viii:w(\langle number \rangle)`, which is in charge of joining the item’s in the same row, in case they are not present we will set the default value `(1)`.

```
5295 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5296 {
5297   \peek_meaning:NTF (
5298     { \__enumext_joined_item_viii:w }
5299     { \__enumext_joined_item_viii:w (1) }
5300 }
```

The function `__enumext_joined_item_viii:w` will first call the function `__enumext_starred_joined_item_viii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “`*`”, if it is present we will call the function `__enumext_starred_item_viii:w` otherwise we will call the function `__enumext_standar_item_viii:w`.

```
5301 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5302 {
5303   \__enumext_starred_joined_item_viii:n {#1}
5304   \peek_meaning_remove:NTF *
5305     { \__enumext_starred_item_viii:w }
5306     { \__enumext_standar_item_viii:w }
5307 }
```

The function `__enumext_standar_item_viii:w` will first look for the argument “`[`”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_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[\langle custom \rangle]` by means of the function `__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@notitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_viii:w[\l__enumext_label_viii_tl]`.

```
5308 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5309 {
5310   \bool_set_false:N \l__enumext_item_starred_viii_bool
5311   \bool_set_false:N \l__enumext_item_wrap_key_bool
5312   \peek_meaning:NTF [
5313     {
5314       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
```

```

5315     \__enumext_start_item_viii:w
5316   }
5317   {
5318     \bool_set_true:N \__enumext_wrap_label_viii_bool
5319     \legacy_if_set_true:n { @noitemarg }
5320     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5321   }
5322 }

```

(End of definition for __enumext_start_item_tmp_viii: and others.)

The function __enumext_starred_item_viii:w together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[\langle content \rangle]`.

```

5323 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5324 {
5325   \bool_set_true:N \__enumext_item_starred_viii_bool
5326   \bool_set_true:N \__enumext_item_wrap_key_bool
5327   \bool_set_true:N \__enumext_wrap_label_viii_bool
5328   \peek_meaning:NTF [
5329     { \__enumext_starred_item_viii_aux_i:w }
5330     { \__enumext_starred_item_viii_aux_ii:w }
5331   }

```

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`.

```

5332 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5333 {
5334   \tl_clear:N \l__enumext_store_current_label_tl
5335   \tl_if_no_value:nF { #1 }
5336   {
5337     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5338     {
5339       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
5340       \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5341     }
5342     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5343   }
5344   \__enumext_starred_item_viii_aux_ii:w
5345 }
5346 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5347 {
5348   \legacy_if_set_true:n { @noitemarg }
5349   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5350 }

```

The function __enumext_keyans_starred_item_star: will be in charge of storing the current *label* for `\item*` followed by the `[\langle content \rangle]` for `\item*[\langle content \rangle]` 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.

```

5351 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5352 {
5353   \tl_put_left:Ne \l__enumext_store_current_label_tl { \__enumext_label_viii_tl }
5354   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5355   \__enumext_keyans_store_ref:
5356   \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
5357   \__enumext_keyans_addto_seq_link:
5358   \int_gincr:N \g__enumext_check_starred_cmd_int
5359   \dim_compare:nNt { \__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5360   {
5361     \dim_set:Nn \__enumext_mark_sym_sep_viii_dim { \__enumext_labelsep_viii_dim }
5362   }
5363   \bool_if:NT \l__enumext_show_answer_bool
5364   {
5365     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5366     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5367     \__enumext_print_keyans_box:NN
5368     \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5369   }
5370   \bool_if:NT \l__enumext_show_position_bool

```

```

5371 {
5372   \tl_set:Nc \l__enumext_mark_answer_sym_tl
5373   {
5374     \group_begin:
5375     \exp_not:N \normalfont
5376     \exp_not:N \footnotesize [ \int_eval:n
5377     {
5378       \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5379     }
5380   ]
5381   \group_end:
5382 }
5383 \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5384 \__enumext_print_keyans_box:NN
5385   \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5386 }
5387 }

```

(End of definition for __enumext_starred_item_viii:w and others.)

The implementation at this is very similar to that of the `enumext*` environment.

```

\__enumext_keyans_wrapper_label_viii:n
  \__enumext_fake_make_label_viii:n

```

```

5388 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5389 {
5390   \bool_lazy_all:nT
5391   {
5392     { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5393     { \bool_if_p:N \l__enumext_show_answer_bool }
5394     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5395     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5396   }
5397   {
5398     \cs_set_eq:NN
5399     \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5400   }
5401   \bool_if:NTF \l__enumext_wrap_label_viii_bool
5402   {
5403     \__enumext_wrapper_label_viii:n {#1}
5404   }
5405   { #1 }
5406 }
5407 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5408 {
5409   \legacy_if:nT { @noitemarg }
5410   {
5411     \legacy_if_set_false:n { @noitemarg }
5412     \legacy_if:nT { @nmbrrlist }
5413     {
5414       \refstepcounter{enumXviii}
5415     }
5416   }
5417   \bool_if:NT \l__enumext_item_starred_viii_bool
5418   {
5419     \__enumext_keyans_starred_item_star:
5420   }
5421   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5422   {
5423     \tl_use:N \l__enumext_label_font_style_viii_tl
5424     \__enumext_keyans_wrapper_label_viii:n {#1}
5425   }
5426   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5427 }

```

(End of definition for __enumext_keyans_wrapper_label_viii:n and __enumext_fake_make_label_viii:n.)

13.48.2 Real definition of \item in keyans*

The implementation at this is very similar to that of the `enumext*` environment.

```

\__enumext_start_item_viii:w
  \__enumext_stop_item_viii:

```

```

5428 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5429 {
5430   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5431   \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5432   {

```

```

5433     \l__enumext_joined_width_viii_dim
5434     + \l__enumext_labelwidth_viii_dim
5435     + \l__enumext_labelsep_viii_dim
5436   }
5437   \__enumext_renew_footnote_starred:
5438   \__enumext_start_list_tag:n {keyans*}
5439   \__enumext_fake_make_label_viii:n {#1}
5440   \__enumext_stop_start_list_tag:
5441   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5442   \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5443   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5444   \__enumext_unskip_unkern:
5445   \__enumext_unskip_unkern:
5446   \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5447   \tl_use:N \l__enumext_fake_item_indent_viii_tl
5448   \bool_if:NT \l__enumext_item_starred_viii_bool
5449   {
5450     \__enumext_keyans_show_item_opt_viii:
5451   }
5452   \tl_use:N \l__enumext_after_list_args_viii_tl
5453 }
5454 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5455 {
5456   \__enumext_endminipage:
5457   \__enumext_stop_list_tag:n {keyans*}
5458   \hbox_set_end:
5459   \int_set:Nn \hbadness { 10000 }
5460   \box_use_drop:N \l__enumext_item_text_viii_box
5461   \int_compare:nNnTF
5462   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5463   {
5464     \par\noindent
5465     \int_zero:N \l__enumext_item_column_pos_viii_int
5466   }
5467   {
5468     \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5469   }
5470 }

```

(End of definition for __enumext_start_item_viii:w and __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii:

The implementation at this is very similar to that of the `enumext*` environment.

```

5471 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5472 {
5473   \int_compare:nNnT
5474   {
5475     \int_mod:nn
5476     { \g__enumext_item_count_all_viii_int }
5477     { \l__enumext_columns_viii_int }
5478   }
5479   =
5480   { 0 }
5481   {
5482     \para_end:
5483     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5484     \skip_vertical:N \c_zero_skip
5485     \int_gzero:N \g__enumext_item_count_all_viii_int
5486   }
5487 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

13.49 The command \getkeyans

\getkeyans
 __enumext_getkeyans_aux:n
 __enumext_getkeyans:n

The `\getkeyans` command takes a *mandatory argument* of the form $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-ans` key.

```

5488 \NewDocumentCommand \getkeyans { m }
5489 {
5490   \exp_args:Ne \__enumext_getkeyans_aux:n
5491   { \tl_to_str:e { \text_expand:n {#1} } }
5492 }

```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```

5493 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5494 {
5495   \str_if_in:nnTF {#1} { : }
5496   {
5497     \use:e
5498     {
5499       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5500       { {##1} {##2} }
5501     }
5502     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5503   }
5504   { \msg_error:nnn { enumext } { missing-colon } {#1} }
5505 }

```

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*.

```

5506 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5507 {
5508   \prop_if_exist:cTF { g__enumext_#1_prop }
5509   {
5510     \prop_item:cn { g__enumext_#1_prop }{#2}
5511   }
5512   {
5513     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5514   }
5515 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 19.)

13.50 The command `\printkeyans`

The `\printkeyans` command prints “*all stored content*” in the *sequence* defined by the *save-ans* key.

The first thing we will do is define a set of *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

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⟩]`.

```

5516 \keys_define:nn { enumext / print }
5517 {
5518   print* .code:n = \keys_precompile:neN { enumext / enumext* }
5519               { \__enumext_filter_save_key:n {#1} }
5520               \l__enumext_print_keyans_starred_tl, % starred cmd
5521   print* .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5522                       rightmargin=opt, listparindent=opt, nosep, label=\arabic*,
5523                       columns=2, first=\small, font=\small },
5524   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5525                  { \__enumext_filter_save_key:n {#1} }
5526                  \l__enumext_print_keyans_i_tl,
5527   print-1 .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5528                       rightmargin=opt, listparindent=opt, nosep, label=\arabic*,
5529                       columns=2, first=\small, font=\small },
5530   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5531                  { \__enumext_filter_save_key:n {#1} }
5532                  \l__enumext_print_keyans_ii_tl,
5533   print-2 .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5534                       rightmargin=opt, listparindent=opt, nosep, label=(\alph*),
5535                       first=\small, font=\small },
5536   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5537                  { \__enumext_filter_save_key:n {#1} }
5538                  \l__enumext_print_keyans_iii_tl,
5539   print-3 .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5540                       rightmargin=opt, listparindent=opt, nosep, label=\roman*,
5541                       first=\small, font=\small },
5542   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }

```



```

5543         { \__enumext_filter_save_key:n {#1} }
5544         \__enumext_print_keyans_iv_tl,
5545     print-4 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5546                           rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5547                           first=\small, font=\small },
5548     print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5549                           { \__enumext_filter_save_key:n {#1} }
5550                           \__enumext_print_keyans_vii_tl, % starred nested
5551     print-* .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5552                           rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5553                           first=\small, font=\small },
5554 }

```

• The reason for storing $\langle keys \rangle$ 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.

`\printkeyans`

`__enumext_printkeyans:nnn`

Create a user command to print “all stored content” in *sequence* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

5555 \NewDocumentCommand \printkeyans { s O{ } m }
5556 {
5557     \group_begin:
5558         \bool_set_true:N \__enumext_print_keyans_cmd_bool
5559         \tl_use:N \__enumext_print_keyans_i_tl
5560         \tl_use:N \__enumext_print_keyans_ii_tl
5561         \tl_use:N \__enumext_print_keyans_iii_tl
5562         \tl_use:N \__enumext_print_keyans_iv_tl
5563         \tl_use:N \__enumext_print_keyans_vii_tl
5564         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5565         \bool_set_false:N \__enumext_print_keyans_cmd_bool
5566     \group_end:
5567 }

```

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.

```

5568 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5569 {
5570     \seq_if_exist:cTF { g__enumext_#3_seq }
5571     {
5572         \seq_if_empty:cF { g__enumext_#3_seq }
5573         {

```

If the *starred argument* ‘*’ is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `__enumext_print_keyans_starred_tl` that contains the default $\langle keys \rangle$ for the environment `enumext*`, we set `__enumext_base_line_fix_bool` and `__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `__enumext_base_line_fix_bool` and `__enumext_print_keyans_star_bool` to false.

```

5574         \bool_if:nTF {#1}
5575         {
5576             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5577             {
5578                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5579             }
5580             {
5581                 \tl_use:N \__enumext_print_keyans_starred_tl
5582                 \bool_set_true:N \__enumext_base_line_fix_bool
5583                 \bool_set_true:N \__enumext_print_keyans_star_bool
5584                 \begin{enumext*}[#2]
5585                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5586                     \end{enumext*}
5587                 \bool_set_false:N \__enumext_base_line_fix_bool
5588                 \bool_set_false:N \__enumext_print_keyans_star_bool
5589             }
5590         }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “first level” then map the *sequence*.

```

5591         {
5592             \begin{enumext}[#2]
5593             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }

```

```

5594             \end{enumext}
5595         }
5596     }
5597 }
5598 {
5599     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5600 }
5601 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 20.)

13.51 The command `\setenumext`

The command `\setenumext` will be in charge of managing the *⟨keys⟩* passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` and `enumext` environments so as not to capture *⟨keys⟩* that complicate us.

```

\__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.

```

5602 \cs_new:Npn \__enumext_filter_level:n #1
5603 {
5604     \use:e
5605     {
5606         \keyval_parse:NNn
5607         \__enumext_filter_level_key:n
5608         \__enumext_filter_level_pair:nn {#1}
5609     }
5610 }

```

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.

```

5611 \cs_new:Npn \__enumext_filter_level_key:n #1
5612 {
5613     \str_case:nnF {#1}
5614     {
5615         { resume* } {} { reset } {} { reset* } {}
5616     }
5617     { , { \exp_not:n {#1} } }
5618 }

```

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.

```

5619 \cs_new:Npn \__enumext_filter_level_pair:nn #1#2
5620 {
5621     \str_case:nnF {#1}
5622     {
5623         { series } {} { save-ans } {} { resume } {}
5624     }
5625     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5626 }

```

(End of definition for `__enumext_filter_level:n`, `__enumext_filter_level_key:n`, and `__enumext_filter_level_pair:nn`.)

Now define a “meta families” of *⟨keys⟩* to access from `\setenumext`.

```

5627 \keys_define:nn { enumext / meta-families }
5628 {
5629     enumext-1 .code:n = {
5630         \keys_set:ne { enumext / level-1 }
5631         {
5632             \__enumext_filter_level:n {#1}
5633         }
5634     },
5635     enumext-2 .code:n = {
5636         \keys_set:ne { enumext / level-2 }
5637         {
5638             \__enumext_filter_level:n {#1}
5639         }
5640     },
5641     enumext-3 .code:n = {
5642         \keys_set:ne { enumext / level-3 }

```

```

5643         {
5644             \__enumext_filter_level:n {#1}
5645         }
5646     },
5647     enumext-4 .code:n = {
5648         \keys_set:ne { enumext / level-4 }
5649         {
5650             \__enumext_filter_level:n {#1}
5651         }
5652     },
5653     enumext* .code:n = {
5654         \keys_set:ne { enumext / enumext* }
5655         {
5656             \__enumext_filter_level:n {#1}
5657         }
5658     },
5659     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
5660     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
5661     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } },
5662     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } },
5663     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } },
5664     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } },
5665     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } },
5666     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } },
5667     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
5668 }

```

We store them in the constant sequence `__enumext_all_families_seq` separated by commas.

```

5669 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5670 {
5671     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5672     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5673 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

\__enumext_set_parse:n 5674 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
\__enumext_set_error:nn 5675 {
5676     \seq_clear:N \l__enumext_setkey_tmpa_seq
5677     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5678     \int_set:Nn \l__enumext_setkey_tmpa_int
5679     {
5680         \seq_count:N \l__enumext_setkey_tmpb_seq
5681     }
5682     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5683     {
5684         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5685         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5686         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5687         {
5688             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5689         }
5690     }
5691     {
5692         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5693     }
5694     \seq_if_empty:NNTF \l__enumext_setkey_tmpa_seq
5695     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5696     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5697     {
5698         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5699     }
5700 }

```

Internal functions used by the `\setenumext` command.

```

5701 \cs_new_protected:Npn \__enumext_set_parse:n #1
5702 {
5703     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5704     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5705     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5706     \tl_if_empty:NNTF \l__enumext_setkey_tmpb_tl
5707     {
5708         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq

```

```

5709         { \tl_trim_spaces:n {#1} }
5710     }
5711     { \__enumext_set_error:nn {#1} { } }
5712 }
5713 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5714 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\setenumext`, `__enumext_set_parse:n`, and `__enumext_set_error:nn`. This function is documented on page 6.)

13.52 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 [Add .meta key to existing keys \(l3keys\)](#).

`\setenumextmeta`

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the *optional argument*.

`\c__enumext_meta_paths_prop`
`__enumext_add_meta_key:nnn`
`__enumext_def_meta_key:nnn`
`__enumext_def_meta_key:Vnn`

```

5715 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5716 {
5717     {enumext,1} = level-1,
5718     {enumext,2} = level-2,
5719     {enumext,3} = level-3,
5720     {enumext,4} = level-4,
5721     {enumext*} = enumext*
5722 }

```

Now we create the user command taking care that unknown cannot be passed as an argument.

```

5723 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5724 {
5725     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5726     { \msg_error:nn { enumext } { prohibited-unknown } }
5727     {
5728         \bool_if:nTF {#1}
5729         {
5730             \int_step_inline:nn { 4 }
5731             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5732             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5733         }
5734         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5735     }
5736 }

```

The internal functions `__enumext_add_meta_key:nnn` and `__enumext_def_meta_key:nnn` will check the *optional argument* and create the “meta-key”.

```

5737 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5738 {
5739     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5740     \tl_replace_all:Nnn \l__enumext_meta_path_tl {~} {}
5741     \prop_get:NVNTF
5742     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5743     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5744     {
5745         \msg_error:nnn { enumext } { unknown-set } {#1}
5746         \use_none:nn
5747     }
5748 }
5749 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5750 {
5751     \bool_lazy_or:nnTF
5752     { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5753     { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5754     { \msg_error:nnn { enumext } { already-defined } {#2} }
5755     {
5756         \keys_define:nn { enumext / #1 }
5757         {
5758             #2 .meta:n = {#3},
5759             #2 .value_forbidden:n = true
5760         }
5761     }
5762 }
5763 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

13.53 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](#).

`\foreachkeyans`

`__enumext_parse_foreach_keys:nn`

`__enumext_parse_foreach_keys:n`

`__enumext_foreach_keyans:nn`

`__enumext_foreach_add_body:n`

We define a set of `<keys>` for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```

5764 \keys_define:nn { enumext / foreach }
5765 {
5766   before .tl_set:N = \l__enumext_foreach_before_tl,
5767   before .value_required:n = true,
5768   after .tl_set:N = \l__enumext_foreach_after_tl,
5769   after .value_required:n = true,
5770   start .int_set:N = \l__enumext_foreach_start_int,
5771   start .value_required:n = true,
5772   stop .int_set:N = \l__enumext_foreach_stop_int,
5773   stop .value_required:n = true,
5774   step .int_set:N = \l__enumext_foreach_step_int,
5775   step .value_required:n = true,
5776   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5777   wrapper .value_required:n = true,
5778   sep .tl_set:N = \l__enumext_foreach_sep_tl,
5779   sep .value_required:n = true,
5780   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5781 }
5782 \keys_precompile:nnN { enumext / foreach }
5783 {
5784   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5785 }
5786 \l__enumext_foreach_default_keys_tl

```

Functions for handling unknown `<keys>`.

```

5787 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5788 {
5789   \tl_if_blank:nTF {#2}
5790   {
5791     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5792   }
5793   {
5794     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5795   }
5796 }
5797 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5798 {
5799   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5800 }

```

We create the command.

```

5801 \NewDocumentCommand \foreachkeyans { +0{ } m }
5802 {
5803   \__enumext_foreach_keyans:nn {#1} {#2}
5804 }

```

Finally the internal functions `__enumext_foreach_keyans:nn` and `__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5805 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5806 {
5807   \tl_use:N \l__enumext_foreach_default_keys_tl
5808   \keys_set:nn { enumext / foreach } {#1}
5809   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5810   \prop_if_exist:cF { g__enumext_#2_prop }
5811   {
5812     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5813   }
5814   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5815   {
5816     \int_set:Nn \l__enumext_foreach_stop_int
5817     { \prop_count:c { g__enumext_#2_prop } }
5818   }
5819   \seq_clear:N \l__enumext_foreach_print_seq
5820   \int_step_function:nnnN

```

```

5821     { \l__enumext_foreach_start_int }
5822     { \l__enumext_foreach_step_int }
5823     { \l__enumext_foreach_stop_int }
5824     \__enumext_foreach_add_body:n
5825     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5826   }
5827   \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5828   {
5829     \seq_put_right:Ne \l__enumext_foreach_print_seq
5830     {
5831       \exp_not:V \l__enumext_foreach_before_tl
5832       \__enumext_foreach_wrapper:n
5833       {
5834         \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5835       }
5836       \exp_not:V \l__enumext_foreach_after_tl
5837     }
5838   }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 19.)

