

enumext

ENUMERATE EXERCISE SHEETS

v1.6 2025-07-01^{*}

©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	152

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 2025-07-01.

[†]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 (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.
The enumext package loads and requires multicol[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

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
- ©2024–2025 by Pablo González L
- 2 / 168

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 \TeX `\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 `shortenumerate` 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*` = {<symbol>} default: \textborn
Sets the *symbol* to be displayed in the “left” of the box containing the current <label> set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in *text* or *math* mode, for example `item-sym*={\star}`.
`item-pos*` = {<rigid length>} default: by levels
Sets the *offset* between the box containing the current <label> defined by `labelwidth` key and the <symbol> set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* 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(<columns>)` 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{<key = val>}</code>	<code>\setenumext[<keyans*>]{<key = val>}</code>
	<code>\setenumext[<enumext, level>]{<key = val>}</code>	<code>\setenumext[<print, level>]{<key = val>}</code>
	<code>\setenumext[<enumext*>]{<key = val>}</code>	<code>\setenumext[<print, *>]{<key = val>}</code>
	<code>\setenumext[<keyans>]{<key = val>}</code>	<code>\setenumext[<print*>]{<key = val>}</code>

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

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

🔒 For security reasons the keys `resume with value`, `resume*`, `reset`, `reset*`, `series` and `save-ans` they can NOT be set by this command and are ignored. The key `save-ans` that 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 {<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta*{<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta [<enumext*>]{<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta [<enumext, level>]{<key name>}{<key-one = val, key-two = val, ...>}</code>

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the {<key name>} must be different from those defined by the package. 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` = { $\langle \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \rangle$ } 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 $\langle \backslash alph* \rangle$, for third level are `\roman*`, and for fourth level are `\Alph*`. For `keyans` and `keyans*` environments the default value is `\Alph*`.

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

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

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

`resume*` *<value forbidden>* default: *not used*

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

- When using the key `resume={ <series name> }` or `resume*` you will have hierarchy in the *<keys>* that are stored in *{ <series name> }* or in an internal version of *{ <series name> }* in the case of `resume*`. If you want to *reset* the value of a *<key>* that is already stored in *{ <series name> }* or in an internal version of *{ <series name> }* this must be placed to the *right* of the key `resume={ <series name> }` or `resume*`.

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 multicol

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

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

`columns-sep = { <rigid length> }` default: *by level*

Set the *space between columns* used by the `multicol` 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 : 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.

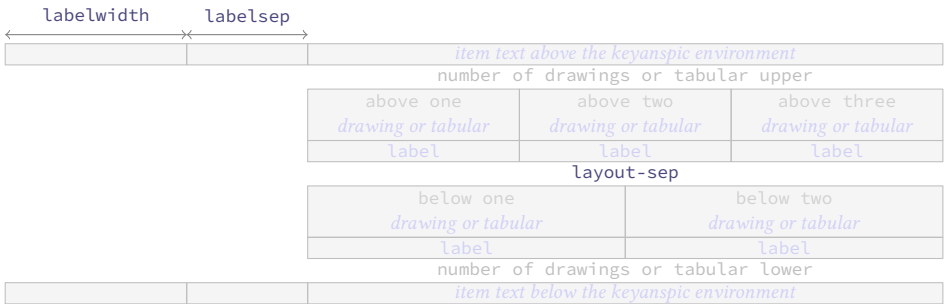


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

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.

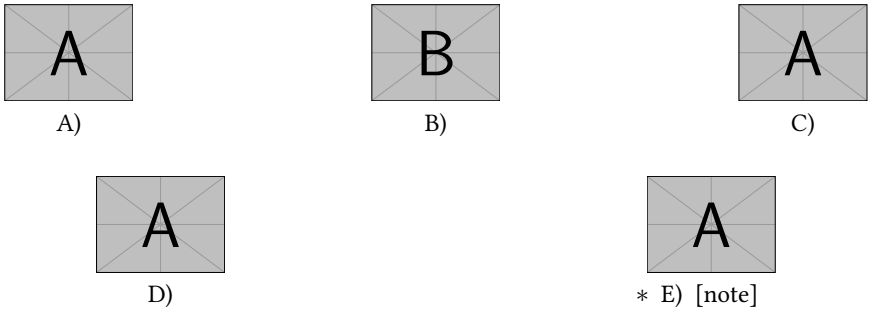
  \begin{keyanspic}[label-pos=above, layout-sty={3,2},layout-sep=0.25cm]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}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

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

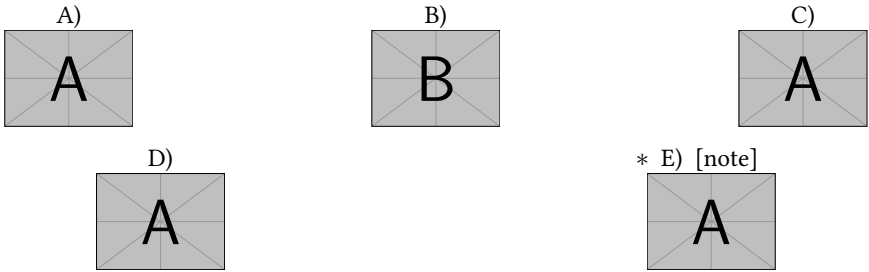
  \begin{keyanspic}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}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

\end{enumext}
```

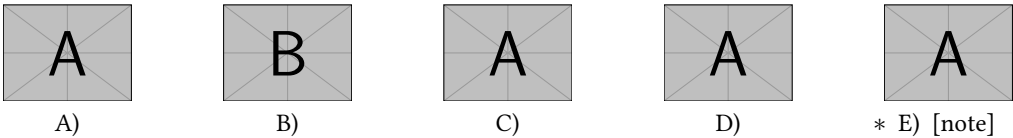
1. Question with images and labels below.



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={\makebox[1em][l]{\#1}}]{\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[print, 1]{ $\langle keys \rangle$ }` and `\setenumext[print*]{ $\langle keys \rangle$ }`.

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[print, level]{ $\langle keys \rangle$ }` 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[print, *]{ $\langle keys \rangle$ }`.

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 e is cool?

 Related to Linux

(a) You use linux?

©2024–2025 by Pablo González L


22 / 168

- (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 L^AT_EX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.
- [11] The L^AT_EX Project. “The L^AT_EX₃ Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.
- [12] The L^AT_EX Project. “The L^AT_EX_{2_ε} sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [13] The L^AT_EX Project. “L^AT_EX News, Issue 41, June 2025”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [14] The L^AT_EX Project. “L^AT_EX 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 – L^AT_EX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2025.
- [19] The L^AT_EX Project. “latex-lab – L^AT_EX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [20] MITTELBACH, FRANK. “L^AT_EX’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.

11 Change history

- v1.6 (ctan), 2025-07-01**
 - Environments can be started with the key `resume` *without value*.
 - Add `\resetenumext`, `reset` and `reset*` keys.
 - The `resume`, `resume*` and `series` keys can now be set per level.
 - Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.
- v1.5 (ctan), 2025-06-11**
 - Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
 - Add keys `beginpenalty`, `midpenalty` and `endpenalty`.
- v1.4 (ctan), 2025-06-09**
 - Improved implementation of the `start` key for *tagged* PDF.
 - Improved implementation of the `ref` key.
 - Fixed the behavior of the `save-sep` key.
 - Fixed the behavior of the `resume*` key.
- v1.3 (ctan), 2025-06-01**
 - Removed dependency on the `scontents` package.
 - The `anskey*` environment has been rewritten using the new `c`-type argument.
- v1.2 (ctan), 2025-03-28**
 - Replace signature (prevent expansion for optional argument).
 - Solve Inconsistent local/global assignment.
- v1.1 (ctan), 2024-11-14**
 - Fixed implementation for `font` and `base-fix` keys.
 - Added new keys for symbol marks.
 - Update and improvements in the internal code.
 - Adjustments in the documentation.
- v1.0 (ctan), 2024-11-01**
 - First public release.

12 Index of Documentation

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

C		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} {2025-07-01} {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_count_bool
\l__enumext_resume_series_X_bool
\g__enumext_save_last_keys_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_series_#1_bool }
51   \tl_new:c { g__enumext_save_last_keys_#1_tl }
52 }
53 \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.

```

54 \dim_new:N \l__enumext_current_widest_dim
55 \tl_new:N \g__enumext_counter_styles_tl
56 \tl_new:N \g__enumext_widest_label_tl
57 \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:Nnnnnnnnnnn` (§13.41.1).

```

58 \cs_set_protected:Npn \__enumext_tmp:n #1
59 {
60   \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }

```

```

61      \dim_new:c { \__enumext_leftmargin_tmp_#1_dim }
62      \dim_new:c { \__enumext_leftmargin_#1_dim }
63      \dim_new:c { \__enumext_itemindent_#1_dim }
64    }
65    \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_multicols_above_X_skip
\__enumext_multicols_below_X_skip
\__enumext_multicols_right_X_skip
\__enumext_align_label_pos_X_str

```

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

```

66 \cs_set_protected:Npn \__enumext_tmp:n #1
67 {
68   \skip_new:c { \__enumext_multicols_above_#1_skip }
69   \skip_new:c { \__enumext_multicols_below_#1_skip }
70   \skip_new:c { \__enumext_multicols_right_#1_skip }
71   \str_new:c { \__enumext_align_label_pos_#1_str }
72 }
73 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_minipage_stat_int
\__enumext_minipage_temp_skip
\__enumext_minipage_left_skip
\__enumext_minipage_right_skip
\__enumext_minipage_after_skip
\__enumext_minipage_right_skip
\__enumext_minipage_after_skip
\__enumext_minipage_left_X_dim
\__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).

```

74 \int_new:N \__enumext_minipage_stat_int
75 \skip_new:N \__enumext_minipage_temp_skip
76 \skip_new:N \__enumext_minipage_left_skip
77 \skip_new:N \__enumext_minipage_right_skip
78 \skip_new:N \__enumext_minipage_after_skip
79 \skip_new:N \__enumext_minipage_right_skip
80 \skip_new:N \__enumext_minipage_after_skip
81 \cs_set_protected:Npn \__enumext_tmp:n #1
82 {
83   \dim_new:c { \__enumext_minipage_left_#1_dim }
84   \bool_new:c { \__enumext_minipage_active_#1_bool }
85 }
86 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_wrap_label_X_bool
\__enumext_wrap_label_opt_X_bool
\__enumext_start_X_int
\__enumext_fake_item_indent_X_tl
\__enumext_label_fill_left_X_tl
\__enumext_label_fill_right_X_tl
\__enumext_vspace_a_star_X_bool
\__enumext_vspace_b_star_X_bool

```

The bool vars `__enumext_wrap_label_X_bool` and `__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.19.1), the variables `__enumext_label_fill_left_X_tl` and `__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). The boolean vars `__enumext_vspace_a_star_X_bool`, `__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.21).

```

87 \cs_set_protected:Npn \__enumext_tmp:n #1
88 {
89   \bool_new:c { \__enumext_wrap_label_#1_bool }
90   \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
91   \int_new:c { \__enumext_start_#1_int }
92   \tl_new:c { \__enumext_fake_item_indent_#1_tl }
93   \tl_new:c { \__enumext_label_fill_left_#1_tl }
94   \tl_new:c { \__enumext_label_fill_right_#1_tl }
95   \bool_new:c { \__enumext_vspace_a_star_#1_bool }
96   \bool_new:c { \__enumext_vspace_b_star_#1_bool }
97 }
98 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_store_active_bool
\__enumext_store_name_tl
\__enumext_store_name_tl
\__enumext_store_current_label_tl
\__enumext_store_current_opt_arg_tl

```

The variable `__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 `__enumext_store_name_tl` saves the `{<store name>}` set by the `save-ans` key of the *sequence* and *prop list* in which we will store, the variable `__enumext_store_name_tl` it's just a global copy of `{<store name>}` used by different functions.

The variables `__enumext_store_current_label_tl` and `__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.

```

99 \bool_new:N \__enumext_store_active_bool
100 \tl_new:N \__enumext_store_name_tl

```

```
101 \tl_new:N \g__enumext_store_name_tl
102 \tl_new:N \l__enumext_store_current_label_tl
103 \tl_new:N \l__enumext_store_current_opt_arg_tl
```

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

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.

```
104 \tl_new:N \l__enumext_store_anskey_arg_tl
105 \tl_new:N \l__enumext_store_anskey_env_tl
106 \bool_new:N \l__enumext_write_anskey_env_bool
107 \tl_new:N \l__enumext_write_anskey_env_file_name_tl
108 \iow_new:N \l__enumext_write_anskey_env_file_iow
```

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

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.

```
109 \str_const:Ne \c__enumext_anskey_env_hidden_space_str
110 { \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`.)

Internal variables used by the command `\setenumext` (§13.51).

```
111 \tl_new:N \l__enumext_setkey_tmpa_tl
112 \tl_new:N \l__enumext_setkey_tmpb_tl
113 \int_new:N \l__enumext_setkey_tmpa_int
114 \seq_new:N \l__enumext_setkey_tmpa_seq
115 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

```
116 \tl_new:N \l__enumext_meta_path_tl
117 \seq_new:N \l__enumext_foreach_print_seq
118 \tl_new:N \l__enumext_foreach_name_prop_tl
119 \tl_new:N \l__enumext_foreach_default_keys_tl
```

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

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*”.

```
120 \tl_new:N \l__enumext_print_keyans_starred_tl
121 \bool_new:N \l__enumext_print_keyans_star_bool
122 \bool_new:N \l__enumext_print_keyans_cmd_bool
123 \str_new:N \l__enumext_mark_position_str
124 \str_new:N \l__enumext_mark_position_v_str
125 \str_new:N \l__enumext_mark_position_viii_str
126 \dim_new:N \l__enumext_mark_sep_tmpa_dim
127 \dim_new:N \l__enumext_mark_sep_tmpb_dim
128 \int_new:N \l__enumext_show_pos_tmp_int
129 \tl_new:N \g__enumext_item_symbol_aux_tl
130 \cs_set_protected:Npn \__enumext_tmp:n #1
131 {
132   \tl_new:c { \l__enumext_print_keyans_#1_tl }
133   \tl_new:c { \l__enumext_store_save_key_#1_tl }
134   \bool_new:c { \l__enumext_store_save_key_#1_bool }
135   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
136 }
137 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_anspic_args_seq
  \l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
  \l__enumext_anspic_label_above_bool
  \l__enumext_anspic_mini_pos_str
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
  \l__enumext_anspic_label_htdp_dim
  \l__enumext_anspic_body_htdp_dim
```

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

```
138 \seq_new:N \l__enumext_anspic_args_seq
139 \dim_new:N \l__enumext_anspic_mini_width_dim
140 \int_new:N \l__enumext_anspic_above_int
141 \int_new:N \l__enumext_anspic_below_int
142 \bool_new:N \l__enumext_anspic_label_above_bool
143 \str_new:N \l__enumext_anspic_mini_pos_str
144 \box_new:N \l__enumext_anspic_label_box
145 \box_new:N \l__enumext_anspic_body_box
146 \dim_new:N \l__enumext_anspic_label_htdp_dim
147 \dim_new:N \l__enumext_anspic_body_htdp_dim
```

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

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

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.

```
148 \bool_new:N \l__enumext_check_answers_bool
149 \bool_new:N \g__enumext_check_ans_key_bool
150 \tl_new:N \l__enumext_check_start_line_env_tl
151 \bool_new:N \l__enumext_item_wrap_key_bool
152 \int_new:N \g__enumext_check_starred_cmd_int
153 \int_new:N \g__enumext_item_anskey_int
154 \int_new:N \g__enumext_item_number_int
155 \bool_new:N \l__enumext_item_number_bool
156 \int_new:N \g__enumext_item_answer_diff_int
```

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

```
\l__enumext_hyperref_bool
  \l__enumext_footnotes_key_bool
```

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

```
157 \bool_new:N \l__enumext_hyperref_bool
158 \bool_new:N \l__enumext_footnotes_key_bool
```

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

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

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

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

```
159 \tl_new:N \l__enumext_newlabel_arg_one_tl
160 \tl_new:N \l__enumext_newlabel_arg_two_tl
161 \tl_new:N \l__enumext_write_aux_file_tl
162 \cs_set_protected:Npn \__enumext_tmp:n #1
163 {
164   \tl_new:c { \l__enumext_label_copy_#1_tl }
165 }
166 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\g__enumext_footnote_standar_int
\g__enumext_footnote_starred_int
\g__enumext_footnote_standar_arg_seq
\g__enumext_footnote_starred_arg_seq
\g__enumext_footnote_standar_int_seq
\g__enumext_footnote_starred_int_seq
```

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

```
167 \int_new:N \g__enumext_footnote_standar_int
168 \int_new:N \g__enumext_footnote_starred_int
169 \seq_new:N \g__enumext_footnote_standar_arg_seq
170 \seq_new:N \g__enumext_footnote_starred_arg_seq
171 \seq_new:N \g__enumext_footnote_standar_int_seq
172 \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
```

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

```
173 \cs_set_protected:Npn \__enumext_tmp:n #1
174 {
175   \bool_new:c { \l__enumext_item_starred_#1_bool }
176   \int_new:c { \l__enumext_item_column_pos_#1_int }
177   \int_new:c { \g__enumext_item_count_all_#1_int }
178   \int_new:c { \l__enumext_joined_item_#1_int }
```



```

179 \int_new:c { \__enumext_joined_item_aux_#1_int }
180 \int_new:c { \__enumext_tmpa_#1_int }
181 \dim_new:c { \__enumext_tmpa_#1_dim }
182 \box_new:c { \__enumext_item_text_#1_box }
183 \dim_new:c { \__enumext_joined_width_#1_dim }
184 \dim_new:c { \__enumext_item_width_#1_dim }
185 \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
186 \str_new:c { \__enumext_align_label_#1_str }
187 \bool_new:c { g__enumext_minipage_active_#1_bool }
188 \box_new:c { \__enumext_miniright_code_#1_box }
189 \bool_new:c { g__enumext_minipage_center_#1_bool }
190 \dim_new:c { g__enumext_minipage_right_#1_dim }
191 \skip_new:c { g__enumext_minipage_right_#1_skip }
192 }
193 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

194 \clist_const:Nn \c__enumext_all_envs_clist
195 {
196   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
197   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
198 }

```

(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).