13.54 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5839 \msg_new:nnn { enumext } { package-load }
5840 {
5841   The~'#1'~package~is~already~loaded.
5842 }
5843 \msg_new:nnn { enumext } { package-not-load }
5844 {
5845   The~'#1'~package~will~be~loaded~as~a~dependency.
5846 }
5847 \msg_new:nnn { enumext } { package-load-foot }
5848 {
5849   The~'#1'~package~is~loaded~with~the~option~'#2'.
5850 }

```

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

```

5851 \msg_new:nnn { enumext } { counters }
5852 {
5853   The~counter~'#1'~is~already~defined~by~some~\\
5854   package~or~macro,~it~cannot~be~continued.
5855 }

```

Message used by `align` and `mark-pos` keys.

```

5856 \msg_new:nnn { enumext } { unknown-choice }
5857 {
5858   The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5859 }

```

Message used by reserved `anskey*` environment by `enumext` package.

```

5860 \msg_new:nnnn { enumext } { anskey-env-error }
5861 {
5862   The~environment~'#1'~is~reserved~by ~\\
5863   'enumext'~package,~it~is~already~defined.
5864 }
5865 {
5866   The~environment~'#1'~is~defined~internally ~
5867   for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
5868 }
5869 \msg_new:nnn { enumext } { anskey-env-nested }
5870 {
5871   The~#1~'#2'~can't~be~nested~\msg_line_context:.
5872 }

```

Message used in the creation of *prop list* by `enumext` package.

```

5873 \msg_new:nnn { enumext } { store-prop }
5874 {
5875   *~Package~enumext:~Creating ~
5876   \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5877 }
5878 \msg_new:nnn { enumext } { store-seq }
5879 {

```

```

5880     *~Package~enumext:~Creating ~
5881     \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5882 }
5883 \msg_new:nnn { enumext } { store-int }
5884 {
5885     *~Package~enumext:~Creating ~
5886     \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5887 }
5888 \msg_new:nnn { enumext } { prop-seq-int-hook }
5889 {
5890     *~Package~enumext:~Elements~in ~
5891     \c_backslash_str g__enumext_#1_prop~=#2.\\
5892     *~Package~enumext:~Elements~in ~
5893     \c_backslash_str g__enumext_#1_seq~=#3.\\
5894     *~Package~enumext:~Value~off ~
5895     \c_backslash_str g__enumext_resume_#1_int~=#4.
5896 }
5897 \msg_new:nnn { enumext } { item-answer-hook }
5898 {
5899     *~Package~enumext:~Value~off ~
5900     \c_backslash_str g__enumext_item_number_int~=#1.\\
5901     *~Package~enumext:~Value~off ~
5902     \c_backslash_str g__enumext_item_anskey_int~=#2.\\
5903     *~Package~enumext:~Difference~item_number_int~~item_anskey_int~=#3.
5904 }

```

Message used by [*key = val*] system and `\setenumext` command.

```

5905 \msg_new:nnn { enumext } { invalid-key }
5906 {
5907     The~key~'#1'~is~not~know~the~level~#2.
5908 }
5909 \msg_new:nnn { enumext } { unknown-key-family }
5910 {
5911     Unknown~key~family~`\_keys_key_str'~for~enumext.
5912 }

```

Messages used in length calculation.

```

5913 \msg_new:nnn { enumext } { width-negative }
5914 {
5915     Ignoring~negative~value~'#1=#2'~\msg_line_context:.\\
5916     The~key~'#1'~ accepts~values ~>=~0pt.
5917 }
5918 \msg_new:nnn { enumext } { width-zero }
5919 {
5920     Invalid~'#1=#2'~\msg_line_context:.\\
5921     The~key~'#1'~ accepts~values ~>~0pt.
5922 }

```

Messages used by `show-length` key in `enumext`.

```

5923 \msg_new:nnn { enumext } { list-lengths }
5924 {
5925     ****~Lengths~used~by~'enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\
5926     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5927     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5928     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5929     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5930     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5931     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5932     \__enumext_show_length:nnn { skip } { topsep } {#1}
5933     \__enumext_show_length:nnn { skip } { parsep } {#1}
5934     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5935     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5936     ****~
5937 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

5938 \msg_new:nnn { enumext } { list-lengths-not-nested }
5939 {
5940     ****~Lengths~used~by~'#2'~environment~\msg_line_context:~\c_space_tl ****\\
5941     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5942     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5943     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5944     \__enumext_show_length:nnn { dim } { leftmargin } {#1}

```



```

5945     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5946     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5947     \__enumext_show_length:nnn { skip } { topsep } {#1}
5948     \__enumext_show_length:nnn { skip } { parsep } {#1}
5949     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5950     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5951     *****
5952 }

```

Messages used by `ref` key.

```

5953 \msg_new:nnn { enumext } { key-ref-empty }
5954 {
5955     Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5956 }

```

Messages used by `save-ans` key.

```

5957 \msg_new:nnn { enumext } { save-ans-empty }
5958 {
5959     Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5960 }
5961 \msg_new:nnn { enumext } { save-ans-log }
5962 {
5963     *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5964 }
5965 \msg_new:nnn { enumext } { save-ans-log-hook }
5966 {
5967     *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5968 }
5969 \msg_new:nnn { enumext } { save-ans-hook }
5970 {
5971     Stop~storing~for~'save-ans=#1'~\msg_line_context:.
5972 }

```

Messages used by the internal system to check answer used by `check-ans` key.

```

5973 \msg_new:nnn { enumext } { need-save-ans }
5974 {
5975     Key~'#1'~ works~only~with~the~'save-ans'~key~in~'#2'~ \msg_line_context:.
5976 }
5977 \msg_new:nnn { enumext } { items-same-answer }
5978 {
5979     *****\
5980     *~Package~enumext:~Checking~answers~in~'#1' ~
5981     for~\c_left_brace_str #2 \c_right_brace_str\
5982     *~started~#3~and~close~\msg_line_context: : ~
5983     'OK',~all~items~with~answer.\
5984     *****
5985 }
5986 \msg_new:nnn { enumext } { item-greater-answer }
5987 {
5988     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\
5989     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\
5990     Items~>~Answers.
5991 }
5992 \msg_new:nnn { enumext } { item-less-answer }
5993 {
5994     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\
5995     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\
5996     Items~<~Answers.
5997 }

```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

5998 \msg_new:nnn { enumext } { missing-starred }
5999 {
6000     Missing~'\c_backslash_str #1'~#2.
6001 }
6002 \msg_new:nnn { enumext } { many-starred }
6003 {
6004     Many~'\c_backslash_str #1'~#2.
6005 }

```

Messages used by `\printkeyans*` command.

```

6006 \msg_new:nnn { enumext } { print-starred }
6007 {

```

```

6008     \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
6009     #2~environment~ \msg_line_context:.
6010 }

```

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

```

6011 \msg_new:nnn { enumext } { list-too-deep }
6012 {
6013     Too~deep~nesting ~for~'enumext'~\msg_line_context:~ \\
6014     The~maximum ~level ~of ~nesting ~is~4.
6015 }

```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

6016 \msg_new:nnn { enumext } { anskey-unnumber-item }
6017 {
6018     Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
6019 }
6020 \msg_new:nnn { enumext } { anskey-already-stored }
6021 {
6022     Content~already~stored~for~this~\c_backslash_str item~\msg_line_context:.
6023 }
6024 \msg_new:nnn { enumext } { anskey-empty-arg }
6025 {
6026     Can't~store~empty~content~\msg_line_context:.
6027 }
6028 \msg_new:nnn { enumext } { anskey-wrong-place }
6029 {
6030     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6031     '\c_backslash_str #1'~works~in~the~environment~'#2'.
6032 }
6033 \msg_new:nnn { enumext } { anskey-nested }
6034 {
6035     The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
6036 }
6037 \msg_new:nnn { enumext } { anskey-math-mode }
6038 {
6039     #1~can't~work~in~math~mode~\msg_line_context:.
6040 }
6041 \msg_new:nnn { enumext } { anskey-env-wrong }
6042 {
6043     The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
6044 }
6045 \msg_new:nnn { enumext } { ansPIC-wrong-place }
6046 {
6047     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6048     '\c_backslash_str #1'~works~in~the~environment~'#2'.
6049 }
6050 \msg_new:nnn { enumext } { command-wrong-place }
6051 {
6052     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6053     '\c_backslash_str #1'~works~outside~the~environment~'#2'.
6054 }
6055 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
6056 {
6057     The~key~'#1'~is~unknown~by~environment~
6058     'anskey*'~and~is~being~ignored.
6059 }
6060 {
6061     The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
6062     Check~that~you~have~spelled~the~key~name~correctly.
6063 }
6064 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
6065 {
6066     The~key~'#1=#2'~is~unknown~by~environment ~
6067     'anskey*'~and~is~being~ignored.
6068 }
6069 {
6070     The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
6071     Check~that~you~have~spelled~the~key~name~correctly.
6072 }
6073 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
6074 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
6075 {

```

```

6076     The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6077     Check~that~you~have~spelled~the~key~name~correctly.
6078 }
6079 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
6080 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
6081 {
6082     The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6083     Check~that~you~have~spelled~the~key~name~correctly.
6084 }
6085 \msg_new:nnn { enumext } { overwrite-file }
6086 {
6087     Overwriting~file~'#1'.
6088 }
6089 \msg_new:nnn { enumext } { writing-file }
6090 {
6091     Writing~file~'#1'.
6092 }
6093 \msg_new:nnn { enumext } { not-writing }
6094 {
6095     File~'#1'~already~exists.~Not~writing.
6096 }

```

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

```

6097 \msg_new:nnn { enumext } { keyans-nested }
6098 {
6099     The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
6100 }
6101 \msg_new:nnn { enumext } { keyans-wrong-level }
6102 {
6103     Wrong~level~position~for~'keyans'~\msg_line_context:~ \\
6104     The~environment~'keyans'~can~only~be~in~the~first~level.
6105 }
6106 \msg_new:nnn { enumext } { wrong-place }
6107 {
6108     Wrong~place~for~'#1'~environment ~\msg_line_context:~ \\
6109     '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
6110 }
6111 \msg_new:nnn { enumext } { keyanspic-nested }
6112 {
6113     The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:~.
6114 }
6115 \msg_new:nnn { enumext } { keyanspic-wrong-level }
6116 {
6117     Wrong~level~position~for~'keyanspic'~\msg_line_context:~ \\
6118     The~environment~'keyans'~can~only~be~in~the~first~level.
6119 }
6120 \msg_new:nnn { enumext } { keyanspic-item-cmd }
6121 {
6122     Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
6123 }
6124 \msg_new:nnnn { enumext } { keyans-unknown-key }
6125 {
6126     The~key~'#1'~is~unknown~by~environment~
6127     '\l_enumext_envir_name_tl'~and~is~being~ignored.
6128 }
6129 {
6130     The~environment~'\l_enumext_envir_name_tl'~does~not
6131     ~have~a~key~called ~'#1'.\\
6132     Check~that~you~have~spelled~the~key~name~correctly.
6133 }
6134 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
6135 {
6136     The~key~'#1=#2'~is~unknown~by~environment ~
6137     '\l_enumext_envir_name_tl'~and~is~being~ignored.
6138 }
6139 {
6140     The~environment~'\l_enumext_envir_name_tl'~does~not
6141     ~have~a~key~called ~'#1'.\\
6142     Check~that~you~have~spelled~the~key~name~correctly.
6143 }

```

Message used by unknown `<keys>` in `enumext*`. environment.

```

6144 \msg_new:nnnn { enumext } { starred-unknown-key }
6145 {
6146   The~key~'#1'~is~unknown~by~environment~
6147   '\l__enumext_envir_name_tl'~and~is~being~ignored.
6148 }
6149 {
6150   The~environment~'\l__enumext_envir_name_tl'~does~not
6151   ~have~a~key~called ~'#1'.\\
6152   Check~that~you~have~spelled~the~key~name~correctly.
6153 }
6154 \msg_new:nnnn { enumext } { starred-unknown-key-value }
6155 {
6156   The~key~'#1=#2'~is~unknown~by~environment ~
6157   '\l__enumext_envir_name_tl'~and~is~being~ignored.
6158 }
6159 {
6160   The~environment~'\l__enumext_envir_name_tl'~does~not
6161   ~have~a~key~called ~'#1'.\\
6162   Check~that~you~have~spelled~the~key~name~correctly.
6163 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

6164 \msg_new:nnnn { enumext } { standar-unknown-key }
6165 {
6166   The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6167   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6168 }
6169 {
6170   The~environment~'\l__enumext_envir_name_tl'~does~not
6171   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6172   Check~that~you~have~spelled~the~key~name~correctly.
6173 }
6174 \msg_new:nnnn { enumext } { standar-unknown-key-value }
6175 {
6176   The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6177   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6178 }
6179 {
6180   The~environment~'\l__enumext_envir_name_tl'~does~not
6181   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6182   Check~that~you~have~spelled~the~key~name~correctly.
6183 }

```

Message used by unknown $\langle keys \rangle$ in `\foreachkeyans`.

```

6184 \msg_new:nnnn { enumext } { for-key-unknown }
6185 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
6186 {
6187   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6188   Check~that~you~have~spelled~the~key~name~correctly.
6189 }
6190 \msg_new:nnnn { enumext } { for-key-value-unknown }
6191 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
6192 {
6193   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6194   Check~that~you~have~spelled~the~key~name~correctly.
6195 }

```

Messages used by `\getkeyans` command.

```

6196 \msg_new:nnn { enumext } { undefined-storage-anskey }
6197 {
6198   Storage~named~'#1'~is~not~defined~\msg_line_context:.
6199 }

```

Messages used by `\miniright` command.

```

6200 \msg_new:nnn { enumext } { missing-miniright }
6201 {
6202   Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
6203   The~key~'mini-env'~need~'\c_backslash_str miniright'.
6204 }
6205 \msg_new:nnn { enumext } { wrong-miniright-place }
6206 {
6207   Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:~ \\
6208   Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.

```

```

6209     }
6210     \msg_new:nnn { enumext } { wrong-miniright-use }
6211     {
6212         Wrong-use~for~'\c_backslash_str miniright'~\msg_line_context:~ \
6213         '\c_backslash_str miniright'~need~a~key~'mini-env'.
6214     }
6215     \msg_new:nnn { enumext } { wrong-miniright-starred }
6216     {
6217         Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
6218     }
6219     \msg_new:nnn { enumext } { many-miniright-used }
6220     {
6221         Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
6222     }

```

Messages used by `\setenumextmeta` command.

```

6223     \msg_new:nnn { enumext } { unknown-set }
6224     {
6225         Argument~[#1]~is~unknown~by~ \c_backslash_str setenumextmeta~\msg_line_context:.
6226     }
6227     \msg_new:nnn { enumext } { already-defined }
6228     {
6229         The~key~'#1'~is~already~defined~\msg_line_context:.
6230     }
6231     \msg_new:nnn { enumext } { prohibited-unknown }
6232     {
6233         The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
6234     }

```

Messages used by `enumext*` and `keyans*` environments.

```

6235     \msg_new:nnn { enumext } { nested }
6236     {
6237         The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
6238     }
6239     \msg_new:nnn { enumext } { nested-horizontal }
6240     {
6241         The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_context:.
6242     }
6243     \msg_new:nnn { enumext } { item-joined }
6244     {
6245         Items~joined~( #1 )~>~#2 ~columns ~\msg_line_context:.
6246     }
6247     \msg_new:nnn { enumext } { item-joined-columns }
6248     {
6249         Not~space~to~join~items~( #1 )~>~#2 ~\msg_line_context:.
6250     }

```

Messages used by `resume` key.

```

6251     \msg_new:nnn { enumext } { unknown-series-starred }
6252     {
6253         The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
6254         ~enumext*~environment~ \msg_line_context:.
6255     }
6256     \msg_new:nnn { enumext } { unknown-series-standar }
6257     {
6258         The~series~'#1'~for~the~resume~key~does~not~exist~at~level~\int_use:N \l__enumext_level_int
6259         \c_space_tl of~enumext~environment~ \msg_line_context:.
6260     }
6261     \msg_new:nnnn { enumext-reset } { invalid-clist }
6262     { The~argument~must~have~1~or~2~elements~separated~by~a~comma. }
6263     { Received:~'#1'. }
6264     \msg_new:nnn { enumext-reset } { invalid-single-arg-star }
6265     { The~single~argument~must~be~exactly~'enumext*'~when~using~a~'*. }
6266     \msg_new:nnnn { enumext-reset } { invalid-single-arg-no-star }
6267     { A~single~argument~is~not~allowed~without~a~'*. }
6268     { Received:~'#1'. }
6269     \msg_new:nnnn { enumext-reset } { out-of-range }
6270     { The~number~must~be~exactly~1,~2,~3~or~4. }
6271     { Received:~'#1'. }
6272     \msg_new:nnnn { enumext-reset } { invalid-package }
6273     { The~first~element~must~be~exactly~'enumext'. }
6274     { Received:~'#1'. }

```

13.55 Finish package

Finish package implementation.

```
6275 \file_input_stop:
6276 </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.