```

199 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
200 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

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

`__enumext_scan_tokens:n` The functions `\tl_rescan:nn` 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?](#).

```

201 \cs_new_protected:Npn \__enumext_scan_tokens:n #1 { \tex_scantokens:D {#1} }

```

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

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

```

202 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
203 {
204   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
205 }

```

(End of definition for `__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.

```

206 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
207 {
208   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
209 }
210 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
211 {
212   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
213 }

```

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

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

```

214 \cs_new:Nn \__enumext_level:
215 {
216   \int_to_roman:n { \l__enumext_level_int }
217 }

```

(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`

```
218 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
219 {
220   \regex_if_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
221   { \prg_return_true: }
222   { \prg_return_false: }
223 }
```

(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.

```
224 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
225 {
226   *~#2
227   \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
228   =~\use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
229 }
```

(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.

```
230 \cs_new_protected:Nn \__enumext_unskip_unkern:
231 {
232   \int_case:nnT { \lastnodetype }
233   {
234     { 11 } { \unskip }
235     { 12 } { \unkern }
236   }
237 }
```

(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:`

```
238 \cs_new_protected:Nn \__enumext_is_not_nested:
239 {
240   \str_case:en { \@currenvir }
241   {
242     {enumext}
243     {
244       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
245       \bool_lazy_and:nnT
246       { \bool_not_p:n { \g__enumext_standar_bool } }
247       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
248       {
249         \bool_gset_true:N \g__enumext_standar_bool
250       }
251     }
252     {enumext*}
253     {
254       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
255       \bool_lazy_and:nnT
256       { \bool_not_p:n { \g__enumext_starred_bool } }
257       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
258       {
259         \bool_gset_true:N \g__enumext_starred_bool
260       }
261     }
262   }
263 }
```

The function `__enumext_is_on_first_level:` will set the variables `__enumext_standar_first_bool` (§13.29.1), `__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.

```

264 \cs_new_protected:Nn \__enumext_is_on_first_level:
265 {
266   \bool_lazy_all:nT
267   {
268     { \bool_if_p:N \g__enumext_standar_bool }
269     { \int_compare_p:nNn { \__enumext_level_int } = { 1 } }
270     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
271   }
272   {
273     \bool_set_true:N \__enumext_standar_first_bool
274     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
275     \tl_gset:Ne \g__enumext_start_line_tl
276     {
277       on~line~\exp_not:V \inputlineno
278     }
279   }
280   \bool_lazy_all:nT
281   {
282     { \bool_if_p:N \g__enumext_starred_bool }
283     { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
284     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
285   }
286   {
287     \bool_set_true:N \__enumext_starred_first_bool
288     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
289     \tl_gset:Ne \g__enumext_start_line_tl
290     {
291       on~line~\exp_not:V \inputlineno
292     }
293   }
294 }

```

(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 `__enumext_check_start_line_env_tl` and `__enumext_envir_name_tl` to use in the `__enumext_check_starred_cmd:n` function.

```

295 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
296 {
297   \str_case:en { \@currentenvir }
298   {
299     {keyans}
300     {
301       \tl_set:Nn \__enumext_envir_name_tl { keyans }
302       \tl_set:Ne \__enumext_check_start_line_env_tl
303       {
304         in~'keyans'~start~on~line~\exp_not:V \inputlineno
305       }
306     }
307     {keyans*}
308     {
309       \tl_set:Nn \__enumext_envir_name_tl { keyans* }
310       \tl_set:Ne \__enumext_check_start_line_env_tl
311       {
312         in~'keyans*'~start~on~line~\exp_not:V \inputlineno
313       }
314     }
315     {keyanspic}
316     {
317       \tl_set:Nn \__enumext_envir_name_tl { keyanspic }
318       \tl_set:Ne \__enumext_check_start_line_env_tl
319       {
320         in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
321       }
322     }

```

```

323     }
324 }

```

(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.

```

325 \cs_new_protected:Nn \__enumext_reset_global_vars:
326 {
327     \__enumext_reset_global_int:
328     \__enumext_reset_global_bool:
329     \__enumext_reset_global_tl:
330 }
331 \cs_new_protected:Nn \__enumext_reset_global_int:
332 {
333     \int_gzero:N \g__enumext_item_number_int
334     \int_gzero:N \g__enumext_item_anskey_int
335     \int_gzero:N \g__enumext_item_answer_diff_int
336 }
337 \cs_new_protected:Nn \__enumext_reset_global_bool:
338 {
339     \bool_gset_false:N \g__enumext_check_ans_key_bool
340     \bool_gset_false:N \g__enumext_standar_bool
341     \bool_gset_false:N \g__enumext_starred_bool
342 }
343 \cs_new_protected:Nn \__enumext_reset_global_tl:
344 {
345     \tl_gclear:N \g__enumext_store_name_tl
346     \tl_gclear:N \g__enumext_start_line_tl
347     \tl_gclear:N \g__enumext_envir_name_tl
348 }

```

(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.

```

349 \cs_new_protected:Nn \__enumext_log_global_vars:
350 {
351     \msg_log:nneeee { enumext } { prop-seq-int-hook }
352     { \g__enumext_store_name_tl }
353     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
354     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
355     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
356 }

```

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.

```

357 \cs_new_protected:Nn \__enumext_log_answer_vars:
358 {
359     \msg_log:nneeee { enumext } { item-answer-hook }
360     { \int_use:N \g__enumext_item_number_int }
361     { \int_use:N \g__enumext_item_anskey_int }
362     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
363 }

```

(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 \TeX has the following plain form:

```

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

```

And `minipage` environment provided by \TeX 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.

```
364 \__enumext_at_begin_document:n
365 {
366   \cs_new_eq:NN \__enumext_start_list:nn \list
367   \cs_new_eq:NN \__enumext_stop_list: \endlist
368   \NewCommandCopy \__enumext_item_std:w \item
369 }
```

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

```
370 \__enumext_at_begin_document:n
371 {
372   \cs_new_eq:NN \__enumext_minipage:w \minipage
373   \cs_new_eq:NN \__enumext_endminipage: \endminipage
374 }
```

(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.

```
375 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
376 \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 `__enumext_footnotes_key_bool` to “true”.

```
377 \cs_new_protected:Nn \__enumext_after_hyperref:
378 {
379   \IfPackageLoadedT { hyperref }
380   {
381     \msg_info:nnn { enumext } { package-load } { hyperref }
382     \bool_set_true:N \l__enumext_hyperref_bool
383     \IfHyperBoolean{hyperfootnotes}
384     {
385       \bool_set_true:N \l__enumext_footnotes_key_bool
386     }
387   }
388 }
```

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`.

```
389 \bool_if:NT \l__enumext_footnotes_key_bool
390 {
391   \IfPackageLoadedTF { footnotehyper }
392   {
393     \msg_info:nnn { enumext } { package-load } { footnotehyper }
394   }
395   {
396     \bool_set_false:N \l__enumext_footnotes_key_bool
397   }
398 }
```

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.

```
399 \bool_if:NTF \l__enumext_hyperref_bool
400 {
401   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
402   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
403 }
404 {
405   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
```

```

406         \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
407     }
408 }

```

(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.

```

409 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
410 {
411     \protected@write \@auxout { }
412     {
413         \token_to_str:N \newlabel {#1}
414         {
415             {#2}
416             \bool_if:NT \__enumext_hyperref_bool
417             { { \thepage } {#2} {#1} }
418             { }
419         }
420     }
421     \__enumext_hypertarget:nn {#1} { }
422     \__enumext_phantomsection:
423 }

```

(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.

```

424 \cs_new_protected:Nn \__enumext_footnotetext:nn
425 {
426     \footnotetext[#1]{#2}
427 }
428 \cs_new_protected:Nn \__enumext_renew_footnote:
429 {
430     \RenewDocumentCommand \footnote { o +m }
431     {
432         \tl_if_novalue:nTF {##1}
433         {
434             \stepcounter{footnote}
435             \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
436         }
437         {
438             \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
439         }
440         \footnotemark [ \g__enumext_footnote_standar_int ]
441         \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
442         \seq_gput_right:NV
443             \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
444     }
445 }
446 \cs_new_protected:Nn \__enumext_print_footnote:
447 {
448     \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
449     {
450         \seq_map_pairwise_function:NNN
451             \g__enumext_footnote_standar_int_seq
452             \g__enumext_footnote_standar_arg_seq
453             \__enumext_footnotetext:nn
454     }
455     \seq_gclear:N \g__enumext_footnote_standar_arg_seq
456     \seq_gclear:N \g__enumext_footnote_standar_int_seq
457 }

```


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.

```

458 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
459 {
460   \RenewDocumentCommand \footnote { o +m }
461   {
462     \tl_if_novalue:nTF {##1}
463     {
464       \stepcounter{footnote}
465       \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
466     }
467     {
468       \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
469     }
470     \footnotemark [ \g__enumext_footnote_starred_int ]
471     \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
472     \seq_gput_right:NV
473       \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
474   }
475 }
476 \cs_new_protected:Nn \__enumext_print_footnote_mini:
477 {
478   \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
479   {
480     \seq_map_pairwise_function:NNN
481       \g__enumext_footnote_starred_int_seq
482       \g__enumext_footnote_starred_arg_seq
483       \__enumext_footnotetext:nn
484   }
485   \seq_gclear:N \g__enumext_footnote_starred_arg_seq
486   \seq_gclear:N \g__enumext_footnote_starred_int_seq
487 }

```

(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.

```

488 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
489 {
490   \bool_if:NT \g__enumext_standar_bool
491   {
492     \IfDocumentMetadataTF
493     {
494       \__enumext_renew_footnote:
495     }
496     {
497       \bool_if:NF \l__enumext_footnotes_key_bool
498       {
499         \__enumext_renew_footnote:
500       }
501     }
502   }
503 }
504 \cs_new_protected:Nn \__enumext_print_footnote_standar:
505 {
506   \bool_if:NT \g__enumext_standar_bool
507   {
508     \IfDocumentMetadataTF
509     {
510       \__enumext_print_footnote:
511     }
512     {
513       \bool_if:NF \l__enumext_footnotes_key_bool
514       {
515         \__enumext_print_footnote:
516       }
517     }
518   }
519 }

```

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.

```

520 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
521 {
522   \IfDocumentMetadataTF
523   {
524     \__enumext_renew_footnote_mini:
525   }
526   {
527     \bool_if:NF \l__enumext_footnotes_key_bool
528     {
529       \__enumext_renew_footnote_mini:
530     }
531   }
532 }
533 \cs_new_protected:Nn \__enumext_print_footnote_starred:
534 {
535   \IfDocumentMetadataTF
536   {
537     \__enumext_print_footnote_mini:
538   }
539   {
540     \bool_if:NF \l__enumext_footnotes_key_bool
541     {
542       \__enumext_print_footnote_mini:
543     }
544   }
545 }

```

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

```

546 \__enumext_after_env:nn { enumext* }
547 {
548   \__enumext_print_footnote_starred:
549 }
550 \__enumext_after_env:nn { keyans* }
551 {
552   \__enumext_print_footnote_starred:
553 }

```

(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)

```

554 \cs_new_protected:Nn \__enumext_internal_mini_page:
555 {
556   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
557   {
558     \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
559     {
560       \__enumext_renew_footnote_standar:
561       \__enumext_minipage:w [ t ] { ##1 }
562       \legacy_if_gset_false:n { @minipage }
563       \skip_vertical:N \c_zero_skip
564     }
565     {
566       \skip_vertical:N \c_zero_skip
567       \__enumext_endminipage:
568       \__enumext_print_footnote_standar:
569     }
570   }
571 }

```

(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.

```
572 \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
573 \cs_new_protected:Npn \__enumext_define_counter:Nn #1 #2
574 {
575   \cs_if_exist:cTF { c@ #2 }
576   { \msg_fatal:nnn { enumext } { counters } { #2 } }
577   {
578     \tl_set:Nn #1 { #2 }
579     \newcounter { #2 }
580   }
581 }
```

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.

```
582 \__enumext_define_counter:Nn \__enumext_counter_i_tl { enumXi }
583 \__enumext_define_counter:Nn \__enumext_counter_ii_tl { enumXii }
584 \__enumext_define_counter:Nn \__enumext_counter_iii_tl { enumXiii }
585 \__enumext_define_counter:Nn \__enumext_counter_iv_tl { enumXiv }
586 \__enumext_define_counter:Nn \__enumext_counter_v_tl { enumXv }
587 \__enumext_define_counter:Nn \__enumext_counter_vi_tl { enumXvi }
588 \__enumext_define_counter:Nn \__enumext_counter_vii_tl { enumXvii }
589 \__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
590 \cs_set_protected:Npn \__enumext_tmp:n #1
591 {
592   \cs_if_exist:cTF { c@ __enumext_resume_#1_int }
593   { \msg_fatal:nne { enumext } { counters } { __enumext_resume_#1_int } }
594   {
595     \newcounter { __enumext_resume_#1_int }
596   }
597 }
598 \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.

```
599 \cs_new_protected:Npn \__enumext_register_default_label_wd:Nn #1 #2
600 {
601   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
602   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
603 }
604 \__enumext_register_default_label_wd:Nn \arabic { 0 }
```

```

605 \__enumext_register_default_label_wd:Nn \Alph { M }
606 \__enumext_register_default_label_wd:Nn \alph { m }
607 \__enumext_register_default_label_wd:Nn \Roman { VIII }
608 \__enumext_register_default_label_wd:Nn \roman { viii }

```

(End of definition for __enumext_register_default_label_wd:Nn.)

```

\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv

```

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

```

609 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
610 {
611   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
612   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
613 }
614 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }

```

(End of definition for __enumext_label_width_by_box:Nn.)

```

\__enumext_label_style:Nnn
\__enumext_label_style:cvn

```

The function __enumext_label_style:Nnn is used by the label key to 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.

```

615 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
616 {
617   \tl_clear_new:N #1
618   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
619   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
620   \tl_map_inline:Nn \g__enumext_counter_styles_tl
621   {
622     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
623     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
624     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
625   }
626   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
627   { \tl_use:N \g__enumext_widest_label_tl }
628   \tl_set_eq:cN { the #2 } #1
629 }
630 \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.

```

631 \cs_set_protected:Npn \__enumext_tmp:n #1
632 {
633   \keys_define:nn { enumext / #1 }
634   {
635     mode-box .bool_set:N = \l__enumext_mode_box_bool,
636     mode-box .initial:n = false,
637     mode-box .value_forbidden:n = true,
638   }
639 }
640 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for mode-box.)

```

font
labelsep
labelwidth
wrap-label
wrap-label*

```

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

```

641 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
642 {
643   \keys_define:nn { enumext / #1 }
644   {
645     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
646     font .value_required:n = true,
647     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },

```

```

648     labelsep .initial:n = {0.333em},
649     labelsep .value_required:n = true,
650     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
651     labelwidth .value_required:n = true,
652     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
653     wrap-label .initial:n = {##1},
654     wrap-label .value_required:n = true,
655     wrap-label* .code:n = {
656         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
657         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
658     },
659     wrap-label* .value_required:n = true,
660 }
661 }
662 \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.

```

663 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
664 {
665     \keys_define:nn { enumext / #1 }
666     {
667         align .choice:,
668         align / left .code:n =
669             {
670                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
671                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
672                 \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
673             },
674         align / right .code:n =
675             {
676                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
677                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
678                 \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
679             },
680         align / center .code:n =
681             {
682                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
683                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
684                 \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
685             },
686         align / unknown .code:n =
687             \msg_error:nneee { enumext } { unknown-choice }
688             { align } { left,~right,~ center } { \exp_not:n {##1} },
689         align .initial:n = left,
690         align .value_required:n = true,
691     }
692 }
693 \clist_map_inline:nn
694 {
695     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
696 }
697 { \__enumext_tmp:nn #1 }
698 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
699 {
700     \keys_define:nn { enumext / #1 }
701     {
702         align .choice:,
703         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
704         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
705         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
706         align / unknown .code:n =
707             \msg_error:nneee { enumext } { unknown-choice }
708             { align } { left,~right,~ center } { \exp_not:n {##1} },
709         align .initial:n = left,
710         align .value_required:n = true,
711     }
712 }
713 \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.

```

label
ref
\l__enumext_label_i_tl 714 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 715 {
\l__enumext_label_iii_tl 716 \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 717 {
718     label .code:n = {
719         \__enumext_label_style:cnv { \l__enumext_label_#2_tl }
720         { \l__enumext_counter_#2_tl } {##1}
721         \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
722         \l__enumext_current_widest_dim
723     },
724     label .initial:n = #3,
725     label .value_required:n = true,
726     ref .code:n = \__enumext_standar_ref:n {##1},
727     ref .value_required:n = true,
728 }
729 }
730 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
731 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
732 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
733 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

`__enumext_standar_ref:n` 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`.

```

734 \cs_new_protected:Npn \__enumext_standar_ref:n #1
735 {
736     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
737     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
738     {
739         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
740     }
741     {
742         \tl_set_eq:Nc \l__enumext_ref_the_count_tl
743         {
744             \l__enumext_the_counter_ \__enumext_level: _tl
745         }
746         \tl_set:ce { \l__enumext_renew_counter_ \__enumext_level: _tl }
747         {
748             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
749             { \exp_not:V \l__enumext_ref_key_arg_tl }
750         }
751     }
752 }

```

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

```

753 \cs_new_protected:Npn \__enumext_standar_ref:
754 {
755     \tl_if_empty:cF { \l__enumext_renew_counter_ \__enumext_level: _tl }
756     {
757         \tl_use:c { \l__enumext_renew_counter_ \__enumext_level: _tl }
758     }
759 }

```

(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 $\langle labels \rangle$ for `enumext*` and `keyans*` environments, along with the default value for `labelwidth` key and `ref` key.

```

label
ref
\l__enumext_label_vii_tl 760 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 761 {
762   \keys_define:nn { enumext / #1 }
763   {
764     label .code:n = {
765       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
766       { \l__enumext_counter_#2_tl } {##1}
767       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
768       \l__enumext_current_widest_dim
769     },
770     label .initial:n = #3,
771     label .value_required:n = true,
772     ref .code:n = \l__enumext_starred_ref:n {##1},
773     ref .value_required:n = true,
774   }
775 }
776 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
777 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

(End of definition for label and others.)

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

```

778 \cs_new_protected:Npn \l__enumext_starred_ref:n #1
779 {
780   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
781   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
782   {
783     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
784     {
785       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
786     }
787     {
788       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
789       \tl_set:Ne \l__enumext_renew_counter_vii_tl
790       {
791         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
792       }
793     }
794   }
795   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
796   {
797     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
798     {
799       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
800     }
801     {
802       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
803       \tl_set:Ne \l__enumext_renew_counter_viii_tl
804       {
805         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
806       }
807     }
808   }
809 }

```

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

```

810 \cs_new_protected:Nn \l__enumext_starred_ref:
811 {
812   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
813   {
814     \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
815     {
816       \tl_use:N \l__enumext_renew_counter_vii_tl
817     }
818   }
819   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
820   {

```

```

821         \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
822         {
823             \tl_use:N \l__enumext_renew_counter_viii_tl
824         }
825     }
826 }

```

(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
827 \keys_define:nn { enumext / keyans }
828 {
829     label .code:n = {
830         \__enumext_label_style:cnv { \__enumext_label_v_tl }
831         { \__enumext_counter_v_tl } {#1}
832         \__enumext_label_style:cnv { \__enumext_label_vi_tl }
833         { \__enumext_counter_vi_tl } {#1}
834         \dim_set_eq:NN
835         \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
836     },
837     label .initial:n = \Alph*,
838     label .value_required:n = true,
839     ref .code:n = \__enumext_keyans_ref:n {#1},
840     ref .value_required:n = true,
841 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
842 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
843 {
844     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
845     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
846     {
847         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
848     }
849     {
850         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
851         \tl_put_right:Ne \l__enumext_renew_counter_v_tl
852         {
853             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
854         }
855     }
856 }

```

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

```

857 \cs_new_protected:Nn \__enumext_keyans_ref:
858 {
859     \tl_if_empty:NF \l__enumext_renew_counter_v_tl
860     {
861         \tl_use:N \l__enumext_renew_counter_v_tl
862     }
863 }

```

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

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

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

```

\__enumext_start_from:ccn #1: \l__enumext_label_X_tl
\__enumext_start_from:cce #2: \l__enumext_start_X_int
\__enumext_start_from:cce #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” (§13.26). This is necessary to be able to “reset” the *start value* by means of the `start` or `start*` keys.

```

864 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
865 {
866   \bool_set_false:N \l__enumext_resume_count_bool
867   \__enumext_if_is_int:nTF { #3 }
868   {
869     \int_set:Nn #2 {#3}
870   }
871   {
872     \regex_if_match:nVT { \c{Alpha} | \c{alpha} } {#1}
873     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
874     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#1}
875     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
876   }
877 }
878 \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 `\Alpha`, `\alpha`, `\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.

```

879 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
880 {
881   \__enumext_if_is_int:nTF {#4}
882   {
883     \setcounter{enumX#1} { #4 }
884   }
885   {
886     \regex_if_match:nVT { \c{Alpha} | \c{alpha} } {#2}
887     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
888     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#2}
889     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
890   }
891   \__enumext_label_width_by_box:cv
892   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
893 }
894 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

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

`widest`

```

895 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
896 {
897   \keys_define:nn { enumext / #1 }
898   {
899     start* .code:n = {
900       \__enumext_start_from:ccn
901       { \l__enumext_label_#2_tl }
902       { \l__enumext_start_#2_int } {##1}
903     },
904     start* .value_required:n = true,
905     start .code:n = {
906       \__enumext_start_from:cce
907       { \l__enumext_label_#2_tl }
908       { \l__enumext_start_#2_int } { \int_eval:n {##1} }
909     },
910     start .initial:n = 1,
911     start .value_required:n = true,
912     widest .code:n = {

```

```

913             \__enumext_widest_from:nccn {#2}
914             { \__enumext_label_#2_tl }
915             { \__enumext_labelwidth_#2_dim } {##1}
916         },
917         widest .value_required:n = true,
918     }
919 }
920 \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 penalty

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.

```

921 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
922 {
923     \keys_define:nn { enumext / #1 }
924     {
925         beginpenalty .int_set:c = { \__enumext_beginparpenalty_#2_int },
926         beginpenalty .initial:n = -51,
927         beginpenalty .value_required:n = true,
928         midpenalty .int_set:c = { \__enumext_itempenalty_#2_int },
929         midpenalty .initial:n = -51,
930         midpenalty .value_required:n = true,
931         endpenalty .int_set:c = { \__enumext_endparpenalty_#2_int },
932         endpenalty .initial:n = -51,
933         endpenalty .value_required:n = true,
934     }
935 }
936 \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.

```

937 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
938 {
939     \keys_define:nn { enumext / #1 }
940     {
941         topsep .skip_set:c = { \__enumext_topsep_#2_skip },
942         topsep .initial:n = {#3},
943         topsep .value_required:n = true,
944         partopsep .skip_set:c = { \__enumext_partopsep_#2_skip },
945         partopsep .initial:n = {#4},
946         partopsep .value_required:n = true,
947         parsep .skip_set:c = { \__enumext_parsep_#2_skip },
948         parsep .initial:n = {#5},
949         parsep .value_required:n = true,
950         itemsep .skip_set:c = { \__enumext_itemsep_#2_skip },
951         itemsep .initial:n = {#6},
952         itemsep .value_required:n = true,
953         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
954         noitemsep .value_forbidden:n = true,
955         nosepp .meta:n = {
956             itemsep = 0pt, parsep = 0pt,
957             topsep = 0pt, partopsep = 0pt,
958         },
959         nosepp .value_forbidden:n = true,
960     }
961 }

```

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

```

962 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
963 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
964 { 4.0pt plus 2.0pt minus 1.0pt }
965 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
966 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
967 { 2.0pt plus 1.0pt minus 1.0pt }

```

```

968 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
969 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
970 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
971 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
972 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
973 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
974 { 2.0pt plus 1.0pt minus 1.0pt }
975 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
976 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
977 { 4.0pt plus 2.0pt minus 1.0pt }
978 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
979 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
980 { 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` We define the key `base-fix` only for the “first level” of `enumext` environment.

```

\__enumext_nested_base_line_fix:
981 \keys_define:nn { enumext / level-1 }
982 {
983   base-fix .bool_set:N = \__enumext_base_line_fix_bool,
984   base-fix .initial:n = false,
985   base-fix .value_forbidden:n = true,
986 }

```

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 `__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `__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 `__enumext_base_line_fix_bool` to false.

```

987 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
988 {
989   \bool_lazy_all:nT
990   {
991     { \bool_if_p:N \__enumext_starred_first_bool }
992     { \bool_if_p:N \__enumext_base_line_fix_bool }
993     { \bool_not_p:n { \__enumext_print_keyans_star_bool } }
994   }
995   {
996     \mode_leave_vertical:
997     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
998     \keys_set:nn { enumext / level-1 }
999     {
1000       topsep = 0pt, above = 0pt, above* = 0pt,
1001     }
1002   }

```

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

```

1003 \bool_lazy_and:nnT
1004 { \bool_if_p:N \__enumext_starred_first_bool }
1005 { \bool_if_p:N \__enumext_print_keyans_star_bool }
1006 {
1007   \mode_leave_vertical:
1008   \skip_vertical:n { -\baselineskip }
1009   \skip_vertical:N \c_zero_skip
1010   \keys_set:nn { enumext / level-1 }
1011   {
1012     topsep = 0pt, above = 0pt, above* = 0pt,

```

```

1013     }
1014   }
1015   \bool_set_false:N \__enumext_base_line_fix_bool
1016 }

```

(End of definition for `base-fix` and `__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.

```

1017 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1018 {
1019   \keys_define:nn { enumext / #1 }
1020   {
1021     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1022     itemindent .value_required:n = true,
1023     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1024     rightmargin .value_required:n = true,
1025     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1026     listparindent .value_required:n = true,
1027     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1028     list-offset .value_required:n = true,
1029     list-indent .code:n =
1030       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1031       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1032     list-indent .value_required:n = true,
1033   }
1034 }
1035 \clist_map_inline:nn
1036 {
1037   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1038 }
1039 { \__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.

```

1040 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1041 {
1042   \keys_define:nn { enumext / #1 }
1043   {
1044     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1045     itemindent .value_required:n = true,
1046     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1047     rightmargin .value_required:n = true,
1048     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1049     listparindent .value_required:n = true,
1050     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1051     list-offset .value_required:n = true,
1052     list-indent .meta:n = { list-offset = ##1 },
1053     list-indent .value_required:n = true,
1054   }
1055 }
1056 \clist_map_inline:nn
1057 {
1058   {enumext*}{vii}, {keyans*}{viii}
1059 }
1060 { \__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 T_EX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

1061 \cs_set_protected:Nn \__enumext_fake_item_indent:
1062 {
1063   \dim_compare:nNt
1064     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1065     >
1066     { \c_zero_dim }

```



```

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

```

(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.

```

1114 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1115 {
1116     \keys_define:nn { enumext / #1 }
1117     {
1118         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1119         show-length .initial:n = false,
1120     }
1121 }
1122 \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

before
before*
after
first

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

```

1123 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1124 {
1125   \keys_define:nn { enumext / #1 }
1126   {
1127     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
1128     before .value_required:n = true,
1129     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1130     before* .value_required:n = true,
1131     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1132     after .value_required:n = true,
1133     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1134     first .value_required:n = true,
1135   }
1136 }
1137 \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}`.

```

1138 \cs_new_protected:Nn \__enumext_before_args_exec:
1139 {
1140   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1141 }

```

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}`

```

1142 \cs_new_protected:Nn \__enumext_before_keys_exec:
1143 {
1144   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1145 }

```

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

```

1146 \cs_new_protected:Nn \__enumext_after_stop_list:
1147 {
1148   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1149 }

```

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.`

```

1150 \cs_new_protected:Nn \__enumext_after_args_exec:
1151 {
1152   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1153 }

```

(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:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
1154 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1155 {
1156   \tl_use:N \l__enumext_before_starred_key_v_tl
1157 }
1158 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1159 {
1160   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1161 }
1162 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1163 {
1164   \tl_use:N \l__enumext_after_stop_list_v_tl
1165 }
1166 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1167 {
1168   \tl_use:N \l__enumext_after_list_args_v_tl
1169 }

```

(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*

```
\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
```

Same implementation as the one used in the [enumext](#) environment.

```
1170 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1171 {
1172   \tl_use:N \l__enumext_before_starred_key_vii_tl
1173 }
1174 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1175 {
1176   \tl_use:N \l__enumext_before_starred_key_viii_tl
1177 }
1178 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1179 {
1180   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1181 }
1182 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1183 {
1184   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1185 }
1186 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1187 {
1188   \tl_use:N \l__enumext_after_stop_list_vii_tl
1189 }
1190 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1191 {
1192   \tl_use:N \l__enumext_after_stop_list_viii_tl
1193 }
1194 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1195 {
1196   \tl_use:N \l__enumext_after_list_args_vii_tl
1197 }
1198 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1199 {
1200   \tl_use:N \l__enumext_after_list_args_viii_tl
1201 }
```

(End of definition for __enumext_before_args_exec_vii: and others.)

13.22 Setting keys for multicol and minipage

```
mini-env
mini-sep
columns-sep
columns
```

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.

```
1202 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1203 {
1204   \keys_define:nn { enumext / #1 }
1205   {
1206     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1207     mini-env .value_required:n = true,
1208     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1209     mini-sep .initial:n = 0.3333em,
1210     mini-sep .value_required:n = true,
1211     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1212     columns-sep .value_required:n = true,
1213     columns .int_set:c = { l__enumext_columns_#2_int },
1214     columns .initial:n = 1,
1215     columns .value_required:n = true,
1216   }
1217 }
1218 \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.

```
1219 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1220 {
1221   \keys_define:nn { enumext / #1 }
1222   {
1223     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1224     mini-right .value_required:n = true,
1225     mini-right* .code:n = {
```

```

1226         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1227         \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1228     },
1229     mini-right* .value_required:n = true,
1230 }
1231 }
1232 \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 multicol

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



Figure 7: Representation of the vertical space in `multicol` 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 `multicol` 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 of 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 multicol 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 `multicol` environment in `enumext`.

We will set the default values taking into account that \TeX 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_multicol_above_X_skip` and `\l__enumext_multicol_below_X_skip` equal to the value of `\topsep` in the *current level*.

```

1233 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1234 {
1235     \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: _skip }
1236     {
1237         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1238     }
1239     \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: _skip }
1240     {
1241         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1242     }
1243     \__enumext_add_pre_parsep:
1244 }

```

(End of definition for `__enumext_multi_set_vskip:`)

`__enumext_add_pre_parsep:` The function `__enumext_add_pre_parsep:` “*adjusted*” the value of `\l__enumext_multicol_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*.

```

1245 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1246 {
1247     \int_case:nn { \l__enumext_level_int }
1248     {
1249         { 2 }{
1250             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1251             {
1252                 \skip_add:Nn \l__enumext_multicol_above_ii_skip
1253                 {
1254                     \l__enumext_parsep_i_skip
1255                 }
1256             }
1257         }

```

```

1258     { 3 }{
1259         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1260         {
1261             \skip_add:Nn \l__enumext_multicols_above_iii_skip
1262             {
1263                 \l__enumext_parsep_ii_skip
1264             }
1265         }
1266     }
1267     { 4 }{
1268         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1269         {
1270             \skip_add:Nn \l__enumext_multicols_above_iv_skip
1271             {
1272                 \l__enumext_parsep_iii_skip
1273             }
1274         }
1275     }
1276 }
1277 }

```

(End of definition for `\l__enumext_add_pre_parsep:`)

`\l__enumext_multi_addvspace:` The function `\l__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*.

```

1278 \cs_new_protected:Nn \l__enumext_multi_addvspace:
1279 {
1280     \l__enumext_multi_set_vskip:
1281     \mode_if_vertical:T
1282     {
1283         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1284         {
1285             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1286         }
1287         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1288         {
1289             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1290         }
1291     }
1292     \par\nopagebreak
1293     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1294 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

13.23.2 Adjustment of vertical spaces for multicols in keyans

`\l__enumext_keyans_multi_set_vskip:` The function `\l__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`.

`\l__enumext_keyans_multi_addvspace:`

```

1295 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
1296 {
1297     \skip_set:Nn \l__enumext_multicols_above_v_skip
1298     {
1299         \l__enumext_topsep_v_skip
1300     }
1301     \skip_set:Nn \l__enumext_multicols_below_v_skip
1302     {
1303         \l__enumext_topsep_v_skip
1304     }
1305 }
1306 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1307 {
1308     \l__enumext_keyans_multi_set_vskip:
1309     \mode_if_vertical:T
1310     {
1311         \skip_add:Nn \l__enumext_multicols_above_v_skip
1312         {
1313             \skip_use:N \l__enumext_partopsep_v_skip
1314         }
1315         \skip_add:Nn \l__enumext_multicols_below_v_skip

```

```

1316         {
1317             \skip_use:N \l__enumext_partopsep_v_skip
1318         }
1319     }
1320     \par\nopagebreak
1321     \addvspace{ \l__enumext_multicols_above_v_skip }
1322 }

```

(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 \langle horizontal mode \rangle or \langle vertical mode \rangle . 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` it 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 \langle vertical mode \rangle and we will add `\partopsep`, followed by that we set the value of `\l__enumext_minipage_after_skip`.

```

1323 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1324 {
1325     \skip_set:Nn \l__enumext_minipage_right_skip
1326     {
1327         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1328     }
1329     \mode_if_vertical:T
1330     {
1331         \skip_add:Nn \l__enumext_minipage_right_skip
1332         {
1333             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1334         }
1335     }
1336     \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:`.