Symbols	
<code>\+</code>	223
<code>\-</code>	223
<code>\\</code> 231, 4612, 4615, 5853, 5862, 5867, 5891, 5893, 5900, 5902, 5915, 5920, 5925, 5940, 5979, 5981, 5983, 5988, 5989, 5994, 5995, 6013, 6030, 6047, 6052, 6061, 6070, 6076, 6082, 6103, 6108, 6117, 6131, 6141, 6151, 6161, 6171, 6181, 6187, 6193, 6202, 6207, 6212	
A	
<code>above</code>	<u>1729</u>
<code>above*</code>	<u>1729</u>
<code>\addvspace</code> 1296, 1324, 1367, 1370, 1538, 1541, 1638, 1644, 1682, 1688, 1709, 1715, 4024, 4196, 4214, 4497, 4501, 4860, 4875, 4921, 4935	
<code>after</code>	<u>1126</u>
<code>align</code>	<u>666</u>
<code>\Alph</code>	44, <u>49</u>
<code>\Alph</code>	608, 736, 780, 840, 5546
<code>\alph</code>	44, <u>49</u>
<code>\alph</code>	609, 734, 5534
<code>\anskey</code>	14, 88, 90, <u>3051</u>
<code>anskey*</code>	15, <u>3181</u>
<code>\anspic</code>	18, 117, 120, <u>4511</u>
<code>\anspic*</code>	81
<code>\arabic</code>	44
<code>\arabic</code>	607, 733, 779, 5522, 5528, 5552
B	
<code>base-fix</code>	<u>984</u>
<code>\baselineskip</code>	58
<code>\baselineskip</code>	1000, 1011
<code>before</code>	<u>1126</u>
<code>before*</code>	<u>1126</u>
<code>beginpenalty</code>	<u>924</u>
<code>below</code>	<u>1729</u>
<code>below*</code>	<u>1729</u>
bool commands:	
<code>\bool_gset_false:N</code> 342, 343, 344, 4877, 4881, 4937	
<code>\bool_gset_true:N</code> 252, 262, 1229, 2484, 2490, 4846, 4878, 4910, 4938	
<code>\bool_if:NTF</code> 392, 402, 419, 493, 500, 509, 516, 530, 543, 1751, 1765, 1778, 1789, 1800, 1811, 1822, 1833, 1847, 1861, 1882, 1929, 1934, 1945, 1950, 2001, 2045, 2429, 2672, 2682, 2762, 2786, 2793, 2817, 2915, 2937, 2977, 3001, 3005, 3055, 3074, 3098, 3150, 3154, 3184, 3202, 3221, 3237, 3260, 3291, 3306, 3378, 3494, 3528, 3564, 3580, 3601, 3747, 3768, 3814, 3857, 3867, 3901, 3906, 3931, 3940, 3979, 4005, 4055, 4073, 4124, 4179, 4204, 4430, 4495, 4513, 4532, 4583, 4610, 4839, 4855, 4861, 4904, 4918, 4922, 4990, 5011, 5018, 5028, 5116, 5122, 5129, 5145, 5239, 5249, 5363, 5370, 5401, 5417, 5448	
<code>\bool_if:nTF</code> 1689, 1716, 2248, 3550, 3726, 4553, 5574, 5728	
<code>\bool_if_p:N</code> .. 271, 285, 994, 995, 1007, 1008, 1661, 2125, 2126, 2141, 2142, 2189, 2190, 2205, 2206, 2442, 2468, 2481, 2482, 2487, 2488, 2850, 2860, 2872, 2887, 2888, 2922, 2963, 2964, 3365, 3366, 3395, 3396, 3408, 3409, 3449, 3450, 3469, 3470, 3760, 3761, 3762, 3952, 3954, 3965, 5392, 5393, 5394	
<code>\bool_lazy_all:nTF</code> 269, 283, 992, 2440, 2466, 2848, 2857, 2870, 2885, 3447, 3467, 3758, 3950, 3963, 5390	
<code>\bool_lazy_and:nnTF</code> .. 248, 258, 1006, 1653, 1660, 2124, 2140, 2188, 2204, 2261, 2480, 2486, 2921, 2928, 2962, 3364	
<code>\bool_lazy_or:nnTF</code> .. 2370, 2377, 3394, 3407, 5751	
<code>\bool_new:N</code> 22, 23, 24, 25, 26, 27, 28, 47, 50, 51, 52, 63, 87, 92, 93, 98, 99, 102, 109, 124, 125, 137, 138, 145, 151, 152, 154, 158, 160, 161, 178, 190, 192	
<code>\bool_not_p:n</code> 249, 259, 996, 1662, 2859, 2923, 2929, 3953, 3966	
<code>\bool_set_eq:NN</code>	3503, 3707, 5069, 5314
<code>\bool_set_false:N</code> 399, 869, 1018, 2414, 2415, 2447, 2452, 2456, 2460, 2473, 3735, 3920, 4072, 4132, 4219, 4366, 4427, 4573, 4981, 5010, 5066, 5255, 5310, 5311, 5565, 5587, 5588	
<code>\bool_set_true:N</code> 276, 290, 385, 388, 659, 1033, 1735, 1740, 1867, 1871, 1888, 1891, 2163, 2170, 2387, 2388, 2704, 2712, 3125, 3497, 3499, 3531, 3533, 3703, 3714, 3728, 3880, 3919, 3959, 3972, 4045, 4129, 4156, 4363, 4555, 4556, 4828, 4893, 4980, 5073, 5080, 5081, 5125, 5253, 5318, 5325, 5326, 5327, 5558, 5582, 5583	
box commands:	
<code>\box_dp:N</code> .. 1584, 1585, 1588, 1595, 1608, 1616, 1622, 1630, 4441, 4447, 4497, 4594	
<code>\box_ht:N</code> .. 1367, 1370, 1381, 1382, 1393, 1395, 1410, 1413, 1421, 1422, 1433, 1435, 1450, 1453, 1460, 1461, 1472, 1474, 1489, 1492, 1538, 1541, 1549, 1550, 1558, 1559, 1571, 1573	
<code>\box_ht_plus_dp:N</code>	4436, 4505, 4541
<code>\box_new:N</code>	60, 147, 148, 185, 191
<code>\box_use_drop:N</code>	4872, 4933, 5181, 5460
<code>\box_wd:N</code>	615
<code>break-col</code>	<u>3021</u> , <u>3107</u>
C	
<code>\c</code>	875, 877, 889, 891
c@ internal commands:	
<code>\c@_enumext_resume_i_int</code>	<u>593</u>
<code>\c@_enumext_resume_ii_int</code>	<u>593</u>
<code>\c@_enumext_resume_iii_int</code>	<u>593</u>
<code>\c@_enumext_resume_iv_int</code>	<u>593</u>
<code>\c@_enumext_resume_vii_int</code>	<u>593</u>
<code>\centering</code>	1691, 1718, 4638, 4865, 4926
<code>check-ans</code>	<u>2406</u>
Document class:	
<code>article</code>	51
clist commands:	
<code>\clist_const:Nn</code>	197
<code>\clist_map_function:nN</code>	4621
<code>\clist_map_inline:Nn</code> .. 665, 923, 939, 1125, 1140, 1221, 1745	
<code>\clist_map_inline:nn</code> 36, 45, 56, 68, 76, 89, 101, 140, 169, 196, 601, 643, 696, 716, 1038, 1059, 1235, 1855, 1878, 2288, 2295, 2310, 2354, 2420, 2599, 2669, 2701, 2845, 3300, 3622, 3637, 3684, 3843, 3846, 3848, 3875, 3887, 3890, 3892, 3911, 5704	
<code>\columnbreak</code>	88
<code>\columnbreak</code>	2925
<code>columns</code>	<u>1205</u>

columns-sep	1205
\columnsep	109
\columnsep	4000, 4177
\columnseprule	109
\columnseprule	4003, 4178
Commands provide by enumext :	
\anskey . 33, 77, 78, 83, 85–87, 89, 90, 95, 109, 129, 138, 140, 148	
\anspic* 33, 34, 81, 85, 95, 96, 119, 120, 138, 140	
\anspic	34, 85, 117, 120, 148
\foreachkeyans	144, 150
\getkeyans	85, 138, 150
\item* 33, 34, 81, 85, 95, 96, 99, 103, 130, 131, 136, 138, 140	
\item	99, 103, 124, 130, 132, 135
\miniright	32, 56, 63–65, 111, 150
\printkeyans*	139
\printkeyans	33, 85, 139, 140
\resetenumext	74
\setenumextmeta	143, 151
\setenumext	33, 140–142, 146
Counters defined by enumext :	
enumXiii	31, 43
enumXii	31, 43
enumXiv	31, 43
enumXi	31, 43
enumXviii	31, 43
enumXvii	31, 43, 131
enumXvi	31, 43
enumXv	31, 43
\counterwithin	2282, 2283
cs commands:	
\cs_generate_variant:Nn . 202, 203, 617, 633, 881, 897, 2285, 2754, 2759, 2835, 3180, 3833, 4623, 5763	
\cs_if_exist:NTF	578, 595
\cs_if_exist_p:N	3763, 5395
\cs_new:Nn	217
\cs_new:Npn . 227, 1897, 1906, 1914, 2716, 2725, 2733, 5602, 5611, 5619	
\cs_new_eq:NN . 369, 370, 375, 376, 404, 405, 408, 409	
\cs_new_protected:Nn . 233, 241, 267, 298, 328, 334, 340, 346, 352, 360, 380, 427, 431, 449, 461, 479, 491, 507, 523, 536, 557, 756, 813, 860, 990, 1141, 1145, 1149, 1153, 1157, 1161, 1165, 1169, 1173, 1177, 1181, 1185, 1189, 1193, 1197, 1201, 1236, 1248, 1281, 1298, 1309, 1326, 1352, 1373, 1498, 1524, 1544, 1577, 1599, 1634, 1640, 1746, 1760, 1774, 1785, 1796, 1807, 1818, 1829, 1999, 2017, 2043, 2058, 2113, 2174, 2221, 2318, 2328, 2342, 2355, 2360, 2385, 2425, 2435, 2478, 2493, 2500, 2509, 2514, 2519, 2524, 2533, 2538, 2543, 2760, 2784, 2791, 2815, 2822, 2836, 3072, 3091, 3200, 3219, 3250, 3289, 3304, 3332, 3362, 3390, 3403, 3416, 3445, 3458, 3536, 3546, 3557, 3573, 3589, 3722, 3740, 3774, 3786, 3912, 3948, 3977, 3984, 4014, 4031, 4053, 4078, 4092, 4122, 4146, 4163, 4188, 4202, 4223, 4234, 4402, 4605, 4619, 4624, 4648, 4658, 4689, 4818, 4837, 4883, 4902, 4967, 4997, 5004, 5016, 5026, 5051, 5192, 5237, 5268, 5274, 5295, 5351, 5471	
\cs_new_protected:Npn 204, 205, 209, 213, 412, 576, 602, 612, 618, 737, 781, 845, 867, 882, 1673, 1702, 1923, 1985, 2083, 2098, 2157, 2272, 2276, 2280, 2286, 2293, 2365, 2548, 2670, 2680, 2702, 2710, 2746, 2755, 2911, 2974, 2999, 3037, 3041, 3134, 3138, 3171, 3230, 3269, 3342, 3383, 3490, 3509, 3638, 3642, 3659, 3663, 3685, 3689, 3699, 3711, 3756, 3802, 3836, 3878, 3923,	

4142, 4411, 4418, 4425, 4530, 4549, 4579, 4720, 4769, 4984, 5057, 5064, 5078, 5086, 5091, 5101, 5261, 5301, 5308, 5323, 5332, 5346, 5388, 5493, 5506, 5568, 5701, 5713, 5737, 5749, 5787, 5797, 5805, 5827	
\cs_new_protected_nopar:Nn . . . 4291, 4335, 4343, 4351, 5036, 5044, 5175, 5280, 5288, 5454	
\cs_new_protected_nopar:Npn . . 4283, 4299, 5107, 5153, 5407, 5428	
\cs_set:Npn 2115, 2176, 2223, 2330, 2846, 2883, 5499	
\cs_set_eq:NN . . 3766, 4957, 4958, 5155, 5226, 5227, 5398, 5430	
\cs_set_protected:Nn 1064, 1080, 1093, 1105	
\cs_set_protected:Npn 32, 39, 48, 61, 69, 84, 90, 133, 165, 176, 593, 634, 644, 666, 701, 717, 763, 898, 924, 940, 1020, 1043, 1117, 1126, 1205, 1222, 1729, 1840, 1856, 2300, 2346, 2406, 2565, 2600, 2688, 2838, 3293, 3611, 3627, 3673, 3834, 3876	
\cs_to_str:N	604, 627

D

\d	223
\DeclareDocumentEnvironment	561
dim commands:	
\dim_abs:n	3807, 3812
\dim_add:Nn	3442, 4445, 4683, 4714
\dim_compare:nNnTF . . 1066, 1082, 1095, 1107, 1385, 1397, 1425, 1437, 1464, 1476, 1553, 1561, 1675, 1704, 2979, 2987, 3437, 3804, 3809, 3815, 3821, 3823, 3825, 3989, 4036, 4150, 4167, 4420, 4660, 4676, 4691, 4707, 4820, 4885, 5359	
\dim_compare:nTF	2947, 4081, 4226
\dim_eval:n	1000, 4503, 4590
\dim_gset_eq:NN	4829, 4894
\dim_gzero:N	4880, 4940
\dim_new:N . 57, 64, 65, 66, 86, 129, 130, 142, 149, 150, 184, 186, 187, 193	
\dim_set:Nn . 615, 1034, 2981, 2989, 3424, 3428, 3433, 3439, 3526, 3807, 3812, 3814, 3817, 3818, 3822, 3824, 3827, 3828, 3830, 3992, 4039, 4080, 4152, 4169, 4225, 4434, 4539, 4626, 4662, 4669, 4693, 4700, 4755, 4804, 4822, 4887, 5103, 5361	
\dim_set_eq:NN 724, 770, 837, 3521, 3845, 3889, 4000, 4177, 4762, 4765, 4766, 4811, 4814, 4815, 5096, 5167, 5442	
\dim_sub:Nn	4086, 4231, 4678, 4709
\dim_use:N . 1067, 1075, 1676, 1686, 2825, 2828, 2833, 2991, 3541, 3543, 3596, 3990, 3994, 3995, 3997, 4037, 4042, 4043, 4049, 4083, 4088	
\dim_zero:N	3881, 4003, 4178, 4448
\dim_zero_new:N	575
\c_zero_dim 1069, 1083, 1096, 1108, 1676, 1704, 2949, 2979, 2987, 3424, 3437, 3804, 3809, 3815, 3822, 3990, 4037, 4083, 4150, 4167, 4228, 4420, 4660, 4676, 4691, 4707, 4820, 4885, 5359	
\dimeval	2634

E

\end . . . 2788, 2819, 4021, 4193, 4485, 4640, 5576, 5586, 5594	
end internal commands:	
\end__enumext_mini_page . 1684, 1711, 4064, 4213, 4844, 4908, 4934	
\endlist	370
\endminipage	376
endpenalty	924
enumext	5, 4099

enumext internal commands:

__enumext_add_meta_key:nnn .. [143](#), [5715](#), [5731](#),
 [5732](#), [5734](#), [5737](#)
 __enumext_add_pre_parsep: . [56](#), [1246](#), [1248](#), [1248](#)
 __enumext_after_args_exec: [54](#), [1141](#), [1153](#), [4113](#)
 __enumext_after_args_exec_v: [1157](#), [1169](#), [4251](#)
 __enumext_after_args_exec_vii: .. [1173](#), [1197](#)
 __enumext_after_args_exec_viii: [1201](#)
 __enumext_after_env:nn [95](#), [112](#), [126](#), [133](#), [209](#), [209](#),
 [549](#), [553](#), [4118](#), [4853](#), [4916](#), [5208](#)
 __enumext_after_hyperref: ... [39](#), [378](#), [378](#), [380](#)
 \l__enumext_after_list_args_v_tl [1171](#)
 \l__enumext_after_list_args_vii_tl [1199](#), [5173](#)
 \l__enumext_after_list_args_viii_tl .. [1203](#),
 [5452](#)
 __enumext_after_list_vii: [126](#), [129](#), [4965](#), [5004](#),
 [5004](#)
 __enumext_after_list_viii: .. [134](#), [5235](#), [5274](#),
 [5274](#)
 __enumext_after_stop_list: [54](#), [111](#), [1141](#), [1149](#),
 [4069](#)
 __enumext_after_stop_list_v: [1157](#), [1165](#), [4220](#)
 \l__enumext_after_stop_list_v_tl [1167](#)
 __enumext_after_stop_list_vii: .. [129](#), [1173](#),
 [1189](#), [5007](#)
 \l__enumext_after_stop_list_vii_tl ... [1191](#)
 __enumext_after_stop_list_viii: . [1193](#), [5277](#)
 \l__enumext_after_stop_list_viii_tl ... [1195](#)
 \l__enumext_align_label_pos_v_str [3420](#), [3792](#)
 \l__enumext_align_label_pos_X_str [69](#)
 \l__enumext_align_label_vii_str [5142](#)
 \l__enumext_align_label_viii_str [5421](#)
 \l__enumext_align_label_X_str [176](#)
 \c__enumext_all_envs_clist .. [197](#), [665](#), [923](#), [939](#),
 [1125](#), [1140](#), [1221](#), [1745](#)
 \c__enumext_all_families_seq .. [142](#), [5669](#), [5695](#)
 __enumext_anskey_env_file_if_writable:n [92](#),
 [3148](#), [3148](#)
 __enumext_anskey_env_file_if_-
 writable:nTF [3148](#), [3173](#)
 __enumext_anskey_env_file_write:nn [93](#), [3171](#),
 [3180](#), [3235](#)
 \l__enumext_anskey_env_force_eol_bool .. [94](#),
 [3121](#), [3237](#)
 \c__enumext_anskey_env_hidden_space_str [33](#),
 [94](#), [112](#), [3241](#)
 \l__enumext_anskey_env_overwrite_bool [3129](#),
 [3154](#)
 __enumext_anskey_env_safe_inner: . [94](#), [3195](#),
 [3200](#), [3219](#)
 __enumext_anskey_env_safe_inner:n [93](#)
 __enumext_anskey_env_safe_outer: . [93](#), [3183](#),
 [3200](#), [3200](#)
 __enumext_anskey_env_unknown:n [92](#), [3132](#), [3134](#),
 [3134](#)
 __enumext_anskey_env_unknown:nn . [3134](#), [3136](#),
 [3138](#)
 \l__enumext_anskey_level_int .. [16](#), [3093](#), [3094](#)
 __enumext_anskey_safe_inner: . [91](#), [3066](#), [3072](#),
 [3091](#)
 __enumext_anskey_safe_inner:n [90](#)
 __enumext_anskey_safe_outer: . [90](#), [3053](#), [3072](#),
 [3072](#)
 __enumext_anskey_show_wrap_arg:n . [89](#), [2974](#),
 [2974](#), [3003](#), [3018](#)
 __enumext_anskey_show_wrap_left:n [89](#), [2919](#),
 [2999](#), [2999](#)
 __enumext_anskey_unknown:n [90](#), [3021](#), [3035](#), [3037](#)
 __enumext_anskey_unknown:nn . [3021](#), [3039](#), [3041](#)
 __enumext_anskey_wrapper:n [2631](#), [2997](#)
 \l__enumext_anspic_above_int . [141](#), [4627](#), [4628](#),
 [4630](#)
 __enumext_anspic_args:nnn [120](#), [121](#), [4511](#), [4527](#),
 [4605](#)
 \l__enumext_anspic_args_seq [120](#)–[122](#), [141](#), [4525](#),
 [4639](#), [4652](#)
 \l__enumext_anspic_below_int . [141](#), [4627](#), [4628](#),
 [4631](#)
 \l__enumext_anspic_body_box ... [141](#), [4538](#), [4541](#)
 __enumext_anspic_body_dim:n .. [120](#), [4511](#), [4530](#),
 [4582](#)
 \l__enumext_anspic_body_htdp_dim .. [120](#), [141](#),
 [4539](#), [4593](#)
 __enumext_anspic_exec: [4511](#)
 __enumext_anspic_exec: ... [119](#), [122](#), [4480](#), [4648](#)
 __enumext_anspic_label:nn [120](#), [4511](#), [4549](#), [4585](#),
 [4600](#)
 \l__enumext_anspic_label_above_bool ... [141](#),
 [4363](#), [4366](#), [4430](#), [4495](#), [4532](#), [4583](#), [4610](#)
 \l__enumext_anspic_label_box .. [141](#), [4433](#), [4436](#)
 \l__enumext_anspic_label_htdp_dim . [118](#), [141](#),
 [4434](#), [4440](#), [4505](#), [4592](#)
 __enumext_anspic_label_pos:nnn .. [121](#), [4511](#),
 [4579](#), [4608](#)
 \l__enumext_anspic_label_sep_skip [4373](#), [4442](#),
 [4506](#), [4595](#), [4612](#)
 \l__enumext_anspic_layout_style_tl [4375](#), [4650](#),
 [4655](#)
 \l__enumext_anspic_mini_pos_str .. [141](#), [4364](#),
 [4367](#), [4637](#)
 \l__enumext_anspic_mini_width_dim [141](#), [4551](#),
 [4626](#), [4637](#)
 __enumext_anspic_print:n [121](#), [122](#), [4511](#), [4619](#),
 [4623](#), [4652](#), [4655](#)
 __enumext_anspic_row:n .. [122](#), [4511](#), [4621](#), [4624](#)
 __enumext_anspic_start_list_tag: [4307](#), [4335](#),
 [4607](#)
 __enumext_anspic_stop_list_tag: . [4307](#), [4351](#),
 [4617](#)
 __enumext_anspic_stop_start_list_tag: [4307](#),
 [4343](#), [4609](#)
 __enumext_at_begin_document:n .. [39](#), [205](#), [205](#),
 [367](#), [373](#)
 \l__enumext_base_line_fix_bool [51](#), [140](#), [986](#), [995](#),
 [1018](#), [5582](#), [5587](#)
 __enumext_before_args_exec: [54](#), [110](#), [128](#), [1141](#),
 [1141](#), [4034](#)
 __enumext_before_args_exec_v: [1157](#), [1157](#), [4149](#)
 __enumext_before_args_exec_vii: . [1173](#), [1173](#),
 [5001](#)
 __enumext_before_args_exec_viii: [1177](#), [5271](#)
 __enumext_before_env:nn [209](#), [213](#)
 __enumext_before_keys_exec: .. [54](#), [1141](#), [1145](#),
 [4109](#)
 __enumext_before_keys_exec_v: [1157](#), [1161](#), [4247](#)
 __enumext_before_keys_exec_vii [1173](#)
 __enumext_before_keys_exec_vii: . [1181](#), [4951](#)
 __enumext_before_keys_exec_viii: [1185](#), [5220](#)

```

\__enumext_before_list: .. 110, 4031, 4031, 4103
\__enumext_before_list_v: ... 4146, 4146, 4242
\__enumext_before_list_vii: ... 128, 4946, 4997,
    4997
\__enumext_before_list_viii: . 134, 5216, 5268,
    5268
\l__enumext_before_no_starred_key_v_tl 1163
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1183
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1187
\l__enumext_before_starred_key_v_tl ... 1159
\l__enumext_before_starred_key_vii_tl . 1175
\l__enumext_before_starred_key_viii_tl 1179
\__enumext_calc_hspace:NNNNNN 106, 3802, 3802,
    3833, 3838, 3882
\__enumext_check_ans_active: 78, 110, 128, 2425,
    2425, 4035, 5000
\g__enumext_check_ans_item_tl ..... 97
\g__enumext_check_ans_key_bool 79, 80, 151, 342,
    2484, 2490, 3260
\l__enumext_check_ans_key_bool 79, 2410, 2415,
    2481, 2487
\__enumext_check_ans_key_hook: .. 79, 111, 129,
    2478, 2478, 4070, 5008
\__enumext_check_ans_level: . 78, 79, 2425, 2431,
    2435
\__enumext_check_ans_log: 80, 94, 2524, 2524, 3264
\__enumext_check_ans_log_msg_greater: 2524,
    2530, 2543
\__enumext_check_ans_log_msg_less: 2524, 2528,
    2533
\__enumext_check_ans_log_msg_same_ok: 2524,
    2529, 2538
\__enumext_check_ans_msg_greater: 2500, 2506,
    2519
\__enumext_check_ans_msg_less: 2500, 2504, 2509
\__enumext_check_ans_msg_same_ok: 2500, 2505,
    2514
\__enumext_check_ans_show: .. 80, 94, 2500, 2500,
    3262
\l__enumext_check_answers_bool 77, 78, 90, 93, 99,
    100, 151, 2388, 2414, 2429, 2762, 2786, 2793, 2817,
    3055, 3184, 3378, 3494, 3528, 5122
\__enumext_check_starred_cmd:n 37, 81, 97, 133,
    2548, 2548, 4254, 4493, 5234
\g__enumext_check_starred_cmd_int . 104, 151,
    2551, 2557, 2562, 3720, 4561, 5358
\l__enumext_check_start_line_env_tl . 37, 151,
    305, 313, 321, 2554, 2560, 2563
\l__enumext_columns_sep_v_dim 4167, 4169, 4177
\l__enumext_columns_sep_vii_dim .. 4660, 4662,
    4671, 4683, 4759, 5189
\l__enumext_columns_sep_viii_dim . 4691, 4693,
    4702, 4714, 4808, 5468
\l__enumext_columns_v_int 1518, 1536, 1707, 4165,
    4173, 4185, 4190
\l__enumext_columns_vii_int .. 4665, 4668, 4672,
    4681, 4723, 4727, 4730, 4736, 4742, 4746, 5183, 5197
\l__enumext_columns_viii_int . 4696, 4699, 4703,
    4712, 4772, 4776, 4779, 4785, 4791, 4795, 5462, 5477
\l__enumext_counter_i_tl ..... 32, 585
\l__enumext_counter_ii_tl ..... 32, 586
\l__enumext_counter_iii_tl ..... 32, 587
\l__enumext_counter_iv_tl ..... 32, 588
\g__enumext_counter_styles_tl . 31, 44, 57, 605,
    623
\l__enumext_counter_v_tl ..... 32, 589
\l__enumext_counter_vi_tl ..... 32, 590
\l__enumext_counter_vii_tl ..... 32, 591
\l__enumext_counter_viii_tl ..... 32, 592
\l__enumext_current_widest_dim 31, 57, 629, 725,
    771, 838
\__enumext_def_meta_key:nnn .. 143, 5715, 5743,
    5749, 5763
\__enumext_default_item:n ... 3490, 3490, 3554
\__enumext_define_counter:Nn . 31, 576, 576, 585,
    586, 587, 588, 589, 590, 591, 592
\__enumext_endminipage: . 39, 367, 376, 570, 4874,
    5177, 5456
\g__enumext_envir_name_tl 37, 22, 277, 291, 350,
    2358, 2363, 2373, 2512, 2517, 2522, 2536, 2541, 2546
\l__enumext_envir_name_tl . 36, 37, 102, 22, 247,
    257, 304, 312, 320, 3632, 3655, 3679, 4398, 6127, 6130,
    6137, 6140, 6147, 6150, 6157, 6160, 6166, 6170, 6176,
    6180, 6237, 6241
\__enumext_execute_after_env: .. 38, 77, 80, 94,
    3250, 3250, 4120, 5210
\__enumext_fake_item_indent: . 1064, 1064, 3866
\l__enumext_fake_item_indent_v_dim 1083, 1088
\l__enumext_fake_item_indent_v_tl 1085, 3704,
    3708, 3715
\__enumext_fake_item_indent_vii: . 1064, 1093,
    3900
\l__enumext_fake_item_indent_vii_dim . 1096,
    1100
\l__enumext_fake_item_indent_vii_tl .. 1098,
    5172
\__enumext_fake_item_indent_viii: 1064, 1105,
    3905
\l__enumext_fake_item_indent_viii_dim 1108,
    1112
\l__enumext_fake_item_indent_viii_tl . 1110,
    5447
\l__enumext_fake_item_indent_X_tl ..... 90
\__enumext_fake_make_label_vii:n . 131, 5107,
    5107, 5164
\__enumext_fake_make_label_viii:n 5388, 5407,
    5439
\__enumext_filter_level:n 141, 5602, 5602, 5632,
    5638, 5644, 5650, 5656
\__enumext_filter_level_key:n 141, 5602, 5607,
    5611
\__enumext_filter_level_pair:nn .. 141, 5602,
    5608, 5619
\__enumext_filter_save_key:n .. 84, 2677, 2685,
    2708, 2714, 2716, 2716, 5519, 5525, 5531, 5537, 5543,
    5549
\__enumext_filter_save_key_key:n .. 84, 2716,
    2721, 2725
\__enumext_filter_save_key_pair:nn 84, 2716,
    2722, 2733
\__enumext_filter_series:n 68, 1897, 1897, 1966,
    1977, 1991, 1996
\__enumext_filter_series_key:n 68, 1897, 1902,
    1906
\__enumext_filter_series_pair:nn .. 69, 1897,
    1903, 1914
\__enumext_first_item_tmp_vii: 127, 129, 4957,
    5036, 5036

```

```

\__enumext_first_item_tmp_viii: .. 135, 5226,
    5280, 5280
\g__enumext_footnote_standar_arg_seq .. 170,
    444, 455, 458
\g__enumext_footnote_standar_int 170, 438, 441,
    443, 446
\g__enumext_footnote_standar_int_seq .. 170,
    446, 451, 454, 459
\g__enumext_footnote_starred_arg_seq .. 170,
    474, 485, 488
\g__enumext_footnote_starred_int 170, 468, 471,
    473, 476
\g__enumext_footnote_starred_int_seq .. 170,
    476, 481, 484, 489
\__enumext_footnotes_key_bool .. 39
\l__enumext_footnotes_key_bool 34, 39, 160, 388,
    392, 399, 500, 516, 530, 543
\__enumext_footnotetext:nn .. 427, 427, 456, 486
\__enumext_foreach_add_body:n 144, 5764, 5824,
    5827
\l__enumext_foreach_after_tl .. 5768, 5836
\l__enumext_foreach_before_tl .. 5766, 5831
\g__enumext_foreach_default_keys_tl ... 144
\l__enumext_foreach_default_keys_tl ... 119,
    5786, 5807
\__enumext_foreach_keyans:nn . 144, 5764, 5803,
    5805
\l__enumext_foreach_name_prop_tl . 119, 5809,
    5834
\l__enumext_foreach_print_seq 119, 5819, 5825,
    5829
\l__enumext_foreach_sep_tl .. 5778, 5825
\l__enumext_foreach_start_int .... 5770, 5821
\l__enumext_foreach_step_int .... 5774, 5822
\l__enumext_foreach_stop_int . 5772, 5814, 5816,
    5823
\__enumext_foreach_wrapper:n .. 5776, 5832
\__enumext_getkeyans:nn .. 139, 5488, 5502, 5506
\__enumext_getkeyans_aux:n 139, 5488, 5490, 5493
\l__enumext_hyperref_bool . 34, 39, 40, 160, 385,
    402, 419, 2964, 3366, 5116
\__enumext_hypertarget:nn 40, 378, 404, 408, 424
\__enumext_if_is_int:n .. 221
\__enumext_if_is_int:nTF .. 221, 870, 884
\__enumext_internal_mini_page: 42, 108, 128, 557,
    557, 3915, 4970
\__enumext_is_not_nested: . 31, 36, 108, 128, 241,
    241, 3914, 4969
\__enumext_is_on_first_level: . 31, 37, 108, 128,
    241, 267, 3921, 4982
\g__enumext_item_anskey_int 90, 97, 151, 337, 364,
    365, 2497, 2913, 3380
\__enumext_item_answer_diff: 80, 94, 2493, 2493,
    3257
\g__enumext_item_answer_diff_int 80, 151, 338,
    2495, 2502, 2526
\l__enumext_item_column_pos_vii_int 130, 4730,
    4736, 4742, 4746, 4753, 5047, 5183, 5186
\l__enumext_item_column_pos_viii_int .. 135,
    4779, 4785, 4791, 4795, 4802, 5291, 5462, 5465
\l__enumext_item_column_pos_X_int .. 176
\g__enumext_item_count_all_vii_int 130, 4754,
    5048, 5197, 5205
\g__enumext_item_count_all_viii_int 135, 4803,
    5292, 5476, 5485
\g__enumext_item_count_all_X_int .. 176
\g__enumext_item_number_bool .. 151
\l__enumext_item_number_bool 79, 158, 2447, 2452,
    2456, 2460, 2473, 3098, 3221, 3497, 3531, 5125
\g__enumext_item_number_int .. 79, 151, 336, 363,
    365, 2446, 2451, 2455, 2459, 2472, 2497, 3496, 3530,
    5124
\__enumext_item_peek_args_vii: 130, 5044, 5049,
    5051
\__enumext_item_peek_args_viii: .. 135, 5288,
    5293, 5295
\__enumext_item_starred_exec: . 100, 3509, 3536,
    3578, 3599
\__enumext_item_starred_exec:nn .. 3509, 3509,
    3552
\l__enumext_item_starred_vii_bool 5066, 5080,
    5129
\l__enumext_item_starred_viii_bool 5310, 5325,
    5417, 5448
\l__enumext_item_starred_X_bool .. 176
\__enumext_item_std:w .. 39, 99, 100, 104, 367, 371,
    3500, 3506, 3534, 3704, 3708, 3715
\g__enumext_item_symbol_aux_tl 100, 123, 3514,
    3517, 3542, 3586, 3606
\g__enumext_item_symbol_aux_vii_tl 5088, 5131,
    5134, 5138, 5140
\g__enumext_item_symbol_aux_X_tl .. 176
\l__enumext_item_symbol_sep_vii_dim .. 5096,
    5103, 5137, 5139
\l__enumext_item_symbol_vii_tl .. 5134
\l__enumext_item_text_vii_box .... 5156, 5181
\l__enumext_item_text_viii_box ... 5431, 5460
\l__enumext_item_text_X_box .. 176
\l__enumext_item_width_vii_dim ... 4669, 4678,
    4757, 4765, 4766
\l__enumext_item_width_viii_dim .. 4700, 4709,
    4806, 4814, 4815
\l__enumext_item_width_X_dim .. 176
\l__enumext_item_wrap_key_bool 104, 151, 3450,
    3470, 3728, 3735, 3762, 4555, 4573, 5311, 5326, 5394
\l__enumext_itemindent_X_dim .. 61
\l__enumext_itemsep_i_skip ... 1379, 1386, 1389,
    1391, 1398, 1402, 1405, 1407, 1547, 1554, 1556, 1557,
    1562, 1566, 1568, 1569
\l__enumext_itemsep_ii_skip .. 1419, 1426, 1429,
    1431, 1438, 1442, 1445, 1447
\l__enumext_itemsep_iii_skip . 1458, 1465, 1468,
    1470, 1477, 1481, 1484, 1486
\l__enumext_itemsep_vii_skip .. 5203
\l__enumext_itemsep_viii_skip .. 5483
\l__enumext_joined_item_aux_vii_int .. 4751,
    4752, 4753, 4754, 4760
\l__enumext_joined_item_aux_viii_int . 4800,
    4801, 4802, 4803, 4809
\l__enumext_joined_item_aux_X_int .... 176
\__enumext_joined_item_vii:w .. 130, 5044, 5054,
    5055, 5057
\l__enumext_joined_item_vii_int .. 4722, 4723,
    4726, 4728, 4734, 4739, 4744, 4749, 4751, 4757
\__enumext_joined_item_viii:w . 135, 5288, 5298,
    5299, 5301
\l__enumext_joined_item_viii_int . 4771, 4772,
    4775, 4777, 4783, 4788, 4793, 4798, 4800, 4806
\l__enumext_joined_item_X_int .. 176

```

```

\l__enumext_joined_width_vii_dim . 4755, 4762,
    4765, 5158, 5166
\l__enumext_joined_width_viii_dim 4804, 4811,
    4814, 5433, 5441
\l__enumext_joined_width_X_dim . . . . . 176
\__enumext_keyans_addto_prop:n 95, 3269, 3269,
    3717, 4558
\__enumext_keyans_addto_seq:n . 96, 3342, 3342,
    3719, 4560
\__enumext_keyans_addto_seq_link: 3342, 3360,
    3362, 5357
\__enumext_keyans_default_item:n . 103, 3699,
    3699, 3736
\l__enumext_keyans_env_bool 22, 3953, 3966, 4129,
    4219
\__enumext_keyans_fake_item_indent: . 1064,
    1080, 3856
\l__enumext_keyans_level_h_int . 134, 16, 798,
    822, 3082, 3210, 3320, 4976, 5243, 5244
\l__enumext_keyans_level_int . 16, 1667, 3078,
    3206, 3315, 3460, 4128, 4133, 4521
\__enumext_keyans_make_label: 104, 3740, 3740,
    3854
\__enumext_keyans_make_label_box: 3740, 3744,
    3749, 3786
\__enumext_keyans_make_label_std: 3740, 3752,
    3774
\__enumext_keyans_mini_right_cmd:n 65, 1669,
    1702, 1702
\__enumext_keyans_mini_set_vskip: . . . . . 61
\__enumext_keyans_minipage_add_space: 1498,
    1524, 4158
\__enumext_keyans_minipage_set_skip: . 1498,
    1498, 1526
\__enumext_keyans_multi_addvspace: 1298, 1309,
    4182
\__enumext_keyans_multi_set_vskip: 57, 1298,
    1298, 1311
\__enumext_keyans_multicols_start: 4146, 4161,
    4163
\__enumext_keyans_multicols_stop: 1706, 4146,
    4188, 4217
\__enumext_keyans_name_and_start: 31, 37, 134,
    298, 298, 4130, 4409, 5248
\__enumext_keyans_parse_keys:n 4142, 4142, 4241
\__enumext_keyans_pic_arg_two: 118, 4402, 4425,
    4456
\l__enumext_keyans_pic_level_int . 16, 1648,
    3086, 3214, 3272, 3310, 3345, 4404, 4405
\__enumext_keyans_pic_parse_keys:n 4402, 4411,
    4455
\__enumext_keyans_pic_safe_exec: . 118, 4402,
    4402, 4454
\__enumext_keyans_pic_skip_abs:N . 118, 4402,
    4418, 4429
\__enumext_keyans_pos_mark_set: 98, 3416, 3416,
    3453, 3485
\__enumext_keyans_pre_itemsep_skip: . 1498,
    1517, 1544
\__enumext_keyans_redefine_item: . 104, 3722,
    3722, 3853
\__enumext_keyans_ref: . . . . . 48, 845, 860, 3855
\__enumext_keyans_ref:n . . . . . 48, 842, 845, 845
\__enumext_keyans_safe_exec: . 4122, 4122, 4240
\__enumext_keyans_save_item_opt:n . 97, 104,
    3383, 3383, 3713, 4557
\__enumext_keyans_set_item_width: 114, 4223,
    4223, 4250
\__enumext_keyans_show_ans: 98, 3416, 3445, 3779,
    3794, 4562
\__enumext_keyans_show_item_opt: 97, 104, 3383,
    3390, 3716, 4570
\__enumext_keyans_show_item_opt_viii: . 97,
    3383, 3403, 5450
\__enumext_keyans_show_pos: 98, 3416, 3458, 3780,
    3795, 4563
\__enumext_keyans_starred_item:n . 104, 3711,
    3711, 3731
\__enumext_keyans_starred_item_star: . 136,
    5323, 5351, 5419
\__enumext_keyans_start_counter: . 4234, 4234,
    4249
\__enumext_keyans_store_ref: . 95, 3289, 3289,
    3718, 4559, 5355
\__enumext_keyans_store_ref_aux_i: 96, 3289,
    3301, 3304
\__enumext_keyans_store_ref_aux_ii: 96, 3289,
    3330, 3332
\__enumext_keyans_unknown_keys:n . 3627, 3633,
    3638, 4399
\__enumext_keyans_unknown_keys:nn 3627, 3640,
    3642
\__enumext_keyans_wraper_label:n . . . . . 105
\__enumext_keyans_wraper_label_viii:n 5388,
    5388, 5424
\__enumext_keyans_wrapper_item_v:n 3763, 3766
\__enumext_keyans_wrapper_item_viii:n 5395,
    5399
\__enumext_keyans_wrapper_label:n 3740, 3756,
    3782, 3797, 4567
\__enumext_keyans_wrapper_opt_v:n . . . . 3398
\__enumext_keyans_wrapper_opt_viii:n . 3411
\l__enumext_label_copy_i_tl . 2879, 3308, 3313,
    3318, 3323
\l__enumext_label_copy_v_tl . . . . . 3318
\l__enumext_label_copy_vi_tl . . . . . 3313
\l__enumext_label_copy_vii_tl 2855, 2866, 2895,
    3308
\l__enumext_label_copy_viii_tl . . . . . 3323
\l__enumext_label_copy_X_tl . . . . . 162
\l__enumext_label_fill_left_v_tl . . . . . 3778
\l__enumext_label_fill_left_X_tl . . . . . 90
\l__enumext_label_fill_right_v_tl . . . . 3783
\l__enumext_label_fill_right_X_tl . . . . . 90
\l__enumext_label_font_style_v_tl 3781, 3796,
    4566, 4574
\l__enumext_label_font_style_vii_tl . . 5144
\l__enumext_label_font_style_viii_tl . 5423
\l__enumext_label_i_tl . . . . . 717
\l__enumext_label_ii_tl . . . . . 717
\l__enumext_label_iii_tl . . . . . 717
\l__enumext_label_iv_tl . . . . . 717
\__enumext_label_style:Nnn 31, 44, 618, 618, 633,
    722, 768, 833, 835
\l__enumext_label_v_tl 96, 830, 3277, 3350, 3419,
    4244, 4433
\l__enumext_label_vi_tl 96, 830, 3274, 3347, 4567,
    4575
\l__enumext_label_vii_tl . 763, 5075, 5098, 5105