```

1337     \skip_set_eq:cN
1338     { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1339     \skip_set_eq:cN
1340     { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1341     \__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`.

```

1342     \int_compare:nNtT
1343     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1344     {
1345         \skip_zero:N \topskip

```



```

1346     \skip_set_eq:Nc \multicolsep { \__enumext_multicols_above_ \__enumext_level: _skip }
1347   }
1348 }

```

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 T_EX is in *horizontal mode* or *vertical mode*. Here we use the plain T_EX 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*.

```

1349 \cs_new_protected:Nn \__enumext_minipage_add_space:
1350 {
1351   \__enumext_minipage_set_skip:
1352   \__enumext_unskip_unkern:
1353   \mode_if_vertical:TF
1354   {
1355     \nopagebreak\nointerlineskip
1356   }
1357   {
1358     \par\nopagebreak\nointerlineskip
1359     \skip_zero:c { \__enumext_partopsep_ \__enumext_level: _skip }
1360   }
1361   \int_compare:nNnTF
1362   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
1363   {
1364     \addvspace{ 0.445\box_ht:N \strutbox }
1365   }
1366   {
1367     \addvspace{ 0.250\box_ht:N \strutbox }
1368   }
1369 }

```

(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.

```

1370 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1371 {
1372   \int_case:nn { \__enumext_level_int }
1373   {
1374     { 2 }{
1375       \skip_if_eq:nnTF
1376       { \__enumext_itemsep_i_skip } { \__enumext_minipage_after_skip }
1377       {
1378         \skip_set:Nn \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1379         \skip_set:Nn \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1380       }
1381       {
1382         \dim_compare:nNnT
1383         { \__enumext_itemsep_i_skip } < { \__enumext_minipage_after_skip }
1384         {
1385           \skip_sub:Nn
1386           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1387           \skip_sub:Nn
1388           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }
1389           \skip_add:Nn
1390           \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1391           \skip_add:Nn
1392           \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1393         }
1394         \dim_compare:nNnT
1395         { \__enumext_itemsep_i_skip } > { \__enumext_minipage_after_skip }
1396         {
1397           \skip_set:Nn \__enumext_minipage_temp_skip
1398           {
1399             \__enumext_itemsep_i_skip - \__enumext_minipage_after_skip
1400           }
1401           \skip_sub:Nn
1402           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1403           \skip_sub:Nn
1404           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }

```

```

1405         \skip_add:Nn
1406         \l__enumext_minipage_after_skip
1407         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1408     \skip_add:Nn
1409     \l__enumext_multicols_below_ii_skip
1410     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1411 }
1412 }
1413 }
1414 { 3 }{
1415     \skip_if_eq:nnTF
1416     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1417     {
1418         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1419         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1420     }
1421     {
1422         \dim_compare:nNnT
1423         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1424         {
1425             \skip_sub:Nn
1426             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1427             \skip_sub:Nn
1428             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1429             \skip_add:Nn
1430             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1431             \skip_add:Nn
1432             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1433         }
1434         \dim_compare:nNnT
1435         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1436         {
1437             \skip_set:Nn \l__enumext_minipage_temp_skip
1438             {
1439                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1440             }
1441             \skip_sub:Nn
1442             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1443             \skip_sub:Nn
1444             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1445             \skip_add:Nn
1446             \l__enumext_minipage_after_skip
1447             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1448             \skip_add:Nn
1449             \l__enumext_multicols_below_iii_skip
1450             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1451         }
1452     }
1453 }
1454 { 4 }{
1455     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1456     {
1457         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1458         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1459     }
1460     {
1461         \dim_compare:nNnT
1462         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1463         {
1464             \skip_sub:Nn
1465             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1466             \skip_sub:Nn
1467             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1468             \skip_add:Nn
1469             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1470             \skip_add:Nn
1471             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1472         }
1473         \dim_compare:nNnT
1474         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1475         {

```

```

1476         \skip_set:Nn \l__enumext_minipage_temp_skip
1477         {
1478             \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1479         }
1480     \skip_sub:Nn
1481     \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1482     \skip_sub:Nn
1483     \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1484     \skip_add:Nn
1485     \l__enumext_minipage_after_skip
1486     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1487     \skip_add:Nn
1488     \l__enumext_multicols_below_iv_skip
1489     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1490 }
1491 }
1492 }
1493 }
1494 }

```

(End of definition for `\l__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`.

```

1495 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1496 {
1497     \skip_zero:N \l__enumext_minipage_after_skip
1498     \skip_zero:N \l__enumext_minipage_left_skip
1499     \skip_zero:N \l__enumext_minipage_right_skip
1500     \skip_set:Nn \l__enumext_minipage_right_skip
1501     {
1502         \l__enumext_topsep_v_skip
1503     }
1504     \mode_if_vertical:T
1505     {
1506         \skip_add:Nn \l__enumext_minipage_right_skip
1507         {
1508             \l__enumext_partopsep_v_skip
1509         }
1510     }
1511     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1512     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1513     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1514     \__enumext_keyans_pre_itemsep_skip:
1515     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1516     {
1517         \skip_zero:N \topskip
1518         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1519     }
1520 }
1521 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1522 {
1523     \__enumext_keyans_minipage_set_skip:
1524     \__enumext_unskip_unkern:
1525     \mode_if_vertical:TF
1526     {
1527         \nopagebreak\nointerlineskip
1528     }
1529     {
1530         \par\nopagebreak\nointerlineskip
1531         \skip_zero:N \l__enumext_partopsep_v_skip
1532     }
1533     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1534     {
1535         \addvspace{ 0.445\box_ht:N \strutbox }
1536     }
1537     {
1538         \addvspace{ 0.250\box_ht:N \strutbox }
1539     }

```

```

1540 }
1541 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1542 {
1543   \skip_if_eq:nnTF
1544   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1545   {
1546     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1547     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1548   }
1549   {
1550     \dim_compare:nNnT
1551     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1552     {
1553       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1554       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1555       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1556       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1557     }
1558     \dim_compare:nNnT
1559     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1560     {
1561       \skip_set:Nn \l__enumext_minipage_temp_skip
1562       {
1563         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1564       }
1565       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1566       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1567       \skip_add:Nn \l__enumext_minipage_after_skip
1568       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1569       \skip_add:Nn \l__enumext_multicols_below_v_skip
1570       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1571     }
1572   }
1573 }

```

(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*`.

```

1574 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1575 {
1576   \skip_zero_new:N \l__enumext_minipage_left_skip
1577   \skip_gzero_new:N \g__enumext_minipage_right_skip
1578   \skip_gzero_new:N \g__enumext_minipage_after_skip
1579   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1580   {
1581     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1582     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1583   }
1584   {
1585     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1586     \skip_gset:Nn \g__enumext_minipage_right_skip
1587     {
1588       \l__enumext_topsep_vii_skip
1589     }
1590     \skip_gset:Nn \g__enumext_minipage_after_skip
1591     {
1592       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1593     }
1594   }
1595 }
1596 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1597 {
1598   \skip_zero_new:N \l__enumext_minipage_after_skip
1599   \skip_zero_new:N \l__enumext_minipage_left_skip
1600   \skip_zero_new:N \l__enumext_minipage_right_skip
1601   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1602   {

```

```

1603     \skip_set:Nn \l__enumext_minipage_left_skip
1604     {
1605         0.5\box_dp:N \strutbox
1606     }
1607     \skip_set:Nn \l__enumext_minipage_right_skip
1608     {
1609         \l__enumext_partopsep_viii_skip
1610     }
1611     \skip_set:Nn \l__enumext_minipage_after_skip
1612     {
1613         1.6\box_dp:N \strutbox
1614     }
1615 }
1616 {
1617     \skip_set:Nn \l__enumext_minipage_left_skip
1618     {
1619         0.5875\box_dp:N \strutbox
1620     }
1621     \skip_set:Nn \l__enumext_minipage_right_skip
1622     {
1623         \l__enumext_topsep_viii_skip
1624     }
1625     \skip_set:Nn \l__enumext_minipage_after_skip
1626     {
1627         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1628     }
1629 }
1630 }

```

(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.

```

1631 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1632 {
1633     \__enumext_mini_set_vskip_vii:
1634     \par\nopagebreak
1635     \addvspace { \l__enumext_minipage_left_skip }
1636 }
1637 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1638 {
1639     \__enumext_mini_set_vskip_viii:
1640     \par\nopagebreak
1641     \addvspace { \l__enumext_minipage_left_skip }
1642 }

```

(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 \TeX 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.

```

1643 \NewDocumentCommand \miniright { s }
1644 {
1645     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1646     {
1647         \msg_error:nnn { enumext } { wrong-miniright-place }
1648     }
1649     % outside
1650     \bool_lazy_and:nnT
1651     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }

```

```

1652     { \int_compare:p:nNn { \l__enumext_level_h_int } = { 0 } }
1653     {
1654         \msg_error:nnn { enumext } { wrong-miniright-place }
1655     }
1656     % starred env
1657     \bool_lazy_and:nnT
1658     { \bool_if_p:N \g__enumext_starred_bool }
1659     { \bool_not_p:n { \l__enumext_standar_bool } }
1660     {
1661         \msg_error:nnn { enumext } { wrong-miniright-starred }
1662     }
1663     % exec
1664     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1665     {
1666         \__enumext_keyans_mini_right_cmd:n {#1}
1667     }
1668     { \__enumext_mini_right_cmd:n {#1} }
1669 }

```

(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 `\l__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 `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1670 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1671 {
1672     \dim_compare:nNnTF
1673     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1674     {
1675         \__enumext_multicols_stop:
1676         \int_compare:nNnT
1677         { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1678         {
1679             \par\addvspace{ \l__enumext_minipage_after_skip }
1680         }
1681         \end__enumext_mini_page
1682         \hfill
1683         \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1684         \par\nointerlineskip
1685         \addvspace { \l__enumext_minipage_right_skip }
1686         \bool_if:nF {#1}
1687         {
1688             \centering
1689         }
1690         \int_gzero:N \g__enumext_minipage_stat_int
1691     }
1692     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1693     % paranoia
1694     \RenewDocumentCommand \miniright { s }
1695     {
1696         \msg_error:nn { enumext } { many-miniright-used }
1697     }
1698 }

```

(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.

```

1699 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1700 {
1701     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1702     {
1703         \__enumext_keyans_multicols_stop:
1704         \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1705         {
1706             \par\addvspace{ \l__enumext_minipage_after_skip }
1707         }

```



```

1708     \end__enumext_mini_page
1709     \hfill
1710     \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1711     \par\nointerlineskip
1712     \addvspace { \__enumext_minipage_right_skip }
1713     \bool_if:nF {#1}
1714     {
1715         \centering
1716     }
1717     \int_gzero:N \g__enumext_minipage_stat_int
1718 }
1719 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1720 % paranoia
1721 \RenewDocumentCommand \miniright { s }
1722 {
1723     \msg_error:nn { enumext } { many-miniright-used }
1724 }
1725 }

```

(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.

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

```

1726 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1727 {
1728     \keys_define:nn { enumext / #1 }
1729     {
1730         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1731         above .value_required:n = true,
1732         above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1733             \keys_set:nn { enumext / #1 } { above = {##1} },
1734         above* .value_required:n = true,
1735         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1736         below .value_required:n = true,
1737         below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1738             \keys_set:nn { enumext / #1 } { below = {##1} },
1739         below* .value_required:n = true,
1740     }
1741 }
1742 \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.

```

1743 \cs_new_protected:Nn \__enumext_vspace_above:
1744 {
1745     \skip_if_eq:nnF
1746     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1747     {
1748         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1749         {
1750             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1751         }
1752         {
1753             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1754         }
1755     }
1756 }

```

(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.

```

1757 \cs_new_protected:Nn \__enumext_vspace_below:
1758 {
1759   \skip_if_eq:nnF
1760     { \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1761   {
1762     \bool_if:cTF { \l__enumext_vspace_b_star_ \__enumext_level: _bool }
1763     {
1764       \vspace*{ \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } }
1765     }
1766     {
1767       \vspace { \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } }
1768     }
1769   }
1770 }

```

(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.

```

1771 \cs_new_protected:Nn \__enumext_vspace_above_v:
1772 {
1773   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1774   {
1775     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1776     {
1777       \vspace*{ \l__enumext_vspace_above_v_skip }
1778     }
1779     { \vspace { \l__enumext_vspace_above_v_skip } }
1780   }
1781 }

```

(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.

```

1782 \cs_new_protected:Nn \__enumext_vspace_below_v:
1783 {
1784   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1785   {
1786     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1787     {
1788       \vspace*{ \l__enumext_vspace_below_v_skip }
1789     }
1790     { \vspace { \l__enumext_vspace_below_v_skip } }
1791   }
1792 }

```

(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`:

```

1793 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1794 {
1795   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1796   {
1797     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1798     {
1799       \vspace*{ \l__enumext_vspace_above_vii_skip }
1800     }
1801     { \vspace { \l__enumext_vspace_above_vii_skip } }
1802   }
1803 }
1804 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1805 {
1806   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1807   {

```

```

1808         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1809         {
1810             \vspace*{ \l__enumext_vspace_above_viii_skip }
1811         }
1812         { \vspace { \l__enumext_vspace_above_viii_skip } }
1813     }
1814 }

```

(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:`

```

1815 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1816 {
1817     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1818     {
1819         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1820         {
1821             \vspace*{ \l__enumext_vspace_below_vii_skip }
1822         }
1823         { \vspace { \l__enumext_vspace_below_vii_skip } }
1824     }
1825 }
1826 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1827 {
1828     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1829     {
1830         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1831         {
1832             \vspace*{ \l__enumext_vspace_below_viii_skip }
1833         }
1834         { \vspace { \l__enumext_vspace_below_viii_skip } }
1835     }
1836 }

```

(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>`.

`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 *start value* of the environments would be increased when using `resume` or `resume*` keys.

```

1837 \cs_set_protected:Npn \__enumext_tmp:n #1
1838 {
1839     \keys_define:nn { enumext / #1 }
1840     {
1841         series .str_set:N = \l__enumext_series_name_str,
1842         series .value_required:n = true,
1843         resume* .code:n = {
1844             \bool_if:NF \l__enumext_print_keyans_cmd_bool
1845             {
1846                 \__enumext_resume_star:
1847             }
1848         },
1849         resume* .value_forbidden:n = true,
1850     }
1851 }
1852 \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*`.

```

1853 \cs_set_protected:Npn \__enumext_tmp:n #1
1854 {
1855     \keys_define:nn { enumext / #1 }
1856     {
1857         resume .code:n = {

```

```

1858         \bool_if:NF \l__enumext_print_keyans_cmd_bool
1859         {
1860             \tl_set:Nn \l__enumext_series_name_tl {##1}
1861             \tl_if_empty:NTF \l__enumext_series_name_tl
1862             {
1863                 \bool_set_true:N \l__enumext_resume_count_bool
1864             }
1865             {
1866                 \__enumext_resume:n {##1}
1867             }
1868         }
1869     },
1870 }
1871 }
1872 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
1873 \keys_define:nn { enumext / enumext* }
1874 {
1875     resume .code:n = {
1876         \bool_if:NF \l__enumext_print_keyans_cmd_bool
1877         {
1878             \tl_set:Nn \l__enumext_series_name_tl {##1}
1879             \tl_if_empty:NTF \l__enumext_series_name_tl
1880             {
1881                 \bool_set_true:N \l__enumext_resume_count_bool
1882             }
1883             {
1884                 \__enumext_resume_vii:n {#1}
1885             }
1886         }
1887     },
1888 }

```

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

13.26.1 Internal function to save counter and integer values

```

\__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).

```

1889 \cs_new_protected:Nn \__enumext_standar_save_counter:
1890 {
1891     \bool_if:NTF \g__enumext_standar_bool
1892     {
1893         \__enumext_standar_save_counter_aux:
1894         \int_compare:nNnT { \l__enumext_level_int } = { 1 }
1895         {
1896             \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1897             {
1898                 \int_gset_eq:cN
1899                 { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1900             }
1901         }
1902     }
1903     {
1904         \__enumext_standar_save_counter_aux:
1905     }
1906 }
1907 \cs_new_protected:Nn \__enumext_standar_save_counter_aux:
1908 {
1909     \str_if_empty:NF \l__enumext_series_name_str
1910     {
1911         \int_gset_eq:cc
1912         { g__enumext_series_ \l__enumext_series_name_str _ \l__enumext_level: _int }

```

```

1913         { c@enumX \__enumext_level: }
1914     }
1915     \tl_if_empty:NTF \l__enumext_series_name_tl
1916     {
1917         \str_if_empty:NT \l__enumext_series_name_str
1918         {
1919             \tl_if_empty:NT \l__enumext_store_name_tl
1920             {
1921                 \int_gset_eq:cc
1922                 { c@ __enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
1923             }
1924         }
1925     }
1926     {
1927         \int_if_exist:cT
1928         { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1929         {
1930             \int_gset_eq:cc
1931             { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1932             { c@enumX \__enumext_level: }
1933         }
1934     }
1935 }
1936 \cs_new_protected:Nn \__enumext_starred_save_counter:
1937 {
1938     \bool_if:NTF \g__enumext_starred_bool
1939     {
1940         \__enumext_starred_save_counter_aux:
1941         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1942         {
1943             \int_gset_eq:cN
1944             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1945         }
1946     }
1947     {
1948         \__enumext_starred_save_counter_aux:
1949     }
1950 }
1951 \cs_new_protected:Nn \__enumext_starred_save_counter_aux:
1952 {
1953     \str_if_empty:NF \l__enumext_series_name_str
1954     {
1955         \int_gset_eq:cN
1956         { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
1957     }
1958     \tl_if_empty:NTF \l__enumext_series_name_tl
1959     {
1960         \str_if_empty:NT \l__enumext_series_name_str
1961         {
1962             \tl_if_empty:NT \l__enumext_store_name_tl
1963             {
1964                 \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
1965             }
1966         }
1967     }
1968     {
1969         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
1970         {
1971             \int_gset_eq:cN
1972             { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
1973         }
1974     }
1975 }

```

(End of definition for __enumext_standar_save_counter: and others.)

13.26.2 Internal function for resume counters

__enumext_resume_counter: The __enumext_resume_counter: function is executed by the `resume*` key and `resume` key *without value*, only the “counters” for the “levels” of the environments in which it is executed will be set. If the `save-ans` key is active it will set the “counter” according to the value of the integer variable created by that key.

```

1976 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1977 {
1978     \cs_set:Npn \__enumext_tmp:n ##1
1979     {
1980         \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
1981         {
1982             \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
1983         }
1984     }
1985     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
1986     {
1987         \bool_lazy_and:nnTF
1988         { \bool_if_p:N \l__enumext_standar_first_bool }
1989         { \bool_if_p:N \l__enumext_store_active_bool }
1990         {
1991             \int_set:Nn \l__enumext_start_i_int
1992             {
1993                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1994             }
1995             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
1996         }
1997     }
1998     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
1999 }
2000 }
2001 \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2002 {
2003     \bool_lazy_and:nnTF
2004     { \bool_if_p:N \l__enumext_starred_first_bool }
2005     { \bool_if_p:N \l__enumext_store_active_bool }
2006     {
2007         \int_set:Nn \l__enumext_start_vii_int
2008         {
2009             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2010         }
2011     }
2012     {
2013         \int_set:Nn \l__enumext_start_vii_int
2014         {
2015             \int_use:c { c@ __enumext_resume_vii_int } + 1
2016         }
2017     }
2018 }
2019 }

```

(End of definition for `__enumext_resume_counter:.`)

13.26.3 Internal functions for series key

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn

```

The function `__enumext_filter_series:n` will be in charge of filtering the *⟨keys⟩* we want to store where *#1* represents the *optional argument* passed to the environment. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in *chat-TeX-SX*

```

2020 \cs_new:Npn \__enumext_filter_series:n #1
2021 {
2022     \use:e
2023     {
2024         \keyval_parse:NNn
2025         \__enumext_filter_series_key:n
2026         \__enumext_filter_series_pair:nn {#1}
2027     }
2028 }

```

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.

```

2029 \cs_new:Npn \__enumext_filter_series_key:n #1
2030 {
2031     \str_case:nnF {#1}
2032     {
2033         { resume } {} { resume* } {} { reset } {} { reset* } {} { base-fix } {}
2034     }
2035     { , { \exp_not:n {#1} } }
2036 }

```


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.

```

2037 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
2038 {
2039   \str_case:nnF {#1}
2040   {
2041     { series } {} { resume } {} { start } {}
2042     { start* } {} { save-ans } {} { save-key } {}
2043   }
2044   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2045 }

```

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

`__enumext_save_last_keys:n` 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.

- The value of boolean variable `\l__enumext_resume_series_X_bool` is set to “true” by the key `resume={⟨series name⟩}`, in this case we must make sure it is set to “false” so that it does not overwrite the default filtered *(keys)*.

```

2046 \cs_new_protected:Npn \__enumext_save_last_keys:n #1
2047 {
2048   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2049   {
2050     \bool_if:cF { l__enumext_resume_series_ \__enumext_level: _bool }
2051     {
2052       \tl_gclear:c { g__enumext_save_last_keys_ \__enumext_level: _tl }
2053       \tl_gset:ce
2054       { g__enumext_save_last_keys_ \__enumext_level: _tl }
2055       { \__enumext_filter_series:n {#1} }
2056     }
2057   }
2058   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2059   {
2060     \bool_if:NF \l__enumext_resume_series_vii_bool
2061     {
2062       \tl_gclear:N g__enumext_save_last_keys_vii_tl
2063       \tl_gset:Ne g__enumext_save_last_keys_vii_tl { \__enumext_filter_series:n {#1} }
2064     }
2065   }
2066 }

```

The `__enumext_resume_last_counter:` function will be in charge of setting the “counters” when the keys `series={⟨series name⟩}` or `resume={⟨series name⟩}` are NOT active and the `resume` key is being used *without value* either in the *optional argument* of the environments or through the `\setenumext` command.

- The value of boolean variable `\l__enumext_resume_count_bool` is set to “true” by the key `resume without value` and set to “false” by the keys `start` and `start*` (§13.15).

```

2067 \cs_new_protected:Npn \__enumext_resume_last_counter:
2068 {
2069   \bool_lazy_and:nnT
2070   { \bool_if_p:N \l__enumext_resume_count_bool }
2071   { \tl_if_empty_p:N \l__enumext_series_name_tl }
2072   {
2073     \__enumext_resume_counter:
2074   }
2075 }

```

(End of definition for `__enumext_save_last_keys:n` and `__enumext_resume_last_counter:`.)

`__enumext_parse_series:n` The `__enumext_parse_series:n` function handled by the `series` key will be responsible for *storing* the *filtered (keys)* from the *optional arguments* of the `enumext` and `enumext*` environments for the `resume` and `resume*` keys. If the `series` key is NOT active it will call the `__enumext_save_last_keys:n` function to *store* the *filtered (keys)* that will be used by the `resume*` key and then the `__enumext_resume_last_counter:` function used by the `resume` key *without value* if it is active, otherwise *store* the *filtered (keys)* in the global variable `\g__enumext_series_⟨series name⟩_X_tl` along with the creation of the integer variable `\g__enumext_series_⟨series name⟩_X_int` used by the `resume` key *with value*.

- This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.42) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.47).

```

2076 \cs_new_protected:Npn \__enumext_parse_series:n #1

```

```

2077 {
2078     \str_if_empty:NTF \l__enumext_series_name_str
2079     {
2080         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2081         {
2082             \__enumext_save_last_keys:n {#1}
2083             \__enumext_resume_last_counter:
2084         }
2085     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2086     {
2087         \__enumext_save_last_keys:n {#1}
2088         \__enumext_resume_last_counter:
2089     }
2090 }
2091 {
2092     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2093     {
2094         \tl_gclear_new:c
2095         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2096         \tl_gset:ce
2097         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2098         { \__enumext_filter_series:n {#1} }
2099         \int_if_exist:cF
2100         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2101         {
2102             \int_new:c
2103             { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2104         }
2105     }
2106     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2107     {
2108         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2109         \tl_gset:ce
2110         { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2111         { \__enumext_filter_series:n {#1} }
2112         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2113         {
2114             \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2115         }
2116     }
2117 }
2118 }

```

(End of definition for __enumext_parse_series:n.)

13.26.4 Internal functions for resume key with value

```

\__enumext_resume:n
\__enumext_resume_vii:n

```

The functions __enumext_resume:n and __enumext_resume_vii:n will handle the *argument* {*series name*} passed to the *resume* key in *enumext* and *enumext** environments. First we will check if the global variable \g__enumext_series_<series name>_X_tl exists, if so we will call the function __enumext_resume_series:n and pass the *keys* stored in \g__enumext_series_<series name>_X_tl to the environments, otherwise we will return an error.

```

2119 \cs_new_protected:Npn \__enumext_resume:n #1
2120 {
2121     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2122     {
2123         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2124         {
2125             \__enumext_resume_series:n {#1}
2126             \exp_args:Ne \keys_set:nv { enumext / level-\int_use:N \l__enumext_level_int }
2127             { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2128         }
2129         {
2130             \msg_error:nnn { enumext } { unknown-series-standar } {#1}
2131         }
2132     }
2133 }
2134 \cs_new_protected:Npn \__enumext_resume_vii:n #1
2135 {
2136     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2137     {
2138         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }

```

```

2139         {
2140             \__enumext_resume_series:n {#1}
2141             \keys_set:nv { enumext / enumext* }
2142             { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2143         }
2144         {
2145             \msg_error:nnn { enumext } { unknown-series-starred } {#1}
2146         }
2147     }
2148 }

```

(End of definition for __enumext_resume:n and __enumext_resume_vii:n.)

```

\__enumext_resume_series:n
\__enumext_resume_integer_series:

```

The function __enumext_resume_series:n will set the variable \l__enumext_resume_series_X_bool to “true” and pass the $\langle argument \rangle$ to the variable \l__enumext_series_name_tl then call the function __enumext_resume_integer_series:.

```

2149 \cs_new_protected:Npn \__enumext_resume_series:n #1
2150 {
2151     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2152     {
2153         \bool_set_true:c { l__enumext_resume_series_ \__enumext_level: _bool }
2154         \tl_clear:N \l__enumext_series_name_tl
2155         \tl_set:Nn \l__enumext_series_name_tl {#1}
2156         \__enumext_resume_integer_series:
2157     }
2158     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2159     {
2160         \bool_set_true:N \l__enumext_resume_series_vii_bool
2161         \tl_clear:N \l__enumext_series_name_tl
2162         \tl_set:Nn \l__enumext_series_name_tl {#1}
2163         \__enumext_resume_integer_series:
2164     }
2165 }

```

The function __enumext_resume_integer_series: will be executed when the resume= $\langle series name \rangle$ key is active, setting the *start value* for the “counter” of the “current level” of the environments in which it is run according to the value of the “integer variables” created by the series key. If the save-ans key is active it will set the *start value* for the “counter” according to the value of the integer variable created by that key.

```

2166 \cs_new_protected:Nn \__enumext_resume_integer_series:
2167 {
2168     \cs_set:Npn \__enumext_tmp:n ##1
2169     {
2170         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2171         {
2172             \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2173             {
2174                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2175             }
2176         }
2177     }
2178     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2179     {
2180         \bool_lazy_and:nnTF
2181         { \bool_if_p:N \l__enumext_standar_first_bool }
2182         { \bool_if_p:N \l__enumext_store_active_bool }
2183         {
2184             \int_set:Nn \l__enumext_start_i_int
2185             {
2186                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2187             }
2188             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2189         }
2190         {
2191             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2192         }
2193     }
2194     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2195     {
2196         \bool_lazy_and:nnTF
2197         { \bool_if_p:N \l__enumext_starred_first_bool }
2198         { \bool_if_p:N \l__enumext_store_active_bool }

```

```

2199         {
2200             \int_set:Nn \l__enumext_start_vii_int
2201             {
2202                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2203             }
2204         }
2205         {
2206             \int_set:Nn \l__enumext_start_vii_int
2207             {
2208                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
2209             }
2210         }
2211     }
2212 }

```

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

13.26.5 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.

```

2213 \cs_new_protected:Nn \__enumext_resume_star:
2214 {
2215     \cs_set:Npn \__enumext_tmp:n ##1
2216     {
2217         \tl_if_empty:cF { g__enumext_save_last_keys_ \int_to_roman:n {##1} _tl }
2218         {
2219             \__enumext_resume_counter:
2220             \exp_args:Ne \keys_set:nv
2221             { enumext / level-\int_use:N \l__enumext_level_int }
2222             { g__enumext_save_last_keys_ \int_to_roman:n {##1} _tl }
2223         }
2224     }
2225     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2226     {
2227         \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2228     }
2229     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2230     {
2231         \tl_if_empty:NF \g__enumext_save_last_keys_vii_tl
2232         {
2233             \__enumext_resume_counter:
2234             \keys_set:nV { enumext / enumext* } \g__enumext_save_last_keys_vii_tl
2235         }
2236     }
2237 }

```

(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>}`.

```

\__enumext_standard_reset:n
\__enumext_starred_reset:n
\__enumext_reset_count_resume:n
\__enumext_reset_count_resume:en
\__enumext_reset_count_resume_all:n
\__enumext_reset_count_resume_levels:n
2238 \NewDocumentCommand \resetenumext { s o m }
2239 {
2240     \bool_if:nTF {#1}
2241     {
2242         \__enumext_reset_count_resume_all:n {#3}
2243     }
2244     {
2245         \tl_if_novalue:nTF {#2}
2246         {
2247             \__enumext_reset_count_resume_levels:n {#3}

```

```

2248     }
2249     {
2250         \str_if_eq:nnTF {#2} { * }
2251         { \__enumext_starred_reset:n {#3} }
2252         {
2253             \bool_lazy_and:nnTF
2254             { \int_compare_p:nNn {#2} > 0 }
2255             { \int_compare_p:nNn {#2} < 5 }
2256             { \__enumext_standard_reset:nn {#2} {#3} }
2257             {
2258                 \msg_error:nne { enumext } { out-of-range } { \int_eval:n {#2} }
2259             }
2260         }
2261     }
2262 }
2263 }
2264 \cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
2265 {
2266     \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } {%#2}
2267 }
2268 \cs_new_protected:Npn \__enumext_starred_reset:n #1
2269 {
2270     \__enumext_reset_count_resume:nn { vii } {#1}
2271 }
2272 \cs_new_protected:Npn \__enumext_reset_count_resume:nn #1 #2
2273 {
2274     \counterwithin*{enumX#1}{#2}
2275     \counterwithin*{\__enumext_resume_#1_int}{#2}
2276 }
2277 \cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
2278 \cs_new_protected:Npn \__enumext_reset_count_resume_all:n #1
2279 {
2280     \clist_map_inline:nn { i,ii,iii,iv,vii }
2281     {
2282         \__enumext_reset_count_resume:nn { ##1 } { #1 }
2283     }
2284 }
2285 \cs_new_protected:Npn \__enumext_reset_count_resume_levels:n #1
2286 {
2287     \clist_map_inline:nn { i,ii,iii,iv }
2288     {
2289         \__enumext_reset_count_resume:nn { ##1 } { #1 }
2290     }
2291 }

```

(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
reset*
2292 \cs_set_protected:Npn \__enumext_tmp:n #1
2293 {
2294     \keys_define:nn { enumext / #1 }
2295     {
2296         reset .code:n = \__enumext_standard_reset_key:,
2297         reset .value_forbidden:n = true,
2298         reset* .code:n = \__enumext_standard_reset_key_star:,
2299         reset* .value_forbidden:n = true,
2300     }
2301 }
2302 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
2303 \keys_define:nn { enumext / enumext* }
2304 {
2305     reset .code:n = \__enumext_starred_reset_key:,
2306     reset .value_forbidden:n = true,
2307     reset* .code:n = \__enumext_starred_reset_key:,
2308     reset* .value_forbidden:n = true,
2309 }

```

(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.

```

2310 \cs_new_protected:Nn \__enumext_standard_reset_key:
2311 {
2312   \int_compare:nNtT { \__enumext_level_int } > { 0 }
2313   {
2314     \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n { \__enumext_level_int } _int }
2315     {
2316       \int_gzero:c { c@__enumext_resume_ \int_to_roman:n { \__enumext_level_int } _int }
2317     }
2318   }
2319 }

```

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*.

```

2320 \cs_new_protected:Nn \__enumext_standard_reset_key_star:
2321 {
2322   \cs_set:Npn \__enumext_tmp:n ##1
2323   {
2324     \int_if_exist:cT { c@__enumext_resume_ \int_to_roman:n {##1} _int }
2325     {
2326       \int_gzero:c { c@__enumext_resume_ \int_to_roman:n {##1} _int }
2327     }
2328   }
2329   \int_compare:nNtT { \__enumext_level_int } > { 0 }
2330   {
2331     \int_step_function:nnN { \__enumext_level_int } { 4 } \__enumext_tmp:n
2332   }
2333 }

```

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.

```

2334 \cs_new_protected:Nn \__enumext_starred_reset_key:
2335 {
2336   \int_gzero:c { c@__enumext_resume_vii_int }
2337 }

```

(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*`.

```

2338 \cs_set_protected:Npn \__enumext_tmp:n #1
2339 {
2340   \keys_define:nn { enumext / #1 }
2341   {
2342     save-ans .code:n = \__enumext_storing_set:n {##1},
2343     save-ans .value_required:n = true,
2344   }
2345 }
2346 \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

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:`.

```

2347 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2348 {
2349   \msg_term:nnVV { enumext } { save-ans-log }

```



```

2350     \g__enumext_envir_name_tl \l__enumext_store_name_tl
2351   }
2352   \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2353   {
2354     \msg_term:nnVV { enumext } { save-ans-log-hook }
2355     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2356   }

```

(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 $\langle store\ name \rangle$ 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.

```

2357   \cs_new_protected:Npn \__enumext_storing_set:n #1
2358   {
2359     \tl_set:Nx \l__enumext_store_name_tl {#1}
2360     \tl_if_empty:NTF \l__enumext_store_name_tl
2361     {
2362       \bool_lazy_or:nnT
2363       { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2364       {
2365         \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2366       }
2367     }
2368     {
2369       \bool_lazy_or:nnT
2370       { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2371       {
2372         \__enumext_start_save_ans_msg:
2373         \__enumext_storing_exec:
2374       }
2375     }
2376   }

```

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 $\langle store\ name \rangle$ into the variable \g__enumext_store_name_tl.