```


`\l__enumext_label_viii_tl` [763](#), [5320](#), [5349](#), [5353](#)
`\l__enumext_label_width_by_box` .. [57](#), [614](#), [615](#)
`__enumext_label_width_by_box:Nn` [44](#), [612](#), [612](#),
[617](#), [629](#), [894](#), [3418](#)
`\l__enumext_labelsep_v_dim` ... [3439](#), [4172](#), [4445](#),
[4569](#)
`\l__enumext_labelsep_vii_dim` . [2981](#), [4664](#), [4674](#),
[4758](#), [5040](#), [5096](#), [5151](#), [5160](#)
`\l__enumext_labelsep_viii_dim` [4695](#), [4705](#), [4807](#),
[5284](#), [5361](#), [5426](#), [5435](#)
`\l__enumext_labelwidth_v_dim` . [838](#), [3429](#), [3434](#),
[3455](#), [3487](#), [3792](#), [4172](#), [4445](#), [4564](#)
`\l__enumext_labelwidth_vii_dim` ... [2984](#), [4664](#),
[4673](#), [4758](#), [5040](#), [5142](#), [5159](#)
`\l__enumext_labelwidth_viii_dim` .. [4695](#), [4704](#),
[4807](#), [5284](#), [5368](#), [5385](#), [5421](#), [5434](#)
`\l__enumext_leftmargin_tmp_v_bool` . [118](#), [4427](#)
`\l__enumext_leftmargin_tmp_X_bool` [61](#)
`\l__enumext_leftmargin_tmp_X_dim` [61](#)
`\l__enumext_leftmargin_X_dim` [61](#)
`__enumext_level:` [217](#), [217](#), [747](#), [749](#), [758](#), [760](#), [1067](#),
[1071](#), [1075](#), [1143](#), [1147](#), [1151](#), [1155](#), [1238](#), [1240](#), [1242](#),
[1244](#), [1286](#), [1288](#), [1290](#), [1292](#), [1296](#), [1330](#), [1336](#), [1341](#),
[1343](#), [1346](#), [1349](#), [1362](#), [1365](#), [1676](#), [1680](#), [1686](#), [1749](#),
[1751](#), [1753](#), [1756](#), [1763](#), [1765](#), [1767](#), [1770](#), [1871](#), [1929](#),
[1963](#), [1965](#), [1967](#), [1969](#), [1989](#), [1991](#), [2022](#), [2031](#), [2036](#),
[2039](#), [2087](#), [2091](#), [2163](#), [2672](#), [2674](#), [2676](#), [2704](#), [2705](#),
[2707](#), [2764](#), [2772](#), [2776](#), [2780](#), [2991](#), [2995](#), [3499](#), [3500](#),
[3504](#), [3505](#), [3506](#), [3514](#), [3522](#), [3523](#), [3526](#), [3533](#), [3534](#),
[3538](#), [3541](#), [3543](#), [3577](#), [3579](#), [3580](#), [3582](#), [3585](#), [3596](#),
[3597](#), [3600](#), [3601](#), [3603](#), [3959](#), [3972](#), [3979](#), [3987](#), [3990](#),
[3992](#), [3994](#), [3995](#), [3996](#), [3997](#), [4000](#), [4005](#), [4011](#), [4017](#),
[4024](#), [4037](#), [4039](#), [4042](#), [4043](#), [4045](#), [4049](#), [4055](#), [4083](#),
[4088](#), [4094](#), [4096](#), [4106](#), [4108](#)
`\l__enumext_level_h_int` [128](#), [16](#), [250](#), [273](#), [286](#), [784](#),
[815](#), [1655](#), [1943](#), [1972](#), [1993](#), [2100](#), [2138](#), [2166](#), [2202](#),
[2237](#), [2443](#), [2463](#), [2874](#), [3967](#), [4971](#), [4972](#)
`\l__enumext_level_int` . [108](#), [16](#), [219](#), [260](#), [272](#), [287](#),
[559](#), [1250](#), [1375](#), [1654](#), [1927](#), [1961](#), [1987](#), [2004](#), [2085](#),
[2090](#), [2122](#), [2132](#), [2135](#), [2159](#), [2186](#), [2196](#), [2199](#), [2229](#),
[2233](#), [2235](#), [2320](#), [2322](#), [2324](#), [2337](#), [2339](#), [2437](#), [2469](#),
[2851](#), [2861](#), [2867](#), [2873](#), [2880](#), [2889](#), [2894](#), [3252](#), [3678](#),
[3870](#), [3916](#), [3917](#), [3928](#), [3939](#), [3957](#), [3970](#), [4001](#), [4137](#),
[4517](#), [5020](#), [5030](#), [5256](#), [6167](#), [6171](#), [6177](#), [6181](#), [6258](#)
`__enumext_list_arg_two_i:` [3834](#)
`__enumext_list_arg_two_ii:` [3834](#)
`__enumext_list_arg_two_iii:` [3834](#)
`__enumext_list_arg_two_iv:` [3834](#)
`__enumext_list_arg_two_v:` [104](#), [3834](#), [4246](#), [4428](#)
`__enumext_list_arg_two_vii:` [3876](#), [4950](#)
`__enumext_list_arg_two_viii:` [3876](#), [5219](#)
`\l__enumext_listoffset_v_dim` . [4174](#), [4228](#), [4231](#)
`\l__enumext_listparindent_vii_dim` [5167](#), [5171](#)
`\l__enumext_listparindent_viii_dim` [5442](#), [5446](#)
`__enumext_log_answer_vars:` . [38](#), [352](#), [360](#), [3259](#)
`__enumext_log_global_vars:` . [38](#), [352](#), [352](#), [3258](#)
`__enumext_make_label:` ... [100](#), [3557](#), [3557](#), [3864](#)
`__enumext_make_label_box:` ... [3557](#), [3561](#), [3566](#),
[3589](#)
`__enumext_make_label_std:` ... [3557](#), [3569](#), [3573](#)
`\l__enumext_mark_answer_sym_tl` [86](#), [2616](#), [2830](#),
[3007](#), [3441](#), [5365](#), [5372](#)
`\l__enumext_mark_answer_sym_v_tl` . [3441](#), [3473](#)
`\l__enumext_mark_answer_sym_viii_tl` ... [5365](#)
`\l__enumext_mark_position_str` [123](#), [2622](#), [2623](#),
[2624](#), [2828](#), [3443](#), [5366](#), [5383](#)
`\l__enumext_mark_position_v_str` .. [123](#), [3443](#)
`\l__enumext_mark_position_viii_str` [123](#), [5366](#),
[5383](#)
`\l__enumext_mark_ref_sym_tl` .. [2604](#), [2969](#), [3374](#)
`\l__enumext_mark_sep_tmpa_dim` [123](#), [3419](#), [3429](#),
[3434](#)
`\l__enumext_mark_sep_tmpe_dim` [123](#), [3424](#), [3428](#),
[3433](#), [3442](#)
`\l__enumext_mark_sym_sep_dim` . [2619](#), [2979](#), [2981](#),
[2984](#), [2987](#), [2989](#)
`\l__enumext_mark_sym_sep_v_dim` ... [3437](#), [3439](#),
[3442](#), [3455](#), [3487](#)
`\l__enumext_mark_sym_sep_viii_dim` [5359](#), [5361](#),
[5368](#), [5385](#)
`\l__enumext_meta_path_tl` . [119](#), [5739](#), [5740](#), [5742](#),
[5743](#)
`\c__enumext_meta_paths_prop` [143](#), [5715](#)
`__enumext_mini_addvspace_vii:` [63](#), [1634](#), [1634](#),
[4832](#)
`__enumext_mini_addvspace_viii:` [63](#), [1634](#), [1640](#),
[4897](#)
`__enumext_mini_env*` [557](#)
`__enumext_mini_page` [1686](#), [1713](#), [4049](#), [4159](#), [4834](#),
[4899](#), [4920](#)
`__enumext_mini_right_cmd:n` . [64](#), [65](#), [1671](#), [1673](#),
[1673](#)
`__enumext_mini_set_vskip_vii:` [62](#), [1577](#), [1577](#),
[1636](#)
`__enumext_mini_set_vskip_viii:` [62](#), [1577](#), [1599](#),
[1642](#)
`__enumext_minipage:w` [39](#), [367](#), [375](#), [564](#), [4857](#), [5166](#),
[5441](#)
`\l__enumext_minipage_active_v_bool` [4156](#), [4179](#),
[4204](#)
`\g__enumext_minipage_active_vii_bool` .. [125](#),
[4846](#), [4855](#), [4877](#)
`\l__enumext_minipage_active_vii_bool` . [4828](#),
[4839](#)
`\g__enumext_minipage_active_viii_bool` [4910](#),
[4918](#), [4937](#)
`\l__enumext_minipage_active_viii_bool` [4893](#),
[4904](#)
`\g__enumext_minipage_active_X_bool` ... [176](#)
`\l__enumext_minipage_active_X_bool` [77](#)
`__enumext_minipage_add_space:` . [59](#), [111](#), [1326](#),
[1352](#), [4047](#)
`\g__enumext_minipage_after_skip` [77](#), [1581](#), [1593](#),
[4875](#), [4935](#)
`\l__enumext_minipage_after_skip` .. [58](#), [111](#), [77](#),
[1339](#), [1379](#), [1381](#), [1386](#), [1389](#), [1393](#), [1398](#), [1402](#), [1405](#),
[1409](#), [1421](#), [1426](#), [1429](#), [1433](#), [1438](#), [1442](#), [1445](#), [1449](#),
[1460](#), [1465](#), [1468](#), [1472](#), [1477](#), [1481](#), [1484](#), [1488](#), [1500](#),
[1514](#), [1547](#), [1549](#), [1554](#), [1556](#), [1558](#), [1562](#), [1566](#), [1568](#),
[1570](#), [1601](#), [1614](#), [1628](#), [1682](#), [1709](#), [4214](#)
`\g__enumext_minipage_center_vii_bool` . [4861](#),
[4878](#)
`\g__enumext_minipage_center_viii_bool` [4922](#),
[4938](#)
`\g__enumext_minipage_center_X_bool` ... [176](#)
`\l__enumext_minipage_hsep_v_dim` [4154](#)
`\l__enumext_minipage_hsep_vii_dim` [4826](#)
`\l__enumext_minipage_hsep_viii_dim` ... [4891](#)
`\l__enumext_minipage_left_skip` [77](#), [1501](#), [1579](#),

[1584](#), [1588](#), [1602](#), [1606](#), [1620](#), [1638](#), [1644](#)
`\l__enumext_minipage_left_v_dim` .. [4152](#), [4159](#)
`\l__enumext_minipage_left_vii_dim` [4822](#), [4834](#)
`\l__enumext_minipage_left_viii_dim` [4887](#), [4899](#)
`\l__enumext_minipage_left_X_dim` [77](#)
`\g__enumext_minipage_right_skip` [77](#), [1580](#), [1585](#),
[1589](#), [4860](#), [4921](#)
`\l__enumext_minipage_right_skip` . [58](#), [77](#), [1328](#),
[1334](#), [1339](#), [1341](#), [1343](#), [1502](#), [1503](#), [1509](#), [1514](#), [1515](#),
[1516](#), [1521](#), [1603](#), [1610](#), [1624](#), [1688](#), [1715](#)
`\l__enumext_minipage_right_v_dim` . [1704](#), [1713](#),
[4150](#), [4154](#)
`\g__enumext_minipage_right_vii_dim` [125](#), [4830](#),
[4857](#), [4880](#)
`\l__enumext_minipage_right_vii_dim` [125](#), [4820](#),
[4825](#), [4831](#)
`\g__enumext_minipage_right_viii_dim` .. [4895](#),
[4920](#), [4940](#)
`\l__enumext_minipage_right_viii_dim` .. [4885](#),
[4890](#), [4896](#)
`\g__enumext_minipage_right_X_dim` [176](#)
`\g__enumext_minipage_right_X_skip` [176](#)
`__enumext_minipage_set_skip`: . [58](#), [1326](#), [1326](#),
[1354](#)
`\g__enumext_minipage_stat_int` .. [111](#), [77](#), [1693](#),
[1720](#), [4046](#), [4057](#), [4062](#), [4157](#), [4206](#), [4211](#)
`\l__enumext_minipage_temp_skip` [77](#), [1400](#), [1410](#),
[1413](#), [1440](#), [1450](#), [1453](#), [1479](#), [1489](#), [1492](#), [1564](#), [1571](#),
[1573](#)
`\l__enumext_miniright_code_vii_box` [4868](#), [4872](#)
`\g__enumext_miniright_code_vii_tl` [126](#), [4863](#),
[4870](#), [4879](#)
`\l__enumext_miniright_code_viii_box` .. [4929](#),
[4933](#)
`\g__enumext_miniright_code_viii_tl` [4924](#), [4931](#),
[4939](#)
`\l__enumext_miniright_code_X_box` [176](#)
`\l__enumext_mode_box_bool` [638](#), [3564](#), [3747](#)
`__enumext_multi_addvspace`: [57](#), [110](#), [1281](#), [1281](#),
[4008](#)
`__enumext_multi_set_vskip`: [56](#), [1236](#), [1236](#), [1283](#)
`\l__enumext_multicols_above_ii_skip` ... [1255](#)
`\l__enumext_multicols_above_iii_skip` .. [1264](#)
`\l__enumext_multicols_above_iv_skip` ... [1273](#)
`\l__enumext_multicols_above_v_skip` [1300](#), [1314](#),
[1324](#), [1515](#)
`\l__enumext_multicols_above_X_skip` [69](#)
`\l__enumext_multicols_below_ii_skip` .. [1382](#),
[1391](#), [1395](#), [1407](#), [1412](#)
`\l__enumext_multicols_below_iii_skip` . [1422](#),
[1431](#), [1435](#), [1447](#), [1452](#)
`\l__enumext_multicols_below_iv_skip` .. [1461](#),
[1470](#), [1474](#), [1486](#), [1491](#)
`\l__enumext_multicols_below_v_skip` [1304](#), [1318](#),
[1516](#), [1550](#), [1557](#), [1559](#), [1569](#), [1572](#), [4196](#)
`\l__enumext_multicols_below_X_skip` [69](#)
`\g__enumext_multicols_right_X_skip` [69](#)
`__enumext_multicols_start`: [109](#), [111](#), [3984](#), [3984](#),
[4051](#)
`__enumext_multicols_stop`: [110](#), [1678](#), [4014](#), [4014](#),
[4067](#)
`__enumext_nested_base_line_fix`: [51](#), [108](#), [984](#),
[990](#), [3935](#)
`__enumext_newlabel:nn` [34](#), [40](#), [88](#), [412](#), [412](#), [2905](#),
[3336](#)
`\l__enumext_newlabel_arg_one_tl` [34](#), [40](#), [87](#), [96](#),
[162](#), [2898](#), [2906](#), [2968](#), [3325](#), [3337](#), [3372](#)
`\l__enumext_newlabel_arg_two_tl` [34](#), [40](#), [86](#), [162](#),
[2854](#), [2864](#), [2877](#), [2892](#), [2907](#), [3312](#), [3317](#), [3322](#), [3338](#)
`__enumext_parse_foreach_keys:n` .. [5764](#), [5780](#),
[5797](#)
`__enumext_parse_foreach_keys:nn` . [5764](#), [5787](#),
[5799](#)
`__enumext_parse_keys:n` [51](#), [69](#), [3923](#), [3923](#), [4102](#)
`__enumext_parse_keys_vii:n` [69](#), [4945](#), [4984](#), [4984](#)
`__enumext_parse_keys_viii:n` . [5215](#), [5261](#), [5261](#)
`__enumext_parse_save_key:n` [84](#), [2697](#), [2702](#), [2702](#)
`__enumext_parse_save_key_vii:n` [84](#), [2692](#), [2702](#),
[2710](#)
`__enumext_parse_series:n` [68](#), [69](#), [108](#), [128](#), [1923](#),
[1923](#), [3933](#), [3942](#), [4992](#)
`__enumext_parse_store_keys:n` [108](#)
`\l__enumext_parsep_i_skip` [1253](#), [1257](#)
`\l__enumext_parsep_ii_skip` [1262](#), [1266](#)
`\l__enumext_parsep_iii_skip` [1271](#), [1275](#)
`\l__enumext_parsep_vii_skip` [5168](#)
`\l__enumext_parsep_viii_skip` [5443](#)
`\l__enumext_partopsep_v_skip` . [1316](#), [1320](#), [1511](#),
[1534](#)
`\l__enumext_partopsep_viii_skip` [1612](#)
`__enumext_phantomsection`: [40](#), [378](#), [405](#), [409](#), [425](#)
`__enumext_pre_itemsep_skip`: .. [59](#), [1344](#), [1373](#),
[1373](#)
`__enumext_print_footnote`: .. [427](#), [449](#), [513](#), [518](#)
`__enumext_print_footnote_mini`: [427](#), [479](#), [540](#),
[545](#)
`__enumext_print_footnote_standar`: [491](#), [507](#),
[571](#)
`__enumext_print_footnote_starred`: [491](#), [536](#),
[551](#), [555](#)
`__enumext_print_keyans_box:NN` [86](#), [2822](#), [2822](#),
[2835](#), [2983](#), [2994](#), [3454](#), [3486](#), [5367](#), [5384](#)
`\l__enumext_print_keyans_cmd_bool` [123](#), [1847](#),
[1861](#), [1882](#), [3931](#), [3940](#), [4073](#), [4990](#), [5011](#), [5558](#), [5565](#)
`\l__enumext_print_keyans_i_tl` ... [5526](#), [5559](#)
`\l__enumext_print_keyans_ii_tl` ... [5532](#), [5560](#)
`\l__enumext_print_keyans_iii_tl` .. [5538](#), [5561](#)
`\l__enumext_print_keyans_iv_tl` ... [5544](#), [5562](#)
`\l__enumext_print_keyans_star_bool` [51](#), [52](#), [140](#),
[123](#), [996](#), [1008](#), [5583](#), [5588](#)
`\l__enumext_print_keyans_starred_tl` [139](#), [140](#),
[123](#), [5520](#), [5581](#)
`\l__enumext_print_keyans_vii_tl` [139](#), [5550](#), [5563](#)
`\l__enumext_print_keyans_X_tl` [123](#)
`__enumext_printkeyans:nnn` [140](#), [5555](#), [5564](#), [5568](#)
`__enumext_redefine_item`: [100](#), [3546](#), [3546](#), [3863](#)
`\l__enumext_ref_key_arg_t` [46](#)
`\l__enumext_ref_key_arg_tl` [37](#), [739](#), [740](#), [752](#), [783](#),
[786](#), [794](#), [800](#), [808](#), [847](#), [848](#), [856](#)
`\l__enumext_ref_the_count_tl` . [46](#), [37](#), [745](#), [751](#),
[791](#), [794](#), [805](#), [808](#), [853](#), [856](#)
`__enumext_register_default_label_wd:Nn` [602](#),
[602](#), [607](#), [608](#), [609](#), [610](#), [611](#)
`__enumext_remove_extra_parsep_vii`: .. [4964](#),
[5192](#), [5192](#)
`__enumext_remove_extra_parsep_viii`: . [5233](#),
[5471](#), [5471](#)
`\l__enumext_renew_counter_v_tl` . [854](#), [862](#), [864](#)
`\l__enumext_renew_counter_vii_tl` [792](#), [817](#), [819](#)