```

2377   \cs_new_protected:Nn \__enumext_storing_exec:
2378   {
2379     \bool_set_true:N \l__enumext_store_active_bool
2380     \bool_set_true:N \l__enumext_check_answers_bool
2381     \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*`.

```

2382   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2383   {
2384     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2385     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2386   }
2387   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2388   {
2389     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2390     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2391   }
2392   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2393   {
2394     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2395     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2396   }
2397   }

```

(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 2398 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   2399 {
            2400   \keys_define:nn { enumext / #1 }
            2401   {
            2402     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
            2403     check-ans .initial:n = false,
            2404     check-ans .value_required:n = true,
            2405     no-store .code:n = {
            2406               \bool_set_false:N \__enumext_check_answers_bool
            2407               \bool_set_false:N \__enumext_check_ans_key_bool
            2408             },
            2409     no-store .value_forbidden:n = true,
            2410   }
            2411 }
            2412 \clist_map_inline:nn
            2413 {
            2414   level-1, level-2, level-3, level-4, enumext*
            2415 }
            2416 { \__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.

```

2417 \cs_new_protected:Nn \__enumext_check_ans_active:
2418 {
2419   \tl_if_empty:NF \l__enumext_store_name_tl
2420   {
2421     \bool_if:NT \l__enumext_check_answers_bool
2422     {
2423       \__enumext_check_ans_level:
2424     }
2425   }
2426 }
```

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”.

```

2427 \cs_new_protected:Nn \__enumext_check_ans_level:
2428 {
2429   \int_case:nn { \l__enumext_level_int }
2430   {
2431     { 1 }{
2432       \bool_lazy_all:nT
2433       {
2434         { \bool_if_p:N \g__enumext_starred_bool }
2435         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2436       }
2437       {
2438         \int_gdecr:N \g__enumext_item_number_int
2439         \bool_set_false:N \l__enumext_item_number_bool
2440       }
2441     }
2442     { 2 }{
2443       \int_gdecr:N \g__enumext_item_number_int
2444       \bool_set_false:N \l__enumext_item_number_bool
2445     }
2446     { 3 }{
2447       \int_gdecr:N \g__enumext_item_number_int
2448       \bool_set_false:N \l__enumext_item_number_bool
2449     }
2450     { 4 }{
2451       \int_gdecr:N \g__enumext_item_number_int
2452       \bool_set_false:N \l__enumext_item_number_bool
2453     }
2454   }

```

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.

```

2455   \int_case:nn { \l__enumext_level_h_int }
2456   {
2457     { 1 }{
2458       \bool_lazy_all:nT
2459       {
2460         { \bool_if_p:N \g__enumext_standar_bool }
2461         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2462       }
2463       {
2464         \int_gdecr:N \g__enumext_item_number_int
2465         \bool_set_false:N \l__enumext_item_number_bool
2466       }
2467     }
2468   }
2469 }

```

(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.

```

2470 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2471 {
2472   \bool_lazy_and:nnT
2473   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2474   { \bool_if_p:N \g__enumext_standar_bool }
2475   {
2476     \bool_gset_true:N \g__enumext_check_ans_key_bool
2477   }
2478   \bool_lazy_and:nnT
2479   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2480   { \bool_if_p:N \g__enumext_starred_bool }
2481   {
2482     \bool_gset_true:N \g__enumext_check_ans_key_bool
2483   }
2484 }

```

(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:`.

```
2485 \cs_new_protected:Nn \__enumext_item_answer_diff:
2486 {
2487     \int_gset:Nn \g__enumext_item_answer_diff_int
2488     {
2489         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2490     }
2491 }
```

(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:`.

```
2492 \cs_new_protected:Nn \__enumext_check_ans_show:
2493 {
2494     \int_case:nn { \g__enumext_item_answer_diff_int }
2495     {
2496         { -1 } { \__enumext_check_ans_msg_less: }
2497         { 0 } { \__enumext_check_ans_msg_same_ok: }
2498         { 1 } { \__enumext_check_ans_msg_greater: }
2499     }
2500 }
2501 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2502 {
2503     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2504     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2505 }
2506 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2507 {
2508     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2509     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2510 }
2511 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2512 {
2513     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2514     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2515 }
```

(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:`.

```
2516 \cs_new_protected:Nn \__enumext_check_ans_log:
2517 {
2518     \int_case:nn { \g__enumext_item_answer_diff_int }
2519     {
2520         { -1 } { \__enumext_check_ans_log_msg_less: }
2521         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2522         { 1 } { \__enumext_check_ans_log_msg_greater: }
2523     }
2524 }
2525 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2526 {
2527     \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2528     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2529 }
2530 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2531 {
2532     \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2533     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2534 }
```

```

2535 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2536 {
2537   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2538   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2539 }

```

(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.

```

2540 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2541 {
2542   \int_compare:nNnT
2543     { \g__enumext_check_starred_cmd_int } = { 0 }
2544     {
2545       \msg_warning:nnnV
2546         { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2547     }
2548   \int_compare:nNnT
2549     { \g__enumext_check_starred_cmd_int } > { 1 }
2550     {
2551       \msg_warning:nnnV
2552         { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2553     }
2554   \int_gzero:N \g__enumext_check_starred_cmd_int
2555   \tl_clear:N \l__enumext_check_start_line_env_tl
2556 }

```

(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`.

```

2557 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2558 {
2559   \keys_define:nn { enumext / #1 }
2560   {
2561     mark-ans* .tl_set:c = { \l__enumext_mark_answer_sym_#2_tl },
2562     mark-ans* .initial:n = \textasteriskcentered,
2563     mark-ans* .value_required:n = true,
2564     mark-pos* .choice:,
2565     mark-pos* / left .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { l },
2566     mark-pos* / right .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { r },
2567     mark-pos* / center .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { c },
2568     mark-pos* / unknown .code:n =
2569       \msg_error:nneee { enumext } { unknown-choice }
2570       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2571     mark-pos* .initial:n = right,
2572     mark-pos* .value_required:n = true,
2573     mark-sep* .dim_set:c = { \l__enumext_mark_sym_sep_#2_dim },
2574     mark-sep* .value_required:n = true,
2575     wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2576     wrap-ans* .value_required:n = true,
2577     wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2578     wrap-opt .initial:n = [{##1}],
2579     wrap-opt .value_required:n = true,
2580     save-sep .tl_set:c = { \l__enumext_store_keyans_item_opt_sep_#2_tl },
2581     save-sep .initial:n = {,~},
2582     save-sep .value_required:n = true,
2583     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2584     show-ans .initial:n = false,
2585     show-ans .value_required:n = true,
2586     show-pos .bool_set:N = \l__enumext_show_position_bool,
2587     show-pos .initial:n = false,

```

```

2588         show-pos .value_required:n = true,
2589     }
2590 }
2591 \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

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

```

2592 \cs_set_protected:Npn \__enumext_tmp:n #1
2593 {
2594     \keys_define:nn { enumext / #1 }
2595     {
2596         mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2597         mark-ref .initial:n = \textreferencemark,
2598         mark-ref .value_required:n = true,
2599         save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2600         save-ref .initial:n = false,
2601         save-ref .value_required:n = true,
2602         show-ans .bool_set:N = \__enumext_show_answer_bool,
2603         show-ans .initial:n = false,
2604         show-ans .value_required:n = true,
2605         show-pos .bool_set:N = \__enumext_show_position_bool,
2606         show-pos .initial:n = false,
2607         show-pos .value_required:n = true,
2608         mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2609         mark-ans .initial:n = \textasteriskcentered,
2610         mark-ans .value_required:n = true,
2611         mark-sep .dim_set:N = \__enumext_mark_sym_sep_dim,
2612         mark-sep .value_required:n = true,
2613         mark-pos .choice:,
2614         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2615         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2616         mark-pos / center .code:n = \str_set:Nn \__enumext_mark_position_str { c },
2617         mark-pos / unknown .code:n =
2618             \msg_error:nnee { enumext } { unknown-choice }
2619             { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2620         mark-pos .initial:n = right,
2621         mark-pos .value_required:n = true,
2622
2623         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2624         wrap-ans .initial:n =
2625             {
2626                 \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2627             },
2628         wrap-ans .value_required:n = true,
2629         mark-ans* .code:n = {
2630             \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2631             \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2632         },
2633         mark-ans* .value_required:n = true,
2634         mark-pos* .code:n = {
2635             \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2636             \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2637         },
2638         mark-pos* .value_required:n = true,
2639         mark-sep* .code:n = {
2640             \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2641             \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2642         },
2643         mark-sep* .value_required:n = true,
2644         wrap-ans* .code:n = {
2645             \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2646             \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2647         },
2648         wrap-ans* .value_required:n = true,
2649         wrap-opt .code:n = {
2650             \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2651             \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }

```

```

2652         },
2653         wrap-opt .value_required:n = true,
2654         save-sep .code:n = {
2655             \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2656             \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2657         },
2658         save-sep .value_required:n = true,
2659     }
2660 }
2661 \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 `\l__enumext_store_save_key_X_bool` is false, that is, the key *store-key* is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered *keys*.

```

2662 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2663 {
2664     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2665     {
2666         \tl_clear:c { \l__enumext_store_save_key_ \__enumext_level: _tl }
2667         \tl_set:ce
2668             { \l__enumext_store_save_key_ \__enumext_level: _tl }
2669             { \__enumext_filter_save_key:n {#1} }
2670     }
2671 }
2672 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2673 {
2674     \bool_if:NF \l__enumext_store_save_key_vii_bool
2675     {
2676         \tl_clear:N \l__enumext_store_save_key_vii_tl
2677         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2678     }
2679 }

```

(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 `\l__enumext_store_save_key_X_tl` variable set by the functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n`. Now define the key *save-key* for all levels of `enumext` and `enumext*` environments.

```

2680 \cs_set_protected:Npn \__enumext_tmp:n #1
2681 {
2682     \keys_define:nn { enumext / enumext* }
2683     {
2684         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2685         save-key .value_required:n = true,
2686     }
2687     \keys_define:nn { enumext / #1 }
2688     {
2689         save-key .code:n = \__enumext_parse_save_key:n {##1},
2690         save-key .value_required:n = true,
2691     }
2692 }
2693 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for *save-key*.)


```

__enumext_parse_save_key:n
  __enumext_parse_save_key_vii:n

```

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

```

2694 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2695 {
2696   \bool_set_true:c { l__enumext_store_save_key_ __enumext_level: _bool }
2697   \tl_clear:c { l__enumext_save_key_ __enumext_level: _tl }
2698   \tl_set:ce
2699     { l__enumext_store_save_key_ __enumext_level: _tl }
2700     { __enumext_filter_save_key:n {#1} }
2701 }
2702 \cs_new_protected:Npn __enumext_parse_save_key_vii:n #1
2703 {
2704   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2705   \tl_clear:N \l__enumext_store_save_key_vii_tl
2706   \tl_set:Ne \l__enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2707 }

```

(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

```

__enumext_filter_save_key:n
  __enumext_filter_save_key_key:n
  __enumext_filter_save_key_pair:nn

```

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

```

2708 \cs_new:Npn __enumext_filter_save_key:n #1
2709 {
2710   \use:e
2711   {
2712     \keyval_parse:NNn
2713       __enumext_filter_save_key_key:n
2714       __enumext_filter_save_key_pair:nn {#1}
2715   }
2716 }

```

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.

```

2717 \cs_new:Npn __enumext_filter_save_key_key:n #1
2718 {
2719   \str_case:nnF {#1}
2720   {
2721     { resume } {} { resume* } {} { reset } {} { reset* } {} { no-store } {} { base-fix } {}
2722   }
2723   { , { \exp_not:n {#1} } }
2724 }

```

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.

```

2725 \cs_new:Npn __enumext_filter_save_key_pair:nn #1#2
2726 {
2727   \str_case:nnF {#1}
2728   {
2729     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2730     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2731     { mark-ans } {} { mark-pos } {} { mark-sep } {} { wrap-ans } {}
2732     { mark-ans* } {} { mark-pos* } {} { mark-sep* } {} { wrap-ans* } {}
2733     { wrap-opt } {} { save-sep } {} { mark-ref } {} { mini-env } {}
2734     { mini-sep } {} { mini-right } {} { mini-right* } {}
2735   }
2736   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2737 }

```

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

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

```

2738 \cs_new_protected:Npn __enumext_store_addto_prop:n #1

```

```

2739 {
2740   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2741   {
2742     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2743   }
2744   { #1 }
2745 }
2746 \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

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.

```

2747 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2748 {
2749   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2750 }
2751 \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

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.

```

2752 \cs_new_protected:Nn \__enumext_store_level_open:
2753 {
2754   \bool_if:NT \l__enumext_check_answers_bool
2755   {
2756     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2757     {
2758       \__enumext_store_addto_seq:n
2759       {
2760         \item \begin{enumext}
2761       }
2762     }
2763     {
2764       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2765       {
2766         \item \begin{enumext} [
2767       }
2768       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2769       {
2770         ]
2771       }
2772       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2773     }
2774   }
2775 }
2776 \cs_new_protected:Nn \__enumext_store_level_close:
2777 {
2778   \bool_if:NT \l__enumext_check_answers_bool
2779   {
2780     \__enumext_store_addto_seq:n { \end{enumext} }
2781   }
2782 }

```

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

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.

```

2783 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2784 {
2785   \bool_if:NT \l__enumext_check_answers_bool
2786   {
2787     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2788     {

```

```

2789         \__enumext_store_addto_seq:n
2790         {
2791             \item \begin{enumext*}
2792         }
2793     }
2794     {
2795         \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2796         {
2797             \item \begin{enumext*}[
2798         }
2799         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2800         {
2801             ]
2802         }
2803         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2804     }
2805 }
2806 }
2807 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2808 {
2809     \bool_if:NT \l__enumext_check_answers_bool
2810     {
2811         \__enumext_store_addto_seq:n { \end{enumext*} }
2812     }
2813 }

```

(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`

```

2814 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2815 {
2816     \mode_leave_vertical:
2817     \skip_horizontal:n { -\dim_use:N #2 }
2818     \hbox_overlap_left:n
2819     {
2820         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2821         {
2822             \tl_use:N \l__enumext_mark_answer_sym_tl
2823         }
2824     }
2825     \skip_horizontal:n { \dim_use:N #2 }
2826 }
2827 \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`.

```

2828 \cs_new_protected:Nn \__enumext_store_internal_ref:
2829 {
2830     \cs_set_protected:Npn \__enumext_tmp:n ##1
2831     {
2832         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2833         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2834         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2835         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2836     }
2837     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2838     \cs_set:Npn \__enumext_tmp:n ##1
2839     { . \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.

```

2840 \bool_lazy_all:nT
2841 {
2842   { \bool_if_p:N \g__enumext_starred_bool }
2843   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2844 }
2845 {
2846   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2847   { \tl_use:N \l__enumext_label_copy_vii_tl }
2848 }
2849 \bool_lazy_all:nT
2850 {
2851   { \bool_not_p:n { \g__enumext_standar_bool } }
2852   { \bool_if_p:N \l__enumext_standar_bool }
2853   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2854 }
2855 {
2856   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2857   {
2858     \tl_use:N \l__enumext_label_copy_vii_tl
2859     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2860   }
2861 }

```

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.

```

2862 \bool_lazy_all:nT
2863 {
2864   { \bool_if_p:N \g__enumext_standar_bool }
2865   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2866   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2867 }
2868 {
2869   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2870   {
2871     \tl_use:N \l__enumext_label_copy_i_tl
2872     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2873   }
2874 }
2875 \cs_set:Npn \l__enumext_tmp:n ##1
2876 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2877 \bool_lazy_all:nT
2878 {
2879   { \bool_if_p:N \g__enumext_standar_bool }
2880   { \bool_if_p:N \l__enumext_starred_bool }
2881   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2882 }
2883 {
2884   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2885   {
2886     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2887     \tl_use:N \l__enumext_label_copy_vii_tl
2888   }
2889 }

```

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

```

2890 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2891 {
2892   \l__enumext_store_name_tl \c_colon_str
2893   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2894 }

```

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

```

2895 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2896 {
2897   \l__enumext_newlabel:nn
2898   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2899   { \l__enumext_newlabel_arg_two_tl }
2900 }
2901 \l__enumext_write_aux_file_tl

```

```
2902 }
```

(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 $\langle \textit{argument} \rangle$ to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the *save-ref* key and 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*” $\langle \textit{argument} \rangle$.

```
2903 \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2904 {
2905   \int_gincr:N \g__enumext_item_anskey_int
2906   \__enumext_store_addto_prop:n {#1}
2907   \bool_if:NT \l__enumext_store_ref_key_bool
2908   {
2909     \__enumext_store_internal_ref:
2910   }
2911   \__enumext_anskey_show_wrap_left:n { #1 }
```

Now we start processing the $[\langle \textit{key} = \textit{val} \rangle]$ 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 $\langle \textit{keys} \rangle$, if the *break-col* key is present and the command is running under *enumext* (not in *enumext**) we will add `\columnbreak` and then `\item`.

```
2912   \tl_clear:N \l__enumext_store_anskey_arg_tl
2913   \bool_lazy_and:nnT
2914     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2915     { \bool_not_p:n { \l__enumext_starred_bool } }
2916     {
2917       \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2918     }
2919   \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 $\langle \textit{number} \rangle$ to `\l__enumext_store_anskey_arg_tl`.

```
2920   \bool_lazy_and:nnT
2921     { \bool_not_p:n { \l__enumext_starred_bool } }
2922     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2923     {
2924       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2925         {
2926           ( \exp_not:V \l__enumext_store_item_join_int )
2927         }
2928     }
```

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 $\langle \textit{argument} \rangle$ for `\anskey` or $\langle \textit{body} \rangle$ for `anskey*`.

```
2929   \bool_if:NTF \l__enumext_store_item_star_bool
2930   {
2931     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2932     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2933     {
2934       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2935         {
2936           [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2937         }
2938     }
2939   \dim_compare:nT
2940   {
2941     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2942   }
2943   {
2944     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2945       {
2946         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2947       }
2948   }
2949   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2950 }
2951 {
2952   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2953 }
```

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*.

```

2954 \bool_lazy_and:nnT
2955 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2956 { \bool_if_p:N \l__enumext_hyperref_bool }
2957 {
2958   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2959   {
2960     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2961     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2962   }
2963 }
2964 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2965 }

```

(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.

```

2966 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2967 {
2968   \par
2969   \bool_if:NTF \l__enumext_starred_bool
2970   {
2971     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2972     {
2973       \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2974     }
2975     \__enumext_print_keyans_box:NN
2976     \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2977   }
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
2982       {
2983         \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2984       }
2985     }
2986     \__enumext_print_keyans_box:cc
2987     { \l__enumext_labelwidth_ \__enumext_level: _dim } { \l__enumext_mark_sym_sep_dim }
2988   }
2989   \__enumext_anskey_wrapper:n { #1 }
2990 }

```

(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.

```

2991 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2992 {
2993   \bool_if:NT \l__enumext_show_answer_bool
2994   {
2995     \__enumext_anskey_show_wrap_arg:n { #1 }
2996   }
2997   \bool_if:NT \l__enumext_show_position_bool
2998   {
2999     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3000     {
3001       \group_begin:
3002       \exp_not:N \normalfont
3003       \exp_not:N \footnotesize [ \int_eval:n
3004       {
3005         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3006       }
3007       ]
3008       \group_end:
3009     }

```

```

3010         \__enumext_anskey_show_wrap_arg:n { #1 }
3011     }
3012 }

```

(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 \keys_define:nn { enumext / anskey }
item-join {
item-star {
item-sym* {
item-pos* {
unknown
\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:nn
3013 \keys_define:nn { enumext / anskey }
3014 {
3015     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3016     break-col .default:n = true,
3017     break-col .value_forbidden:n = true,
3018     item-join .int_set:N = \l__enumext_store_item_join_int,
3019     item-join .value_required:n = true,
3020     item-star .bool_set:N = \l__enumext_store_item_star_bool,
3021     item-star .default:n = true,
3022     item-star .value_forbidden:n = true,
3023     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
3024     item-sym* .value_required:n = true,
3025     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
3026     item-pos* .value_required:n = true,
3027     unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
3028 }

```

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`.

```

3029 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
3030 {
3031     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
3032 }
3033 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
3034 {
3035     \tl_if_blank:nTF {#2}
3036     {
3037         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
3038     }
3039     {
3040         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
3041     }
3042 }

```

(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`.

```

3043 \NewDocumentCommand \anskey { o +m }
3044 {
3045     \__enumext_anskey_safe_outer:
3046     \group_begin:
3047         \bool_if:NT \l__enumext_check_answers_bool
3048         {
3049             \tl_if_no_value:nF {#1}
3050             {
3051                 \keys_set:nn { enumext / anskey } {#1}
3052             }
3053             \tl_if_blank:nTF {#2}
3054             {
3055                 \msg_error:nn { enumext } { anskey-empty-arg }

```



```

3056         }
3057     {
3058         \__enumext_anskey_safe_inner:
3059         \__enumext_store_anskey_arg:n {#2}
3060     }
3061 }
3062 \group_end:
3063 }

```

(End of definition for `\anskey`. This function is documented on page 14.)

13.33.1 Internal functions for the command

`__enumext_anskey_safe_outer:` 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.

`__enumext_anskey_safe_inner:`

```

3064 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
3065 {
3066     \bool_if:NF \l__enumext_store_active_bool
3067     {
3068         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
3069     }
3070     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3071     {
3072         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
3073     }
3074     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3075     {
3076         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
3077     }
3078     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3079     {
3080         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
3081     }
3082 }

```

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.

```

3083 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3084 {
3085     \int_incr:N \l__enumext_anskey_level_int
3086     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
3087     {
3088         \msg_error:nn { enumext } { anskey-nested }
3089     }
3090     \bool_if:NF \l__enumext_item_number_bool
3091     {
3092         \msg_error:nn { enumext } { anskey-unnumber-item }
3093     }
3094     \mode_if_math:T
3095     {
3096         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
3097     }
3098 }

```

(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.

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.

```

break-col 3099 \keys_define:nn { enumext / anskey* }
item-join 3100 {
item-star 3101     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
item-sym* 3102     break-col .default:n = true,
item-pos* 3103     break-col .value_forbidden:n = true,
force-eol 3104     item-join .int_set:N = \l__enumext_store_item_join_int,
write-env 3105     item-join .value_required:n = true,
overwrite
unknown

```

```

3106     item-star .bool_set:N = \l__enumext_store_item_star_bool,
3107     item-star .default:n = true,
3108     item-star .value_forbidden:n = true,
3109     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
3110     item-sym* .value_required:n = true,
3111     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
3112     item-pos* .value_required:n = true,
3113     force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
3114     force-eol .initial:n = false,
3115     force-eol .default:n = true,
3116     write-env .code:n = {
3117         \bool_set_true:N \l__enumext_write_anskey_env_bool
3118         \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
3119     },
3120     write-env .value_required:n = true,
3121     overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
3122     overwrite .initial:n = false,
3123     overwrite .default:n = true,
3124     unknown .code:n = { \__enumext_anskey_env_unknown:n {#1} },
3125 }

```

(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`.

```

3126 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
3127 {
3128     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
3129 }
3130 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
3131 {
3132     \tl_if_blank:nTF {#2}
3133     {
3134         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
3135     }
3136     {
3137         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3138     }
3139 }

```

(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.

```

3140 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
3141 {
3142     \bool_if:NTF \l__enumext_write_anskey_env_bool
3143     {
3144         \file_if_exist:nTF {#1}
3145         {
3146             \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
3147             {
3148                 \msg_warning:nne { enumext } { overwrite-file } {#1}
3149                 \prg_return_true:
3150             }
3151             {
3152                 \msg_warning:nne { enumext } { not-writing } {#1}
3153                 \prg_return_false:
3154             }
3155         }
3156         {
3157             \msg_warning:nne { enumext } { writing-file } {#1}
3158             \prg_return_true:
3159         }
3160     }
3161     { \prg_return_false: }
3162 }

```

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.

```

3163 \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2

```

```

3164 {
3165     \__enumext_anskey_env_file_if_writable:nT {#1}
3166     {
3167         \iow_open:Nn \__enumext_write_anskey_env_file_iow {#1}
3168         \iow_now:Nn \__enumext_write_anskey_env_file_iow {#2}
3169         \iow_close:N \__enumext_write_anskey_env_file_iow
3170     }
3171 }
3172 \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`.

```

3173 \NewDocumentEnvironment{anskey*}{o c }
3174 {
3175     \__enumext_anskey_env_safe_outer:
3176     \bool_if:NT \__enumext_check_answers_bool
3177     {
3178         \tl_if_novalue:nF {#1}
3179         {
3180             \keys_set:nn { enumext / anskey* } {#1}
3181         }
3182         \tl_if_blank:nTF {#2}
3183         {
3184             \msg_error:nn { enumext } { anskey-empty-arg }
3185         }
3186         {
3187             \__enumext_anskey_env_safe_inner:
3188             \__enumext_store_anskey_env:n {#2}
3189         }
3190     }
3191 } { }

```

(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.

```

3192 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3193 {
3194     \bool_if:NF \__enumext_store_active_bool
3195     {
3196         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3197     }
3198     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3199     {
3200         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
3201     }
3202     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
3203     {
3204         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3205     }
3206     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3207     {
3208         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3209     }
3210 }

```

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.

```

3211 \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
3212 {
3213     \bool_if:NF \__enumext_item_number_bool
3214     {
3215         \msg_error:nn { enumext } { anskey-unnumber-item }
3216     }

```

```

3217 \mode_if_math:T
3218 {
3219   \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
3220 }
3221 }

```

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.

```

3222 \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
3223 {
3224   \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
3225   \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^J }
3226   \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N ^^J }
3227   \__enumext_anskey_env_file_write:VV
3228   \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
3229   \bool_if:NF \l__enumext_anskey_env_force_eol_bool
3230   {
3231     \tl_put_right:Ne \l__enumext_store_anskey_env_tl
3232     {
3233       \c__enumext_anskey_env_hidden_space_str
3234     }
3235   }
3236   \exp_args:Ne
3237   \__enumext_store_anskey_arg:n
3238   {
3239     \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
3240   }
3241 }

```

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.

```

3242 \cs_new_protected:Nn \__enumext_execute_after_env:
3243 {
3244   \int_compare:nNt { \l__enumext_level_int } = { 0 }
3245   {
3246     \tl_if_empty:NF \g__enumext_store_name_tl
3247     {
3248       \__enumext_stop_save_ans_msg:
3249       \__enumext_item_answer_diff:
3250       \__enumext_log_global_vars:
3251       \__enumext_log_answer_vars:
3252       \bool_if:NTF \g__enumext_check_ans_key_bool
3253       {
3254         \__enumext_check_ans_show:
3255       }
3256       { \__enumext_check_ans_log: }
3257     }
3258     \__enumext_reset_global_vars:
3259   }
3260 }

```

This function is passed to the function `__enumext_after_env:nn` 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 `\item*` in `keyans` environment and the current $\langle label \rangle$ for `\anspic*` in `keyanspic` environment followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `__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`.

```

3261 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3262 {
3263   \tl_clear:N \__enumext_store_current_label_tl
3264   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
3265   {
3266     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
3267   }
3268   {
3269     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
3270   }

```

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

```

3271   \tl_if_novalue:nF { #1 }
3272   {
3273     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_v_tl
3274     {
3275       \tl_put_right:NV \__enumext_store_current_label_tl \__enumext_store_keyans_item_opt_sep_v_tl
3276     }
3277     \tl_put_right:Nn \__enumext_store_current_label_tl { #1 }
3278   }
3279   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
3280 }

```

(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 `\item*` and `\anspic*` 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)`.

`__enumext_keyans_store_ref:`

`__enumext_keyans_store_ref_aux_i:`

`__enumext_keyans_store_ref_aux_ii:`

The function `__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the `save-ref` key for `\item*` and `\anspic*` commands. First 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.

```

3281 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3282 {
3283   \bool_if:NT \__enumext_store_ref_key_bool
3284   {
3285     \cs_set_protected:Npn \__enumext_tmp:n ##1
3286     {
3287       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3288       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3289       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3290       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3291     }
3292     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3293     \__enumext_keyans_store_ref_aux_i:
3294   }
3295 }

```

The auxiliary function `__enumext_keyans_store_ref_aux_i:` set the variable `__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`.

```

3296 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3297 {
3298   \bool_if:NT \g__enumext_starred_bool
3299   {
3300     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
3301   }
3302   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3303   {
3304     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3305     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }

```

```

3306     }
3307     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3308     {
3309         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3310         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3311     }
3312     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3313     {
3314         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3315         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3316     }
3317     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3318     {
3319         \l__enumext_store_name_tl \c_colon_str
3320         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3321     }
3322     \__enumext_keyans_store_ref_aux_ii:
3323 }

```

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.

```

3324 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3325 {
3326     \tl_put_right:Ne \l__enumext_write_aux_file_tl
3327     {
3328         \__enumext_newlabel:nn
3329         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3330         { \l__enumext_newlabel_arg_two_tl }
3331     }
3332     \l__enumext_write_aux_file_tl
3333 }

```

(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 *⟨label⟩* `\l__enumext_label_v_tl` for the *keyans* environment and the `\l__enumext_label_vi_tl` for the *keyanspic* environment when using *⟨item⟩* and *⟨anspic⟩*, followed by the *⟨contents⟩* of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the *save-ans* key.

```

3334 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3335 {
3336     \tl_clear:N \l__enumext_store_current_label_tl
3337     \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
3338     {
3339         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3340     }
3341     {
3342         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3343     }
3344     \tl_if_novalue:nF { #1 }
3345     {
3346         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3347         {
3348             \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt.
3349         }
3350         \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3351     }
3352     \__enumext_keyans_addto_seq_link:
3353 }

```

Checks if the *save-ref* key is active along with the *hyperref* package load, if both conditions are met, it will create the *⟨hyperlink⟩* and then store using the `__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_current_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `__enumext_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_item_anskey_int` handled by the *check-ans* key.

```

3354 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3355 {
3356     \bool_lazy_and:nnT

```

```

3357     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3358     { \bool_if_p:N \l__enumext_hyperref_bool }
3359     {
3360         \tl_put_right:Ne \l__enumext_store_current_label_tl
3361         {
3362             \hfill \exp_not:N \hyperlink
3363             {
3364                 \exp_not:V \l__enumext_newlabel_arg_one_tl
3365             }
3366             { \exp_not:V \l__enumext_mark_ref_sym_tl }
3367         }
3368     }
3369     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3370     \bool_if:NT \l__enumext_check_answers_bool
3371     {
3372         \int_gincr:N \g__enumext_item_anskey_int
3373     }
3374 }

```

(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

The function `__enumext_keyans_save_item_opt:n` will save the optional argument of `\item*` and `\anspic*` in the variable `\l__enumext_store_current_opt_arg_tl`.

```

\__enumext_keyans_save_item_opt:n
\__enumext_keyans_show_item_opt:
\__enumext_keyans_show_item_opt_viii:

```

```

3375 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3376 {
3377     \tl_if_novalue:nF { #1 }
3378     {
3379         \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3380     }
3381 }

```

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.

```

3382 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3383 {
3384     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3385     {
3386         \bool_lazy_or:nnT
3387         { \bool_if_p:N \l__enumext_show_answer_bool }
3388         { \bool_if_p:N \l__enumext_show_position_bool }
3389         {
3390             \__enumext_keyans_wrapper_opt_v:n
3391             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3392         }
3393     }
3394 }

```

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.

```

3395 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3396 {
3397     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3398     {
3399         \bool_lazy_or:nnT
3400         { \bool_if_p:N \l__enumext_show_answer_bool }
3401         { \bool_if_p:N \l__enumext_show_position_bool }
3402         {
3403             \__enumext_keyans_wrapper_opt_viii:n
3404             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3405         }
3406     }
3407 }

```

(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)`.

```

3408 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3409 {

```



```

3410 \__enumext_label_width_by_box:Nn
3411 \__enumext_mark_sep_tmpa_dim { \__enumext_label_v_tl }
3412 \str_case:Vn \__enumext_align_label_pos_v_str
3413 {
3414   { l }
3415   {
3416     \dim_set:Nn \__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3417   }
3418   { r }
3419   {
3420     \dim_set:Nn \__enumext_mark_sep_tmpb_dim
3421     { \__enumext_labelwidth_v_dim - \__enumext_mark_sep_tmpa_dim }
3422   }
3423   { c }
3424   {
3425     \dim_set:Nn \__enumext_mark_sep_tmpb_dim
3426     { 0.5\__enumext_labelwidth_v_dim - 0.5\__enumext_mark_sep_tmpa_dim }
3427   }
3428 }

```

Here we set the default values for the key `mark-ans*`, `mark-sep*` and `mark-pos*`.

```

3429 \dim_compare:nNt { \__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3430 {
3431   \dim_set:Nn \__enumext_mark_sym_sep_v_dim { \__enumext_labelsep_v_dim }
3432 }
3433 \tl_set_eq:NN \__enumext_mark_answer_sym_tl \__enumext_mark_answer_sym_v_tl
3434 \dim_add:Nn \__enumext_mark_sym_sep_v_dim { \__enumext_mark_sep_tmpb_dim }
3435 \str_set_eq:NN \__enumext_mark_position_str \__enumext_mark_position_v_str
3436 }

```

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

```

3437 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3438 {
3439   \bool_lazy_all:nT
3440   {
3441     { \bool_if_p:N \__enumext_show_answer_bool }
3442     { \bool_if_p:N \__enumext_item_wrap_key_bool }
3443   }
3444   {
3445     \__enumext_keyans_pos_mark_set:
3446     \__enumext_print_keyans_box:NN
3447     \__enumext_labelwidth_v_dim \__enumext_mark_sym_sep_v_dim
3448   }
3449 }

```

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.

```

3450 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3451 {
3452   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
3453   {
3454     \int_incr:N \__enumext_show_pos_tmp_int
3455   }
3456   {
3457     \int_zero:N \__enumext_show_pos_tmp_int
3458   }
3459   \bool_lazy_all:nT
3460   {
3461     { \bool_if_p:N \__enumext_show_position_bool }
3462     { \bool_if_p:N \__enumext_item_wrap_key_bool }
3463   }
3464   {
3465     \tl_set:Nc \__enumext_mark_answer_sym_v_tl
3466     {
3467       \group_begin:
3468       \exp_not:N \normalfont
3469       \exp_not:N \footnotesize [ \int_eval:n
3470       {
3471         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3472         + \__enumext_show_pos_tmp_int
3473       }

```

```

3474         ]
3475     \group_end:
3476 }
3477 \__enumext_keyans_pos_mark_set:
3478 \__enumext_print_keyans_box:NN
3479     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3480 }
3481 }

```

(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[⟨symbol⟩]` 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`.

```

3482 \cs_new_protected:Npn \__enumext_default_item:n #1
3483 {
3484     \tl_if_novalue:nTF {#1}
3485     {
3486         \bool_if:NT \l__enumext_check_answers_bool
3487         {
3488             \int_gincr:N \g__enumext_item_number_int
3489             \bool_set_true:N \l__enumext_item_number_bool
3490         }
3491         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3492         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3493     }
3494     {
3495         \bool_set_eq:cc
3496         { \l__enumext_wrap_label_ \__enumext_level: _bool }
3497         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3498         \__enumext_item_std:w {#1} \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3499     }
3500 }

```

(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`.

```

3501 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3502 {
3503     \tl_if_novalue:nTF {#1}
3504     {
3505         \tl_gset_eq:Nc
3506         \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3507     }

```

```

3508     {
3509         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3510     }
3511     \tl_if_novalue:nTF {#2}
3512     {
3513         \dim_set_eq:cc
3514         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
3515         { \__enumext_labelsep_ \__enumext_level: _dim }
3516     }
3517     {
3518         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3519     }
3520     \bool_if:NT \l__enumext_check_answers_bool
3521     {
3522         \int_gincr:N \g__enumext_item_number_int
3523         \bool_set_true:N \l__enumext_item_number_bool
3524     }
3525     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3526     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3527 }

```

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

```

3528 \cs_new_protected:Nn \__enumext_item_starred_exec:
3529 {
3530     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3531     {
3532         \mode_leave_vertical:
3533         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3534         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3535         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3536     }
3537 }

```

(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).

```

3538 \cs_new_protected:Nn \__enumext_redefine_item:
3539 {
3540     \RenewDocumentCommand \item { s o o }
3541     {
3542         \bool_if:nTF {##1}
3543         {
3544             \__enumext_item_starred_exec:nn {##2} {##3}
3545         }
3546         { \__enumext_default_item:n {##2} }
3547     }
3548 }

```

(End of definition for `__enumext_redefine_item:.`)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makelabel` 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).

```

3549 \cs_new_protected:Nn \__enumext_make_label:
3550 {
3551     \IfDocumentMetadataTF
3552     {
3553         \__enumext_make_label_box:
3554     }
3555     {
3556         \bool_if:NTF \l__enumext_mode_box_bool
3557         {
3558             \__enumext_make_label_box:
3559         }
3560         {
3561             \__enumext_make_label_std:
3562         }
3563     }
3564 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3565 \cs_new_protected:Nn \__enumext_make_label_std:
3566 {
3567   \RenewDocumentCommand \makeLabel { m }
3568   {
3569     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3570     \__enumext_item_starred_exec:
3571     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3572     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3573     {
3574       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3575     }
3576     { ##1 }
3577     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3578     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3579   }
3580 }

```

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`.

```

3581 \cs_new_protected:Nn \__enumext_make_label_box:
3582 {
3583   \RenewDocumentCommand \makeLabel { m }
3584   {
3585     \strut\smash
3586     {
3587       \makebox
3588       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3589       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3590       {
3591         \__enumext_item_starred_exec:
3592         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3593         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3594         {
3595           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3596         }
3597         { ##1 }
3598         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3599       }
3600     } % close smash
3601   }
3602 }

```

(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*
3603 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3604 {
3605   \keys_define:nn { enumext / #1 }
3606   {
3607     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3608     item-sym* .value_required:n = true,
3609     item-sym* .initial:n = {\textborn},
3610     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3611     item-pos* .value_required:n = true,
3612   }
3613 }
3614 \clist_map_inline:nn
3615 {
3616   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3617 }
3618 { \__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`.