`\l__enumext_renew_counter_viii_tl` . 806, 824, 826
`\l__enumext_renew_counter_X_tl` 37
`__enumext_renew_footnote:` . . 427, 431, 497, 502
`__enumext_renew_footnote_mini:` 427, 461, 527, 532
`__enumext_renew_footnote_standar:` 491, 491, 563
`__enumext_renew_footnote_starred:` 491, 523, 5162, 5437
`__enumext_reset_count_resume:nn` . 2246, 2274, 2278, 2280, 2285, 2290, 2297
`__enumext_reset_count_resume_all:n` . . 2246, 2250, 2286
`__enumext_reset_count_resume_levels:n` 2246, 2255, 2293
`__enumext_reset_global_bool:` . . 328, 331, 340
`__enumext_reset_global_int:` . . . 328, 330, 334
`__enumext_reset_global_tl:` 328, 332, 346
`__enumext_reset_global_vars:` . 38, 94, 328, 328, 3266
`__enumext_resume:n` 72, 1872, 2083, 2083
`\l__enumext_resume_count_bool` . . 47, 869, 1867, 1888, 1934, 1950
`__enumext_resume_counter:` . . 72, 73, 1939, 1955, 2113, 2113, 2227, 2241
`__enumext_resume_counter_series:` . 73, 2113, 2164, 2171, 2174
`__enumext_resume_save_counter:` . . . 111, 129
`__enumext_resume_series:n` 69, 72, 73, 2089, 2104, 2113, 2157
`\l__enumext_resume_series_vii_bool` . . . 1891
`__enumext_resume_star:` . . . 74, 1849, 2221, 2221
`__enumext_resume_vii:n` . . . 72, 1892, 2083, 2098
`\l__enumext_resume_vii_bool` 1945, 2170
`\l__enumext_resume_X_bool` 46
`\l__enumext_rightmargin_vii_dim` . . 4676, 4680, 4685
`\l__enumext_rightmargin_viii_dim` . 4707, 4711, 4716
`__enumext_safe_exec:` . . 42, 108, 3912, 3912, 4101
`__enumext_safe_exec_vii:` . 42, 4944, 4967, 4967
`__enumext_safe_exec_viii:` 134, 5214, 5237, 5237
`__enumext_save_last_keys:n` . 69, 70, 1923, 1931, 1947, 1985
`\g__enumext_save_last_keys_vii_tl` 1995, 1996, 2239, 2242
`\g__enumext_save_last_keys_X_tl` 46
`__enumext_scan_tokens:n` . . . 94, 204, 204, 3247
`__enumext_second_part:` . . 111, 4053, 4053, 4116
`__enumext_second_part_v:` . . . 4146, 4202, 4255
`\l__enumext_series_name_str` . 70, 108, 128, 1844, 1925, 1963, 1965, 1967, 1969, 1974, 1976, 1978, 1980, 2019, 2022, 2026, 2060, 2063, 2067, 3927, 4988
`\l__enumext_series_name_tl` 70, 73, 46, 1863, 1864, 1884, 1885, 1936, 1952, 2024, 2036, 2039, 2065, 2076, 2079, 2161, 2162, 2168, 2169, 2178, 2182, 2216
`\g__enumext_series_name_X_tl` 46
`__enumext_set_error:nn` 5674, 5711, 5713
`__enumext_set_item_width:` 111, 4078, 4078, 4112
`__enumext_set_parse:n` 5674, 5685, 5701
`\l__enumext_setkey_tmpa_int` . . . 114, 5678, 5682
`\l__enumext_setkey_tmpa_seq` . . 114, 5676, 5686, 5692, 5694, 5696, 5708
`\l__enumext_setkey_tmpa_tl` 114, 5684, 5688
`\l__enumext_setkey_tmpb_seq` . . 114, 5677, 5680, 5684, 5685
`\l__enumext_setkey_tmpb_tl` 114, 5703, 5705, 5706
`\l__enumext_show_answer_bool` . 2591, 2610, 3001, 3395, 3408, 3449, 3761, 5363, 5393
`__enumext_show_length:nnn` . . 53, 227, 227, 5926, 5927, 5928, 5929, 5930, 5931, 5932, 5933, 5934, 5935, 5941, 5942, 5943, 5944, 5945, 5946, 5947, 5948, 5949, 5950
`\l__enumext_show_pos_tmp_int` . 123, 3462, 3465, 3480
`\l__enumext_show_position_bool` . . . 2594, 2613, 3005, 3396, 3409, 3469, 5370
`\g__enumext_standar_bool` 36, 108, 22, 249, 252, 271, 343, 493, 509, 2001, 2468, 2482, 2859, 2872, 2887, 3954
`\l__enumext_standar_bool` 108, 111, 22, 1662, 2860, 3919, 4072, 4981
`\l__enumext_standar_first_bool` 37, 108, 22, 276, 2125, 2189, 2371, 2378
`__enumext_standar_item_vii:w` . 130, 5044, 5062, 5064
`__enumext_standar_item_viii:w` 135, 5288, 5306, 5308
`__enumext_standar_ref:` 47, 737, 756, 3865
`__enumext_standar_ref:n` 729, 737, 737
`__enumext_standar_save_counter:` . . 70, 1999, 1999, 4075
`__enumext_standar_save_counter_aux:` . 1999, 2003, 2014, 2017
`__enumext_standar_unknown_keys:n` 3673, 3680, 3685
`__enumext_standar_unknown_keys:nn` 3673, 3687, 3689
`__enumext_standard_ref:n` 46
`__enumext_standard_reset:nn` . 2246, 2264, 2272
`__enumext_standard_reset_key:` 76, 2304, 2318, 2318
`__enumext_standard_reset_key_star:` 76, 2306, 2318, 2328
`\g__enumext_starred_bool` 36, 128, 22, 259, 262, 285, 344, 1661, 2045, 2442, 2488, 2850, 3306, 4881
`\l__enumext_starred_bool` 128, 129, 134, 22, 2888, 2923, 2929, 2977, 3920, 4980, 5010, 5249, 5253
`__enumext_starred_columns_set_vii:` . . 4658, 4658, 4955
`__enumext_starred_columns_set_viii:` . 4658, 4689, 5224
`\l__enumext_starred_first_bool` 37, 128, 22, 290, 994, 1007, 2141, 2205, 2371, 2378
`__enumext_starred_item_vii:w` . 130, 5044, 5061, 5078
`__enumext_starred_item_vii_aux_i:w` . . 5044, 5083, 5086
`__enumext_starred_item_vii_aux_ii:w` . 5044, 5084, 5089, 5091
`__enumext_starred_item_vii_aux_iii:w` 5044, 5094, 5101
`__enumext_starred_item_viii:w` 135, 136, 5305, 5323, 5323
`__enumext_starred_item_viii_aux_i:w` . . 136, 5323, 5329, 5332
`__enumext_starred_item_viii_aux_ii:w` . 136, 5323, 5330, 5344, 5346

__enumext_starred_joined_item_vii:n 124, 130, 4720, 4720, 5059
 __enumext_starred_joined_item_viii:n . 124, 135, 4720, 4769, 5303
 __enumext_starred_ref: 48, 781, 813, 3897
 __enumext_starred_ref:n 47, 775, 781, 781
 __enumext_starred_reset:n . . . 2246, 2259, 2276
 __enumext_starred_reset_key: . 76, 2313, 2315, 2318, 2342
 __enumext_starred_save_counter: . . 70, 1999, 2043, 5013
 __enumext_starred_save_counter_aux: . 1999, 2047, 2055, 2058
 __enumext_starred_unknown_keys:n 3652, 3656, 3659
 __enumext_starred_unknown_keys:nn 3652, 3661, 3663
 __enumext_start_counter: . . . 4092, 4092, 4111
 __enumext_start_from:NNn 49, 867, 867, 881, 903, 909
 \l__enumext_start_i_int 2128, 2192
 __enumext_start_item_tmp_vii: 127, 4958, 5044, 5044
 __enumext_start_item_tmp_viii: . . 5227, 5288, 5288
 __enumext_start_item_vii:w 130, 132, 5070, 5075, 5098, 5105, 5153, 5153
 __enumext_start_item_viii:w . . 135, 5315, 5320, 5349, 5428, 5428
 \g__enumext_start_line_tl 37, 22, 278, 292, 349, 2512, 2517, 2522, 2536, 2541, 2546
 __enumext_start_list:nn 39, 105, 367, 369, 4105, 4243, 4948, 5217
 __enumext_start_list_tag:n . . 4257, 4283, 5163, 5438
 __enumext_start_mini_vii: 128, 4818, 4818, 5002
 __enumext_start_mini_viii: . . 134, 4883, 4883, 5272
 __enumext_start_save_ans_msg: 77, 2355, 2355, 2380
 __enumext_start_store_level: . 109, 3948, 3948, 4104
 __enumext_start_store_level_vii: 129, 4947, 5016, 5016
 \l__enumext_start_vii_int 2144, 2150, 2208, 2214
 \l__enumext_start_X_int 90
 __enumext_stop_item_tmp_vii: 127, 129, 130, 132, 4957, 4963, 5046, 5155
 __enumext_stop_item_tmp_viii: 135, 5226, 5232, 5290, 5430
 __enumext_stop_item_vii: 132, 5153, 5155, 5175
 __enumext_stop_item_viii: . . 5428, 5430, 5454
 __enumext_stop_list: 39, 126, 129, 367, 370, 4019, 4027, 4192, 4199, 4841, 4849, 4906, 4913
 __enumext_stop_list_tag:n . . 4257, 4299, 5178, 5457
 __enumext_stop_mini_vii: 125, 129, 4818, 4837, 5006
 __enumext_stop_mini_viii: 134, 4883, 4902, 5276
 __enumext_stop_save_ans_msg: . 77, 2355, 2360, 3256
 __enumext_stop_start_list_tag: . . 4257, 4291, 5165, 5440
 __enumext_stop_store_level: . . 109, 110, 3977, 3977, 4020, 4028
 __enumext_stop_store_level_vii: . . 126, 129, 4842, 4850, 5016, 5026
 \l__enumext_store_active_bool 33, 77, 102, 2126, 2142, 2190, 2206, 2387, 3074, 3202, 3952, 3965, 4124, 4132, 4513, 5018, 5028, 5239, 5255
 __enumext_store_active_keys:n . 83, 108, 2670, 2670, 3945
 __enumext_store_active_keys_vii:n . 83, 128, 2670, 2680, 4994
 __enumext_store_addto_prop:n 85, 95, 2746, 2746, 2754, 2914, 3287, 5354
 __enumext_store_addto_seq:n 85, 97, 2755, 2755, 2759, 2766, 2780, 2788, 2797, 2811, 2819, 2972, 3377
 __enumext_store_anskey_arg:n . . 88, 90, 93, 94, 2911, 2911, 3067, 3245
 \l__enumext_store_anskey_arg_tl . . 33, 88, 107, 2920, 2925, 2927, 2932, 2939, 2942, 2952, 2957, 2960, 2966, 2972
 __enumext_store_anskey_env:n . 94, 3196, 3200, 3230
 \l__enumext_store_anskey_env_tl . . 33, 94, 107, 3232, 3234, 3236, 3239, 3247
 __enumext_store_anskey_safe_outer: . . 91, 93
 \l__enumext_store_columns_break_bool . 2922, 3023, 3109
 \l__enumext_store_current_label_tl 33, 95-97, 136, 102, 3271, 3274, 3277, 3283, 3285, 3287, 3344, 3347, 3350, 3356, 3358, 3368, 3377, 5334, 5339, 5340, 5353, 5354, 5356
 \l__enumext_store_current_opt_arg_tl . 33, 97, 136, 102, 3387, 3392, 3399, 3405, 3412, 5342
 __enumext_store_internal_ref: . . 86, 88, 2836, 2836, 2917
 \l__enumext_store_item_join_int . . 2930, 2934, 3026, 3112
 \l__enumext_store_item_star_bool . 2937, 3028, 3114
 \l__enumext_store_item_symbol_sep_dim 2949, 2954, 3033, 3119
 \l__enumext_store_item_symbol_tl . 2940, 2944, 3031, 3117
 \l__enumext_store_keyans_item_opt_sep_v_-tl 3281, 3283, 3354, 3356
 \l__enumext_store_keyans_item_opt_sep_-viii_tl 5337, 5339
 __enumext_store_level_close: . 85, 2760, 2784, 3981
 __enumext_store_level_close_vii: . 86, 2791, 2815, 5032
 __enumext_store_level_open: 85, 109, 2760, 2760, 3960, 3973
 __enumext_store_level_open_vii: . . 86, 2791, 2791, 5022
 \g__enumext_store_name_tl 33, 77, 102, 348, 355, 356, 357, 358, 2363, 2389, 2511, 2516, 2521, 2535, 2540, 2545, 3254
 \l__enumext_store_name_tl 33, 77, 78, 102, 2006, 2009, 2028, 2048, 2051, 2069, 2130, 2146, 2194, 2210, 2358, 2367, 2368, 2389, 2390, 2392, 2393, 2395, 2397, 2398, 2400, 2402, 2403, 2427, 2748, 2750, 2757, 2900, 2901, 3013, 3327, 3328, 3479, 5378
 \l__enumext_store_ref_key_bool 88, 2607, 2915, 2963, 3291, 3365
 \l__enumext_store_save_key_vii_bool . . 2682,

2712
 \l__enumext_store_save_key_vii_tl 2684, 2685,
 2713, 2714, 2795, 2803, 2807, 2811
 \l__enumext_store_save_key_X_bool .. 83, 123
 \l__enumext_store_save_key_X_tl 83, 123
 \l__enumext_store_upper_level_X_bool .. 123
 __enumext_storing_exec: .. 77, 2365, 2381, 2385
 __enumext_storing_set:n .. 77, 2350, 2365, 2365
 \l__enumext_the_counter_v_tl 853
 \l__enumext_the_counter_vii_tl 791
 \l__enumext_the_counter_viii_tl 805
 \l__enumext_the_counter_X_tl 37
 __enumext_tmp:n 32, 36, 39, 45, 48, 56, 61, 68, 69, 76,
 84, 89, 90, 101, 133, 140, 165, 169, 176, 196, 593, 601,
 634, 643, 1840, 1855, 1856, 1878, 2115, 2132, 2135,
 2176, 2196, 2199, 2223, 2235, 2300, 2310, 2330, 2339,
 2346, 2354, 2406, 2424, 2600, 2669, 2688, 2701, 2838,
 2845, 2846, 2867, 2880, 2883, 2894, 3293, 3300, 3627,
 3637, 3673, 3684, 3834, 3875, 3876, 3911
 __enumext_tmp:nn 644, 665, 666, 700, 701, 716, 898,
 923, 924, 939, 1020, 1042, 1043, 1063, 1117, 1125,
 1126, 1140, 1205, 1221, 1222, 1235, 1729, 1745, 2565,
 2599, 3611, 3626
 __enumext_tmp:nnn 717, 733, 734, 735, 736, 763, 779,
 780
 __enumext_tmp:nnnnn 940, 965, 968, 971, 973, 975,
 978, 981
 __enumext_tmp:w 5499, 5502
 \l__enumext_tmpa_vii_int 4668, 4671, 4680, 4711
 \l__enumext_tmpa_viii_int 4699, 4702
 \l__enumext_tmpa_X_dim 176
 \l__enumext_tmpa_X_int 176
 \l__enumext_topsep_v_skip 1302, 1306, 1505, 4506
 \l__enumext_topsep_vii_skip .. 1582, 1591, 1595
 \l__enumext_topsep_viii_skip . 1604, 1626, 1630
 __enumext_unskip_unkern: .. 36, 233, 233, 1355,
 1527, 4022, 4023, 4063, 4194, 4195, 4212, 5169, 5170,
 5444, 5445
 \l__enumext_vspace_a_star_v_bool 1778
 \l__enumext_vspace_a_star_vii_bool ... 1800
 \l__enumext_vspace_a_star_viii_bool ... 1811
 \l__enumext_vspace_a_star_X_bool 90
 __enumext_vspace_above: 65, 110, 1746, 1746, 4033
 __enumext_vspace_above_v: . 66, 1774, 1774, 4148
 \l__enumext_vspace_above_v_skip .. 1776, 1780,
 1782
 __enumext_vspace_above_vii: 66, 128, 1796, 1796,
 4999
 \l__enumext_vspace_above_vii_skip 1798, 1802,
 1804
 __enumext_vspace_above_viii: . 66, 1796, 1807,
 5270
 \l__enumext_vspace_above_viii_skip 1809, 1813,
 1815
 \l__enumext_vspace_b_star_v_bool 1789
 \l__enumext_vspace_b_star_vii_bool ... 1822
 \l__enumext_vspace_b_star_viii_bool ... 1833
 \l__enumext_vspace_b_star_X_bool 90
 __enumext_vspace_below: 66, 111, 1760, 1760, 4071
 __enumext_vspace_below_v: . 66, 1785, 1785, 4221
 \l__enumext_vspace_below_v_skip .. 1787, 1791,
 1793
 __enumext_vspace_below_vii: 67, 129, 1818, 1818,
 5009

\l__enumext_vspace_below_vii_skip 1820, 1824,
 1826
 __enumext_vspace_below_viii: . 67, 1818, 1829,
 5278
 \l__enumext_vspace_below_viii_skip 1831, 1835,
 1837
 __enumext_widest_from:nnn .. 49, 882, 882, 897,
 916
 \g__enumext_widest_label_tl 31, 44, 57, 622, 626,
 630
 \l__enumext_wrap_label_opt_v_bool 3707
 \l__enumext_wrap_label_opt_vii_bool 130, 5069
 \l__enumext_wrap_label_opt_viii_bool .. 135,
 5314
 \l__enumext_wrap_label_opt_X_bool 90
 \l__enumext_wrap_label_v_bool 3703, 3707, 3714,
 3760, 3768, 4556
 \l__enumext_wrap_label_vii_bool .. 130, 5069,
 5073, 5081, 5145
 \l__enumext_wrap_label_viii_bool . 135, 5314,
 5318, 5327, 5392, 5401
 \l__enumext_wrap_label_X_bool 90
 __enumext_wrapper_label_v:n . 3766, 3770, 4575
 __enumext_wrapper_label_vii:n 5147
 __enumext_wrapper_label_viii:n .. 5399, 5403
 \l__enumext_write_anskey_env_bool .. 33, 107,
 3125, 3150
 \l__enumext_write_anskey_env_file_iow .. 33,
 107, 3175, 3176, 3177
 \l__enumext_write_anskey_env_file_name_-
 tl 33, 107, 3126, 3236
 \l__enumext_write_aux_file_tl . 34, 88, 96, 162,
 2903, 2909, 3334, 3340
 enumext* 5, 4942
 enumXi 576
 enumXii 576
 enumXiii 576
 enumXiv 576
 enumXv 576
 enumXvi 576
 enumXvii 576
 enumXviii 576

Environments provide by **enumext**:

anskey* 30, 33, 35, 77, 82, 83, 87, 89, 92, 93, 109, 129, 138,
 140, 145, 148
 enumext* .. 30, 31, 35, 36, 40-44, 47-50, 52-56, 62, 63,
 66-70, 72, 74-79, 82-88, 90, 95, 96, 101, 102, 105,
 107-109, 115, 123, 124, 126, 129, 131-135, 137-141,
 143, 146, 149, 151
 enumext 30, 31, 35, 36, 40-58, 61, 64-70, 72, 74-79, 82-85,
 87, 88, 90, 95, 96, 99-101, 103, 105, 109, 112, 113, 118,
 122, 125, 128, 129, 131, 134, 139-141, 143, 146, 148, 150
 keyans* 30, 31, 33-37, 40-44, 47-50, 52-56, 62, 63, 66, 67,
 77, 78, 81, 82, 85, 93, 95, 97, 102, 105, 107, 115, 123,
 124, 133, 134, 146, 149, 151
 keyanspic .. 30, 31, 33, 34, 37, 43, 48, 77, 78, 81, 85, 93,
 95-97, 102, 115-121, 149
 keyans 30, 31, 33, 34, 36, 37, 40, 41, 43-45, 48-50, 52-55,
 57, 61, 64-66, 77, 78, 81, 82, 85, 93, 95-98, 102-105,
 112-114, 117, 118, 121, 125, 135, 146, 149

Environments:

center 123
 description 101, 122
 enumerate 122

flushleft	123
flushright	123
itemize	122
list	35, 39, 50, 90, 101, 105, 106, 110, 112, 115, 117–119, 122, 123, 126
lrbox	132
minipage	35, 39, 41, 42, 56, 58, 59, 117, 120, 122, 123, 126, 132
multicols	56–59, 64, 109–111
quotation	122
quote	122
tabbing	123
trivlist	123
verbatim	123
verse	122
exp commands:	
\exp_after:wN	5502
\exp_args:Ne	2090, 2117, 2180, 2228, 3244, 3938, 5490
\exp_args:NV	3039, 3136, 3640, 3661, 3687, 5799
\exp_not:N	43, 625, 751, 794, 808, 856, 1073, 1076, 1087, 1088, 1089, 1100, 1101, 1112, 1113, 2968, 3010, 3011, 3370, 3476, 3477, 5375, 5376, 5499
\exp_not:n	280, 294, 307, 315, 323, 691, 711, 751, 752, 794, 808, 856, 1074, 1912, 1921, 2578, 2627, 2731, 2744, 2906, 2934, 2944, 2954, 2968, 2969, 3337, 3372, 3374, 4370, 5617, 5625, 5831, 5836

F

\fbox	2634
\fboxrule	2634
\fboxsep	2634
file commands:	
\file_if_exist:nTF	3152
\file_input_stop:	6275
first	1126
font	644
\footnote	40
\footnote	40, 433, 463
\footnotemark	443, 473
\footnotesize	3011, 3477, 5376
\footnotetext	429
force-eol	3107
\foreachkeyans	19, 144, 5764

G

\getkeyans	19, 138, 5488
group commands:	
\group_begin:	3009, 3054, 3475, 5374, 5557
\group_end:	3016, 3070, 3483, 5381, 5566

H

\hbadness	5180, 5459
hbox commands:	
\hbox_overlap_left:n	2826, 3542, 5138
\hbox_set:Nn	614, 4433
\hbox_set_end:	5179, 5458
\hbox_set_to_wd:Nnw	5156, 5431
\hfill	674, 679, 685, 686, 1685, 1712, 2968, 3370, 4845, 4909
hook commands:	
\hook_gput_code:nnn	5, 207, 211, 215, 378
\hook_gset_rule:nnnn	379
\hyperlink	89, 97
\hyperlink	2968, 3370
\hypertarget	40
\hypertarget	404

I

\IfDocumentMetadataT	4285, 4293, 4301, 4337, 4345, 4353, 4457, 4466, 4474, 4481, 4486, 4534, 4543, 4633, 4641, 4843, 4907, 4954, 4962, 5114, 5223, 5231
\IfDocumentMetadataTF	495, 511, 525, 538, 3559, 3742
\IfHyperBoolean	386
\IfPackageLoadedT	382
\IfPackageLoadedTF	7, 394
\ignorespaces	1076, 1089, 1101, 1113, 4446, 4959, 5042, 5075, 5098, 5105, 5151, 5171, 5228, 5286, 5320, 5349, 5426, 5446
\inputlineno	280, 294, 307, 315, 323
int commands:	
\int_add:Nn	4753, 4802
\int_case:nn	1250, 1375, 2437, 2463, 2502, 2526
\int_case:nnTF	235
\int_compare:nNnTF	559, 784, 798, 815, 822, 1345, 1364, 1518, 1536, 1648, 1667, 1679, 1707, 1927, 1943, 1961, 1972, 1987, 1993, 2004, 2085, 2100, 2122, 2138, 2159, 2166, 2186, 2202, 2233, 2237, 2320, 2337, 2550, 2556, 3078, 3082, 3086, 3094, 3206, 3210, 3214, 3252, 3272, 3310, 3315, 3320, 3345, 3460, 3917, 3928, 3957, 3970, 3986, 4001, 4016, 4057, 4133, 4137, 4165, 4190, 4206, 4405, 4517, 4521, 4723, 4733, 4749, 4772, 4782, 4798, 4972, 4976, 5020, 5030, 5182, 5194, 5244, 5256, 5461, 5473, 5682, 5814
\int_compare_p:nNn	250, 260, 272, 273, 286, 287, 1654, 1655, 2262, 2263, 2443, 2469, 2851, 2861, 2873, 2874, 2889, 2930, 3967
\int_decr:N	4752, 4801
\int_eval:n	365, 911, 2266, 2750, 2901, 3011, 3328, 3477, 4096, 4236, 4741, 4790, 4953, 5222, 5376
\int_from_alph:n	876, 890
\int_from_roman:n	878, 892
\int_gadd:Nn	4754, 4803
\int_gdecr:N	2446, 2451, 2455, 2459, 2472
\int_gincr:N	2913, 3380, 3496, 3530, 3720, 4046, 4157, 4561, 5048, 5124, 5292, 5358
\int_gset:Nn	441, 471, 2495
\int_gset_eq:NN	438, 468, 2008, 2021, 2030, 2038, 2050, 2062, 2071, 2078
\int_gzero:N	336, 337, 338, 1693, 1720, 2324, 2334, 2344, 2562, 4062, 4211, 5205, 5485
\int_if_exist:NTF	1967, 1978, 2006, 2036, 2048, 2076, 2178, 2322, 2332, 2400
\int_incr:N	3093, 3462, 3916, 4128, 4404, 4971, 5047, 5243, 5291
\int_mod:nn	5196, 5475
\int_new:N	16, 17, 18, 19, 20, 21, 77, 94, 116, 131, 143, 144, 155, 156, 157, 159, 170, 171, 179, 180, 181, 182, 183, 1969, 1980, 2403
\int_set:Nn	872, 876, 878, 2117, 2128, 2144, 2150, 2180, 2192, 2208, 2214, 3678, 4627, 4628, 4668, 4699, 4722, 4728, 4744, 4771, 4777, 4793, 5180, 5459, 5678, 5816
\int_set_eq:NN	3849, 3893, 4751, 4800
\int_sign:n	2497
\int_step_function:nnN	2132, 2135, 2196, 2199, 2235, 2339, 2867, 2880, 2894
\int_step_function:nnnN	5820
\int_step_inline:nn	5730
\int_step_inline:nnn	4629
\int_to_roman:n	219, 2117, 2119, 2178, 2180, 2182, 2225, 2230, 2274, 2322, 2324, 2332, 2334, 2847, 2884
\int_use:N	358, 363, 364, 1346, 1365, 1680, 2090, 2119, 2130, 2146, 2152, 2182, 2194, 2210, 2216, 2229, 3870,

3939, 3987, 3996, 4011, 4017, 4096, 4236, 4726, 4727,
4739, 4775, 4776, 4788, 4953, 5222, 6167, 6171, 6177,
6181, 6258

\int_zero:N 3465, 5186, 5465

iow commands:

\iow_char:N 3233, 3234

\iow_close:N 3177

\iow_new:N 111

\iow_now:Nn 3176

\iow_open:Nn 3175

\item 99, 103, 129, 132, 135, 137, 371, 2768, 2774, 2799, 2805,
2927, 3347, 3350, 3548, 3724, 4461, 4462, 4956, 4958,
5225, 5227, 5356

\item* 5, 17, 81, 3722

item-join 3021, 3107

item-pos* 3021, 3107, 3611

item-star 3021, 3107

item-sym* 3021, 3107, 3611

\itemindent 106

\itemindent 106

itemindent 1020

\itemsep 4450

\itemwidth . 575, 2634, 4080, 4086, 4225, 4231, 4762, 4766,
4811, 4815

K

keyans 16, 4238

keyans* 16, 5212

keyanspic 17, 4452

Keys for \anskey provide by enumext:

break-col 88, 90

force-eol 92

item-join 88, 90

item-pos* 88, 90

item-star 88, 90

item-sym* 88, 90

overwrite 92

write-env 92

Keys for anskey* provide by enumext:

break-col 88, 90

force-eol 92

item-join 88, 90

item-pos* 88, 90

item-star 88, 90

item-sym* 88, 90

overwrite 92

write-env 92

Keys for environments provide by enumext:

above* 32, 51, 65, 66, 110, 128

above 32, 51, 65, 66, 110, 128, 134

after 54, 111, 129, 134

align 32, 45, 98-100, 104, 131, 145

base-fix 51, 68, 84, 108

before* 54, 110, 128, 134

before 54

below* 32, 65-67, 111, 129

below 32, 65-67, 111, 129, 134

check-ans 34, 35, 37, 76-81, 84, 94, 97, 111, 112, 129, 133,
147

columns-sep 55, 109, 132

columns 32, 55, 65, 109

first 54, 132

font 45, 100, 104, 120, 131

item-pos* 100, 101

item-sym* 33, 100, 101

itemindent 32, 52, 53, 99, 100, 103-105, 132

itemsep 50, 107, 132

label-pos 117, 118, 120, 121

label-sep 117

labelsep 45, 106, 131

labelwidth 44-49, 106, 131

label 31, 44, 46, 49, 118, 122

layout-sep 117

layout-sty 117, 122

layout-top 117

lisparindent 107

list-indent 32, 52, 118

list-offset 52, 111, 114

listparindent 52, 132

mark-ans* 81, 84, 98

mark-ans 82, 84, 89

mark-pos* 81, 84, 98

mark-pos 33, 82, 84, 145

mark-ref 82, 84, 86, 89

mark-sep* 81, 84, 98

mark-sep 33, 82, 84, 136

mini-env 32, 40-42, 55, 64, 65, 84, 110, 123, 125, 126, 128,
129, 134

mini-right* 32, 35, 56, 84, 126, 128, 129

mini-right 32, 35, 56, 63, 84, 126, 128, 129

mini-sep 32, 55, 84, 110

mode-box 44, 99-101, 104, 105

no-store 34, 76-78, 84, 90, 93, 99, 100

noitemsep 50

nosep 50

overwrite 33, 92

parindent 107

parsep 50, 107, 118, 132

partopsep 50

ref 31, 46-48, 105, 147

reset* 68, 75, 76, 84, 141

reset 68, 75, 76, 84, 141

resume* . . . 31, 43, 67-69, 74, 76, 77, 84, 111, 129, 141

resume 31, 38, 43, 49, 67-70, 72-74, 76, 77, 84, 111, 129,
141, 151

rightmargin 52, 123

save-ans 33, 38, 69, 70, 72, 73, 76-78, 80, 83-85, 90, 91,
93-96, 103, 112, 120, 131, 134-136, 138, 139, 141, 147

save-key 33, 69, 83, 84, 108, 128

save-pos 84

save-ref . . . 34, 40, 82, 84, 86, 88, 89, 95, 97, 104, 136

save-sep 81, 84, 95, 136

series 31, 67, 69, 70, 72-74, 84, 108, 111, 128, 129, 141

show-ans . . . 33, 81, 82, 84, 86, 88, 89, 97, 98, 120, 136

show-length 36, 53, 105, 146

show-pos 33, 81, 82, 86, 88, 89, 97, 120, 136

start* 32, 49, 69

start 32, 36, 49, 69

store-key 83

topsep 50, 51, 118

widest 31, 36, 49

wrap-ans* 34, 81, 84, 104, 105, 120

wrap-ans 43, 82, 84, 86, 89

wrap-label* . . . 32, 45, 99, 100, 103-105, 130, 131, 135

wrap-label 32, 45, 99, 100, 103-105, 118, 120, 130, 131,
135

wrap-opt 81, 84, 97, 104, 120

wrap-sep 89

write-env 33, 92, 93

keys commands:

<code>\keys_define:nn</code>	636, 646, 668, 703, 719, 765, 830, 900, 926, 942, 984, 1022, 1045, 1119, 1128, 1207, 1224, 1731, 1842, 1858, 1879, 2302, 2311, 2348, 2408, 2567, 2602, 2690, 2695, 3021, 3107, 3613, 3629, 3652, 3675, 4359, 5516, 5627, 5756, 5764
<code>\keys_if_exist_p:nn</code>	5752, 5753
<code>\l_keys_key_str</code>	90, 92, 3039, 3136, 3640, 3661, 3687, 5799, 5911
<code>\keys_precompile:nnN</code>	140, 202, 202, 5518, 5524, 5530, 5536, 5542, 5548, 5782
<code>\keys_set:nn</code>	660, 1001, 1013, 1230, 1736, 1741, 2090, 2105, 2228, 2242, 2638, 2639, 2643, 2644, 2648, 2649, 2653, 2654, 2658, 2659, 2663, 2664, 3059, 3188, 3930, 3938, 4144, 4377, 4379, 4381, 4383, 4385, 4387, 4389, 4391, 4393, 4395, 4415, 4989, 5265, 5630, 5636, 5642, 5648, 5654, 5659, 5660, 5661, 5662, 5663, 5664, 5665, 5666, 5698, 5808

keyval commands:

<code>\keyval_parse:NNn</code>	1901, 2720, 5606
--------------------------------	------------------

L

<code>label</code>	717, 763, 830
<code>label-pos</code>	4359
<code>label-sep</code>	4359

Labels provide by enumext:

<code>\Alph*</code>	44
<code>\Roman*</code>	44
<code>\alph*</code>	44
<code>\arabic*</code>	44
<code>\roman*</code>	44

<code>labelsep</code>	644
<code>\labelwidth</code>	44
<code>labelwidth</code>	644
<code>\lastnodetype</code>	235
<code>layout-sep</code>	4359
<code>layout-sty</code>	4359
<code>layout-top</code>	4359
<code>\leftmargin</code>	106
<code>\leftmargin</code>	106, 4445

legacy commands:

<code>\legacy_if:nTF</code>	5109, 5112, 5409, 5412
<code>\legacy_if_gset_false:n</code>	565, 4858
<code>\legacy_if_set_false:n</code>	5111, 5411
<code>\legacy_if_set_true:n</code>	5074, 5097, 5104, 5118, 5319, 5348

<code>\linewidth</code>	110
<code>\linewidth</code>	4041, 4080, 4154, 4225, 4626, 4671, 4702, 4824, 4889
<code>\list</code>	369
<code>list-indent</code>	1020
<code>list-offset</code>	1020
<code>\listparindent</code>	4448
<code>listparindent</code>	1020

M

<code>\makebox</code>	122
<code>\makebox</code>	2828, 3595, 3792, 4551, 4564, 5142, 5421
<code>\makelabel</code>	99, 100, 104, 122
<code>\makelabel</code>	99, 103, 3575, 3591, 3776, 3788
<code>mark-ans</code>	2600, 4359
<code>mark-ans*</code>	2565, 2600
<code>mark-pos</code>	2600, 4359
<code>mark-pos*</code>	2565, 2600
<code>mark-ref</code>	2600

<code>mark-sep</code>	2600, 4359
<code>mark-sep*</code>	2565, 2600
<code>midpenalty</code>	924
<code>mini-env</code>	1205
<code>mini-sep</code>	1205
<code>\minipage</code>	375
<code>\miniright</code>	12, 63, 1646, 1697, 1724, 4060, 4209

mode commands:

<code>\mode_if_math:TF</code>	3102, 3225
<code>\mode_if_vertical:TF</code>	1284, 1312, 1332, 1356, 1507, 1528
<code>\mode_leave_vertical:</code>	999, 1010, 1073, 1087, 2824, 3540, 5136

<code>mode-box</code>	634
-----------------------	-----

msg commands:

<code>\msg_error:nn</code>	1699, 1726, 3063, 3096, 3100, 3192, 3223, 4135, 4139, 4407, 4464, 4519, 4974, 5246, 5258, 5667, 5726
<code>\msg_error:nnn</code>	742, 788, 802, 850, 1650, 1657, 1664, 1695, 1722, 2094, 2109, 2266, 2373, 3045, 3104, 3142, 3204, 3208, 3212, 3216, 3227, 3646, 3667, 3693, 4978, 5251, 5504, 5513, 5599, 5714, 5745, 5754, 5791, 5812
<code>\msg_error:nnnn</code>	3048, 3076, 3080, 3084, 3088, 3145, 3649, 3670, 3696, 4126, 4515, 4523, 5241, 5578, 5794
<code>\msg_error:nnnnn</code>	690, 710, 2577, 2626, 4369
<code>\msg_fatal:nn</code>	3918
<code>\msg_fatal:nnn</code>	579, 596
<code>\msg_info:nnn</code>	9, 12, 384, 396
<code>\msg_line_context:</code>	5871, 5876, 5881, 5886, 5915, 5920, 5925, 5940, 5955, 5959, 5963, 5967, 5971, 5975, 5982, 5989, 5995, 6009, 6013, 6018, 6022, 6026, 6030, 6035, 6039, 6043, 6047, 6052, 6099, 6103, 6108, 6113, 6117, 6122, 6198, 6202, 6207, 6212, 6217, 6221, 6225, 6229, 6233, 6237, 6241, 6245, 6249, 6254, 6259
<code>\msg_log:nnn</code>	2392, 2397, 2402
<code>\msg_log:nnnnn</code>	362, 2535, 2540, 2545
<code>\msg_log:nnnnnn</code>	354
<code>\msg_new:nnn</code>	5839, 5843, 5847, 5851, 5856, 5869, 5873, 5878, 5883, 5888, 5897, 5905, 5909, 5913, 5918, 5923, 5938, 5953, 5957, 5961, 5965, 5969, 5973, 5977, 5986, 5992, 5998, 6002, 6006, 6011, 6016, 6020, 6024, 6028, 6033, 6037, 6041, 6045, 6050, 6085, 6089, 6093, 6097, 6101, 6106, 6111, 6115, 6120, 6196, 6200, 6205, 6210, 6215, 6219, 6223, 6227, 6231, 6235, 6239, 6243, 6247, 6251, 6256, 6264
<code>\msg_new:nnnn</code>	5860, 6055, 6064, 6073, 6079, 6124, 6134, 6144, 6154, 6164, 6174, 6184, 6190, 6261, 6266, 6269, 6272
<code>\msg_term:nnnn</code>	2357, 2362, 3859, 3869, 3902, 3907
<code>\msg_term:nnnnn</code>	2516
<code>\msg_warning:nn</code>	4059, 4208
<code>\msg_warning:nnn</code>	3156, 3160, 3165
<code>\msg_warning:nnnn</code>	2553, 2559, 3806, 3811, 4725, 4738, 4774, 4787
<code>\msg_warning:nnnnn</code>	2511, 2521

<code>\multicolsep</code>	109
<code>\multicolsep</code>	1349, 1521, 4007, 4181

N

<code>\NeedsTeXFormat</code>	3
<code>\NewCommandCopy</code>	371
<code>\newcounter</code>	582, 598
<code>\NewDocumentCommand</code>	1646, 2246, 3051, 4511, 5488, 5555, 5674, 5723, 5801

[\NewDocumentEnvironment](#) . 3181, 4099, 4238, 4452, 4942, 5212

[\newlabel](#) 40

[\newlabel](#) 416

[no-store](#) 2406

[\noindent](#) 4048, 4833, 4898, 5185, 5464

[\nointerlineskip](#) 1358, 1361, 1530, 1533, 1687, 1714, 4833, 4898

[noitemsep](#) 940

[\nopagebreak](#) 1295, 1323, 1358, 1361, 1530, 1533, 1637, 1643

[\normalfont](#) 3010, 3476, 5375

[nosep](#) 940

O

[\obeyedline](#) 3233, 3234

[overwrite](#) 3107

P

Packages:

[caption](#) 126

[enumext](#) 30, 43, 46, 76, 81, 101, 106, 117, 145

[enumitem](#) 43, 44

[expl3](#) 122

[footnotehyper](#) 39, 41, 42

[hyperref](#) 34, 35, 39, 40, 89, 97, 131, 145

[latex-lab-block](#) 39

[ltxcmd](#) 39, 91

[ltsockets](#) 115

[lua-visual-debug](#) 58

[multicol](#) 30, 145

[scontents](#) 91, 92

[shortlst](#) 122, 127, 132

[tagpdf](#) 115

[\par](#) .. 1295, 1323, 1361, 1533, 1637, 1643, 1682, 1687, 1709, 1714, 2976, 4024, 4196, 4214, 4497, 4500, 4646, 4860, 4875, 4921, 4935, 5185, 5464

para commands:

[\para_end:](#) 5202, 5482

[\parbox](#) 2634

[\parindent](#) 5167, 5442

[\parsep](#) 56, 118

[\parsep](#) 1000, 3894, 4429, 4438

[parsep](#) 940

[\parskip](#) 5168, 5443

[\partopsep](#) 3895, 4212, 4449

[partopsep](#) 940

peek commands:

[\peek_meaning:NTF](#) 5053, 5067, 5082, 5093, 5297, 5312, 5328

[\peek_meaning_remove:NTF](#) 5060, 5304

[\peek_remove_spaces:n](#) 3729

[\phantomsection](#) 40

[\phantomsection](#) 405

prg commands:

[\prg_do_nothing:](#) 409

[\prg_new_protected_conditional:Npnn](#) 221, 3148

[\prg_replicate:nn](#) 230

[\prg_return_false:](#) 225, 3161, 3169

[\prg_return_true:](#) 224, 3157, 3166

[\printkeyans](#) 20, 139, 5555

prop commands:

[\prop_const_from_keyval:Nn](#) 5715

[\prop_count:N](#) 356, 2750, 2901, 3013, 3328, 3479, 5378, 5817

[\prop_get:NnNTF](#) 5741

[\prop_gput_if_not_in:Nnn](#) 2748

[\prop_if_exist:NTF](#) 2390, 5508, 5810

[\prop_item:Nn](#) 5510, 5834

[\prop_new:N](#) 2393

[\ProvidesExplPackage](#) 4

R

[\raggedcolumns](#) 4010, 4184

[\raisebox](#) 4588

[\ref](#) 86, 95

[ref](#) 717, 763, 830

[\refstepcounter](#) 5121, 5414

regex commands:

[\regex_if_match:nnTF](#) 223, 875, 877, 889, 891

[\renewcommand](#) 751, 794, 808, 856

[\RenewDocumentCommand](#) . 433, 463, 1697, 1724, 3233, 3548, 3575, 3591, 3724, 3776, 3788, 4462

[\RequirePackage](#) 13

[reset](#) 2300

[reset*](#) 2300

[\resetenumext](#) 11, 74, 2246

[resume](#) 1840

[resume*](#) 1840

[rightmargin](#) 1020

[\Roman](#) 44, 49

[\Roman](#) 610

[\roman](#) 44, 49

[\roman](#) 611, 735, 5540

S

[save-ans](#) 2346

[save-key](#) 2688

[save-ref](#) 2600

[save-sep](#) 2565, 2600, 4359

scan commands:

[\scan_stop:](#) 4461, 4956, 5225, 5499, 5502

seq commands:

[\seq_clear:N](#) 5676, 5819

[\seq_const_from_clist:Nn](#) 5669

[\seq_count:N](#) 357, 4652, 5680

[\seq_gclear:N](#) 458, 459, 488, 489

[\seq_gput_right:Nn](#) 444, 445, 474, 475, 2757

[\seq_if_empty:NTF](#) 451, 481, 5572, 5694

[\seq_if_exist:NTF](#) 2395, 5570

[\seq_if_in:NnTF](#) 5576

[\seq_item:Nn](#) 4639

[\seq_map_function:NN](#) 5685

[\seq_map_inline:Nn](#) 5585, 5593, 5695, 5696

[\seq_map_pairwise_function:NNN](#) 453, 483

[\seq_new:N](#) 117, 118, 120, 141, 172, 173, 174, 175, 2398

[\seq_pop_left:NN](#) 5684

[\seq_put_right:Nn](#) 4525, 5692, 5708, 5829

[\seq_set_from_clist:Nn](#) 5677

[\seq_set_map_e:NNn](#) 5686

[\seq_use:Nn](#) 202, 203, 5825

[series](#) 1840

[\setcounter](#) .. 886, 890, 892, 4094, 4236, 4494, 4953, 5222

[\setenumext](#) 6, 141, 5674

[\setenumextmeta](#) 6, 143, 5715

[show-ans](#) 2565, 2600, 4359

[show-length](#) 1117

[show-pos](#) 2565, 2600, 4359

skip commands:

[\skip_add:Nn](#) 1255, 1264, 1273, 1286, 1290, 1314, 1318, 1334, 1392, 1394, 1408, 1411, 1432, 1434, 1448, 1451,

1471, 1473, 1487, 1490, 1509, 1558, 1559, 1570, 1572, 4438, 4447	
\skip_gset:Nn	1585, 1589, 1593
\skip_gzero_new:N	1580, 1581
\skip_horizontal:N	1088, 1100, 1112, 5139, 5151, 5189, 5426, 5468
\skip_horizontal:n	1074, 2825, 2833, 3541, 3543, 4569, 5038, 5137, 5171, 5282, 5446
\skip_if_eq:nnTF 1253, 1262, 1271, 1378, 1418, 1458, 1546, 1582, 1604, 1748, 1762, 1776, 1787, 1798, 1809, 1820, 1831	
\skip_new:N	71, 72, 73, 78, 79, 80, 81, 82, 83, 194
\skip_set:Nn 1238, 1242, 1300, 1304, 1328, 1381, 1382, 1400, 1421, 1422, 1440, 1460, 1461, 1479, 1503, 1549, 1550, 1564, 1584, 1588, 1606, 1610, 1614, 1620, 1624, 1628, 4422	
\skip_set_eq:NN 1339, 1340, 1342, 1349, 1514, 1515, 1516, 1521, 3847, 3891, 3894, 5168, 5443	
\skip_sub:Nn 1388, 1390, 1404, 1406, 1428, 1430, 1444, 1446, 1467, 1469, 1483, 1485, 1556, 1557, 1568, 1569	
\skip_use:N 1240, 1244, 1288, 1292, 1296, 1316, 1320, 1330, 1336, 1749, 1753, 1756, 1763, 1767, 1770, 4024	
\skip_vertical:N	566, 569, 1012, 4859, 4873, 5204, 5484
\skip_vertical:n	1011, 5203, 5483
\skip_zero:N 1348, 1362, 1500, 1501, 1502, 1520, 1534, 3895, 4007, 4181, 4449, 4450	
\skip_zero_new:N	1579, 1601, 1602, 1603
\c_zero_skip	566, 569, 1012, 1253, 1262, 1271, 1419, 1458, 1582, 1604, 1749, 1763, 1776, 1787, 1798, 1809, 1820, 1831, 4859, 4873, 5204, 5484
\small	5523, 5529, 5535, 5541, 5547, 5553
\smash	3593, 3790
socket commands:	
\socket_assign_plug:nn	4287, 4295, 4303, 4339, 4347, 4355
\socket_new:nn	4257, 4307
\socket_new_plug:nnn 4258, 4266, 4274, 4308, 4316, 4325	
\socket_use:n	4340, 4348, 4356
\socket_use:nn	4288, 4296, 4304
start	898
start*	898
start-list-tags	4257, 4307
\stepcounter	437, 467, 4432, 4581
stop-list-tags	4257, 4307
stop-start-tags	4257, 4307
str commands:	
\c_backslash_str 3104, 5876, 5881, 5886, 5891, 5893, 5895, 5900, 5902, 6000, 6004, 6008, 6018, 6022, 6030, 6031, 6035, 6047, 6048, 6052, 6053, 6074, 6076, 6080, 6082, 6122, 6185, 6187, 6191, 6193, 6202, 6203, 6207, 6212, 6213, 6217, 6221, 6225	
\c_circumflex_str	113
\c_colon_str	2900, 3327, 5499
\c_left_brace_str	5981, 5988, 5994
\c_percent_str	113
\c_right_brace_str	5981, 5988, 5994
\str_case:nn	243, 300, 3420
\str_case:nnTF	1908, 1916, 2727, 2735, 5613, 5621
\str_clear:N	3927, 4988
\str_const:Nn	112
\str_count:n	230
\str_if_empty:NTF	1925, 2019, 2026, 2060, 2067
\str_if_eq:nnTF	2258, 3851, 3898, 5725

\str_if_in:nnTF	5495
\str_new:N	74, 126, 127, 128, 146, 189
\str_set:Nn	675, 681, 687, 706, 707, 708, 2573, 2574, 2575, 2622, 2623, 2624, 4364, 4367
\str_set_eq:NN	3443, 5366, 5383
\str_use:N	3597
\strut	3593, 3790
\strutbox	1367, 1370, 1381, 1382, 1393, 1395, 1410, 1413, 1421, 1422, 1433, 1435, 1450, 1453, 1460, 1461, 1472, 1474, 1489, 1492, 1538, 1541, 1549, 1550, 1558, 1559, 1571, 1573, 1584, 1585, 1588, 1595, 1608, 1616, 1622, 1630, 4441, 4447, 4497, 4505, 4594

T

tag commands:	
\tag_mc_begin:n	4264, 4314, 4323
\tag_mc_begin_pop:n	4280, 4332, 4489, 4491
\tag_mc_end:	4268, 4318, 4327
\tag_mc_end_push:	4261, 4311, 4477
\tag_resume:n	4260, 4310, 4468, 4476, 4545, 4643, 4843, 4907
\tag_struct_begin:n	4262, 4263, 4270, 4271, 4272, 4312, 4313, 4320, 4321, 4322, 4478
\tag_struct_end:n 4269, 4276, 4277, 4278, 4279, 4319, 4328, 4329, 4330, 4331, 4488, 4490, 4962, 5231	
\tag_suspend:n	4281, 4333, 4459, 4470, 4483, 4536, 4635, 4954, 5223
\tag_tool:n	4469

TeX and LaTeX 2_ε commands:

\auxout	414
\@currentenv	243, 300
\protected@write	414

tex commands:	
\tex_scantokens:D	204

text commands:	
\text_expand:n	5491
\textasteriskcentered	2570, 2617
\textborn	3617
\textreferencemark	2605
\thepage	420

tl commands:	
\c_space_tl 3399, 3412, 5925, 5940, 5963, 5967, 6166, 6167, 6176, 6177, 6237, 6241, 6259	
\tl_clear:N	673, 680, 2161, 2168, 2563, 2674, 2684, 2705, 2713, 2920, 3271, 3344, 5334
\tl_clear_new:N	620
\tl_const:Nn	604
\tl_gclear:N	348, 349, 350, 1989, 1995, 3586, 3606, 4879, 4939, 5140
\tl_gclear_new:N	1963, 1974
\tl_gput_right:Nn	605
\tl_greplace_all:Nnn	626
\tl_gset:Nn 277, 278, 291, 292, 1964, 1975, 1990, 1996, 2389, 3517, 5088	
\tl_gset_eq:NN	622, 3513, 5133
\tl_if_blank:nTF 3043, 3061, 3140, 3190, 3644, 3665, 3691, 5131, 5789	
\tl_if_empty:NTF	740, 758, 786, 800, 817, 824, 848, 862, 1864, 1885, 1936, 1952, 2024, 2028, 2065, 2069, 2225, 2239, 2368, 2427, 2764, 2795, 2940, 3254, 3281, 3354, 3392, 3405, 3538, 4650, 5337, 5706
\tl_if_exist:NTF	2087, 2102
\tl_if_novalue:nTF 435, 465, 2253, 3057, 3186, 3279, 3352, 3385, 3492, 3511, 3519, 3701, 3925, 4413, 4986, 5263, 5335	
\tl_map_inline:Nn	623

<code>\tl_new:N</code>	29, 30, 31, 34, 37, 38, 41, 42, 46, 53, 54, 58, 59, 95, 96, 97, 103, 104, 105, 106, 107, 108, 110, 114, 115, 119, 121, 122, 123, 132, 135, 136, 153, 162, 163, 164, 167, 188
<code>\tl_put_left:Nn</code>	2772, 2803, 2925, 4863, 4924, 5353, 5356
<code>\tl_put_right:Nn</code>	621, 854, 2776, 2807, 2854, 2864, 2877, 2892, 2898, 2903, 2927, 2932, 2939, 2942, 2952, 2957, 2960, 2966, 3239, 3274, 3277, 3283, 3285, 3312, 3317, 3322, 3325, 3334, 3347, 3350, 3356, 3358, 3368, 5339, 5340
<code>\tl_remove_all:Nn</code>	5705
<code>\tl_remove_once:Nn</code>	2842, 3297
<code>\tl_replace_all:Nnn</code>	625, 3234, 5740
<code>\tl_reverse:N</code>	2841, 2843, 3296, 3298
<code>\tl_set:Nn</code>	43, 247, 257, 304, 305, 312, 313, 320, 321, 581, 674, 679, 685, 686, 739, 749, 783, 792, 806, 847, 1071, 1085, 1098, 1110, 1863, 1884, 2162, 2169, 2367, 2675, 2685, 2706, 2714, 3007, 3126, 3232, 3387, 3473, 3632, 3655, 3679, 4398, 5342, 5372, 5703, 5739, 5809
<code>\tl_set_eq:NN</code>	631, 745, 791, 805, 853, 2840, 3295, 3308, 3441, 5365
<code>\tl_to_str:n</code>	2087, 2091, 2102, 2106, 5491
<code>\tl_trim_spaces:n</code>	621, 5692, 5703, 5709, 5725
<code>\tl_use:N</code>	627, 630, 760, 819, 826, 864, 1143, 1147, 1151, 1155, 1159, 1163, 1167, 1171, 1175, 1179, 1183, 1187, 1191, 1195, 1199, 1203, 2830, 2847, 2855, 2866, 2879, 2884, 2895, 3500, 3506, 3534, 3577, 3579, 3585, 3600, 3704, 3708, 3715, 3778, 3781, 3783, 3796, 4106, 4244, 4566, 4574, 4870, 4931, 5144, 5172, 5173, 5423, 5447, 5452, 5559, 5560, 5561, 5562, 5563, 5581, 5688, 5807
token commands:	
<code>\token_to_str:N</code>	416
<code>\topsep</code>	4212, 4447
<code>topsep</code>	940
<code>\topskip</code>	1348, 1520
<code>\typeout</code>	1866, 1887, 1938, 1954
U	
<code>\unkern</code>	238
<code>unknown</code>	3021, 3107, 3627, 3652, 3673
<code>\unskip</code>	237
use commands:	
<code>\use:N</code>	231, 3582, 3603, 4108
<code>\use:n</code>	1899, 2718, 5497, 5604
<code>\use_none:nn</code>	408, 5746
<code>\usecounter</code>	3850, 3896
V	
<code>\value</code>	2009, 2051, 2063, 2079
vbox commands:	
<code>\vbox_set:Nn</code>	4538
<code>\vbox_set_top:Nn</code>	4868, 4929
<code>\vspace</code>	1000, 1753, 1756, 1767, 1770, 1780, 1782, 1791, 1793, 1802, 1804, 1813, 1815, 1824, 1826, 1835, 1837
W	
<code>widest</code>	898
<code>wrap-ans</code>	2600
<code>wrap-ans*</code>	2565, 2600, 4359
<code>wrap-label</code>	644
<code>wrap-label*</code>	644
<code>wrap-opt</code>	2565, 2600, 4359
<code>write-env</code>	3107