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3619 \cs_set_protected:Npn \__enumext_tmp:n #1
3620 {
3621   \keys_define:nn { enumext / #1 }
3622   {
3623     unknown .code:n = {
3624       \tl_set:Nn \l__enumext_envir_name_tl {#1}
3625       \__enumext_keyans_unknown_keys:n {#1}
3626     },
3627   }
3628 }
3629 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3630 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3631 {
3632   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3633 }
3634 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3635 {
3636   \tl_if_blank:nTF {#2}
3637   {
3638     \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3639   }
3640   {
3641     \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3642   }
3643 }
```

(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.

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3644 \keys_define:nn { enumext / enumext* }
3645 {
3646   unknown .code:n = {
3647     \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
3648     \__enumext_starred_unknown_keys:n {#1}
3649   },
3650 }
```

Internal functions for handling `unknown` key.

```

3651 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3652 {
3653   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3654 }
3655 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3656 {
3657   \tl_if_blank:nTF {#2}
3658   {
3659     \msg_error:nne { enumext } { starred-unknown-key } {#1}
3660   }
3661   {
3662     \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}
3663   }
3664 }
```

(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.

```

3665 \cs_set_protected:Npn \__enumext_tmp:n #1
3666 {
3667   \keys_define:nn { enumext / level-#1 }
3668   {
3669     unknown .code:n = {
3670       \int_set:Nn \__enumext_level_int { #1 }
3671       \tl_set:Nn \__enumext_envir_name_tl { enumext }
3672       \__enumext_standar_unknown_keys:n {##1}
3673     },
3674   }
3675 }
3676 \clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3677 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3678 {
3679   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3680 }
3681 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3682 {
3683   \tl_if_blank:nTF {#2}
3684   {
3685     \msg_error:nne { enumext } { standar-unknown-key } {#1}
3686   }
3687   {
3688     \msg_error:nnee { enumext } { standar-unknown-key-value } {#1} {#2}
3689   }
3690 }

```

(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`.

```

3691 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3692 {
3693   \tl_if_novalue:nTF { #1 }
3694   {
3695     \bool_set_true:N \__enumext_wrap_label_v_bool
3696     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3697   }
3698   {
3699     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3700     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3701   }
3702 }

```

(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.

```

3703 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3704 {
3705   \__enumext_keyans_save_item_opt:n { #1 }
3706   \bool_set_true:N \__enumext_wrap_label_v_bool
3707   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3708   \__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.

```

3709   \__enumext_keyans_addto_prop:n { #1 }

```

```

3710     \__enumext_keyans_store_ref:
3711     \__enumext_keyans_addto_seq:n { #1 }
3712     \int_gincr:N \__enumext_check_starred_cmd_int
3713 }

```

(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).

```

3714 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3715 {
3716     \RenewDocumentCommand \item { s o }
3717     {
3718         \bool_if:nTF {##1}
3719         {
3720             \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3721             \peek_remove_spaces:n
3722             {
3723                 \__enumext_keyans_starred_item:n {##2}
3724             }
3725         }
3726         {
3727             \bool_set_false:N \l__enumext_item_wrap_key_bool
3728             \__enumext_keyans_default_item:n {##2}
3729         }
3730     }
3731 }

```

(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).

```

3732 \cs_new_protected:Nn \__enumext_keyans_make_label:
3733 {
3734     \IfDocumentMetadataTF
3735     {
3736         \__enumext_keyans_make_label_box:
3737     }
3738     {
3739         \bool_if:NTF \l__enumext_mode_box_bool
3740         {
3741             \__enumext_keyans_make_label_box:
3742         }
3743         {
3744             \__enumext_keyans_make_label_std:
3745         }
3746     }
3747 }

```

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

```

3748 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3749 {
3750     \bool_lazy_all:nT
3751     {
3752         { \bool_if_p:N \l__enumext_wrap_label_v_bool }
3753         { \bool_if_p:N \l__enumext_show_answer_bool }
3754         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3755         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3756     }
3757     {
3758         \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3759     }
3760     \bool_if:NTF \l__enumext_wrap_label_v_bool
3761     {
3762         \__enumext_wrapper_label_v:n { #1 }

```



```

3763     }
3764     { #1 }
3765 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3766 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3767 {
3768   \RenewDocumentCommand \makeLabel { m }
3769   {
3770     \tl_use:N \l__enumext_label_fill_left_v_tl
3771     \__enumext_keyans_show_ans:
3772     \__enumext_keyans_show_pos:
3773     \tl_use:N \l__enumext_label_font_style_v_tl
3774     \__enumext_keyans_wrapper_label:n { ##1 }
3775     \tl_use:N \l__enumext_label_fill_right_v_tl
3776   }
3777 }

```

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

```

3778 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3779 {
3780   \RenewDocumentCommand \makeLabel { m }
3781   {
3782     \strut\smash
3783     {
3784       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3785       {
3786         \__enumext_keyans_show_ans:
3787         \__enumext_keyans_show_pos:
3788         \tl_use:N \l__enumext_label_font_style_v_tl
3789         \__enumext_keyans_wrapper_label:n { ##1 }
3790       }
3791     }
3792   }
3793 }

```

(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.

13.41.1 Calculation of `\leftmargin` and `\itemindent`

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

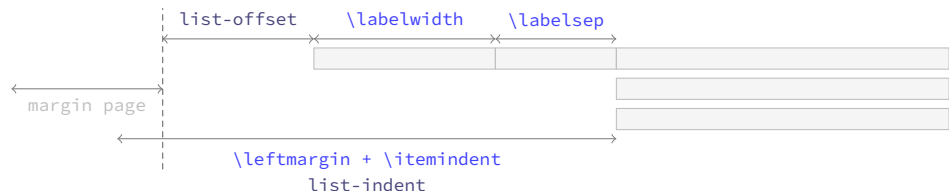


Figure 9: Representation of standard horizontal lengths in `list` environment.

The idea is to have control over these margins so that 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.

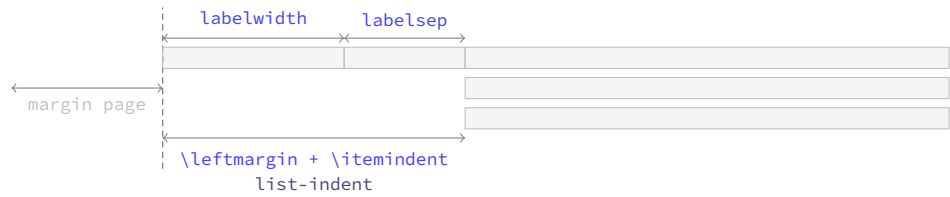
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:cccccc

```

The function `__enumext_calc_hspace:NNNNNNN` takes seven arguments to be able to determine horizontal spaces for all list environment:

Figure 10: Representation of horizontal lengths concept in list in `enumext`.Figure 11: Default horizontal lengths in `enumext`.

```

#1: \l__enumext_labelwidth_X_dim      #2: \l__enumext_labelsep_X_dim
#3: \l__enumext_listoffset_X_dim      #4: \l__enumext_leftmargin_tmp_X_dim
#5: \l__enumext_leftmargin_X_dim      #6: \l__enumext_itemindent_X_dim
#7: \l__enumext_leftmargin_tmp_X_bool

```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

```

3794 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3795 {
3796   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3797   {
3798     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3799     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3800   }
3801   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3802   {
3803     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3804     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3805   }

```

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

```

3806   \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

We now analyze the cases and set the values for `\leftmargin` and `\itemindent`.

```

3807   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3808   {
3809     \dim_set:Nn #6 { #1 + #2 - #4 }
3810     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3811   }
3812   {
3813     \dim_compare:nNnT { #4 } = { #1 + #2 }
3814     { \dim_set:Nn #6 { \c_zero_dim } }
3815     \dim_compare:nNnT { #4 } < { #1 + #2 }
3816     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3817     \dim_compare:nNnT { #4 } > { #1 + #2 }
3818     {
3819       \dim_set:Nn #6 { -#1 - #2 + #4 }
3820       \dim_set:Nn #6 { #6*-1 }
3821     }
3822     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3823   }
3824 }
3825 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

13.41.2 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.

```

3826 \cs_set_protected:Npn \__enumext_tmp:n #1
3827 {
3828   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }

```

```

3829 {
3830   \__enumext_calc_hspace:ccccc
3831   { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3832   { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3833   { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3834   { \__enumext_leftmargin_tmp_#1_bool }
3835   \clist_map_inline:nn
3836     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3837     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3838   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3839     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3840   \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3841     { \int_set_eq:cc {@####1} { \__enumext_####1_#1_int } }
3842   \usecounter { enumX#1 }
3843   \str_if_eq:nnTF {#1} { v }
3844   {
3845     \__enumext_keyans_redefine_item:
3846     \__enumext_keyans_make_label:
3847     \__enumext_keyans_ref:
3848     \__enumext_keyans_fake_item_indent:
3849     \bool_if:cT { \__enumext_show_length_#1_bool }
3850     {
3851       \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3852     }
3853   }
3854   {
3855     \__enumext_redefine_item:
3856     \__enumext_make_label:
3857     \__enumext_standar_ref:
3858     \__enumext_fake_item_indent:
3859     \bool_if:cT { \__enumext_show_length_#1_bool }
3860     {
3861       \msg_term:nnne { enumext } { list-lengths } {#1}
3862       { \int_use:N \__enumext_level_int }
3863     }
3864   }
3865 }
3866 }
3867 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_list_arg_two_vii:
  \__enumext_list_arg_two_viii:

```

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `\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 `\listparindent` and `parsep` to set the value of `\parskip` locally.

```

3868 \cs_set_protected:Npn \__enumext_tmp:n #1
3869 {
3870   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3871   {
3872     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3873     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3874     \__enumext_calc_hspace:ccccc
3875     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3876     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3877     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3878     { \__enumext_leftmargin_tmp_#1_bool }
3879     \clist_map_inline:nn
3880       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3881       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3882     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3883       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3884     \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3885       { \int_set_eq:cc {@####1} { \__enumext_####1_#1_int } }
3886     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3887     \skip_zero:N \partopsep
3888     \usecounter { enumX#1 }
3889     \__enumext_starred_ref:
3890     \str_if_eq:nnTF {#1} { vii }
3891     {
3892       \__enumext_fake_item_indent_vii:

```

```

3893         \bool_if:cT { \__enumext_show_length_vii_bool }
3894         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3895     }
3896     {
3897         \__enumext_fake_item_indent_viii:
3898         \bool_if:cT { \__enumext_show_length_#1_bool }
3899         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3900     }
3901 }
3902 }
3903 \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”.

```

3904 \cs_new_protected:Nn \__enumext_safe_exec:
3905 {
3906     \__enumext_is_not_nested:
3907     \__enumext_internal_mini_page:
3908     \int_incr:N \l__enumext_level_int
3909     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3910     { \msg_fatal:nn { enumext } { list-too-deep } }
3911     \bool_set_true:N \l__enumext_standar_bool
3912     \bool_set_false:N \l__enumext_starred_bool
3913     \__enumext_is_on_first_level:
3914 }

```

(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.

```

3915 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3916 {
3917     \tl_if_novalue:nF {#1}
3918     {
3919         \str_clear:N \l__enumext_series_name_str
3920         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3921         {
3922             \keys_set:nn { enumext / level-1 } {#1}
3923             \bool_if:NF \l__enumext_print_keyans_cmd_bool
3924             {
3925                 \__enumext_parse_series:n {#1}
3926             }
3927             \__enumext_nested_base_line_fix:
3928         }
3929         {
3930             \exp_args:Ne \keys_set:nn
3931             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3932             \bool_if:NF \l__enumext_print_keyans_cmd_bool
3933             {
3934                 \__enumext_parse_series:n {#1}
3935             }
3936         }
3937         \__enumext_store_active_keys:n {#1}
3938     }
3939 }

```

(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*`.

```

3940 \cs_new_protected:Nn \__enumext_start_store_level:
3941 {
3942   \bool_lazy_all:nT
3943   {
3944     { \bool_if_p:N \l__enumext_store_active_bool }
3945     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3946     { \bool_if_p:N \g__enumext_standar_bool }
3947   }
3948   {
3949     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3950     {
3951       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3952       \__enumext_store_level_open:
3953     }
3954   }

```

If `enumext` are nested in `enumext*` add `__enumext_store_level_open:` to preserve the “*storing structure*”.

```

3955   \bool_lazy_all:nT
3956   {
3957     { \bool_if_p:N \l__enumext_store_active_bool }
3958     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3959     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3960   }
3961   {
3962     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3963     {
3964       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3965       \__enumext_store_level_open:
3966     }
3967   }
3968 }

```

(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*`.

```

3969 \cs_new_protected:Nn \__enumext_stop_store_level:
3970 {
3971   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3972   {
3973     \__enumext_store_level_close:
3974   }
3975 }

```

(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.

```

3976 \cs_new_protected:Nn \__enumext_multicols_start:
3977 {
3978   \int_compare:nNnT
3979   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3980   {
3981     \dim_compare:nNnT
3982     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3983     {
3984       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3985       {
3986         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3987           + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3988         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3989         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3990       }
3991     }
3992     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3993     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3994     {
3995       \dim_zero:N \columnseprule
3996     }

```

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.

```

3997     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3998     {
3999         \skip_zero:N \multicolsep
4000         \__enumext_multi_addvspace:
4001     }
4002     \raggedcolumns
4003     \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
4004 }
4005 }

```

(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:`.

```

4006 \cs_new_protected:Nn \__enumext_multicols_stop:
4007 {
4008     \int_compare:nNnTF
4009     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
4010     {
4011         \__enumext_stop_list:
4012         \__enumext_stop_store_level:
4013         \end{multicols}
4014         \__enumext_unskip_unkern:
4015         \__enumext_unskip_unkern:
4016         \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
4017     }
4018     {
4019         \__enumext_stop_list:
4020         \__enumext_stop_store_level:
4021     }
4022 }

```

(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.

```

4023 \cs_new_protected:Nn \__enumext_before_list:
4024 {
4025     \__enumext_vspace_above:
4026     \__enumext_before_args_exec:
4027     \__enumext_check_ans_active:

```

When the `mini-env` key is active it will set the value of the `__enumext_minipage_right_X_dim` to be the *width* of the `__enumext_mini_page` environment on the “*right side*”, using this value together with the value of the `__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `__enumext_minipage_left_X_dim` will be set, which will be the *width* of `__enumext_mini_page` environment on the “*left side*”, always having a current `\linewidth` as *maximum width* between them.

```

4028     \dim_compare:nNnT
4029     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
4030     {
4031         \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
4032         {
4033             \linewidth
4034             - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
4035             - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
4036         }

```

The boolean variable `__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_minipage_add_space:` is called and the `__enumext_mini_page` environment on the “*left side*” will be initialized followed by the “*vertical spacing*” applied to preserve the “*baseline*” between the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

```

4037     \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
4038     \int_gincr:N \g__enumext_minipage_stat_int
4039     \__enumext_minipage_add_space:

```

```

4040         \noindent
4041         \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
4042     }
4043     \__enumext_multicols_start:
4044 }

```

(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.

```

4045 \cs_new_protected:Nn \__enumext_second_part:
4046 {
4047     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
4048     {
4049         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
4050         {
4051             \msg_warning:nn { enumext } { missing-miniright }
4052             \miniright
4053         }
4054         \int_gzero:N \g__enumext_minipage_stat_int
4055         \__enumext_unskip_unkern: % remove topsep + [partopsep]
4056         \end__enumext_mini_page
4057     }
4058     {
4059         \__enumext_multicols_stop:
4060     }

```

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.

```

4061     \__enumext_after_stop_list:
4062     \__enumext_check_ans_key_hook:
4063     \__enumext_vspace_below:
4064     \bool_set_false:N \l__enumext_standar_bool
4065     \bool_if:NF \l__enumext_print_keyans_cmd_bool
4066     {
4067         \__enumext_standar_save_counter:
4068     }
4069 }

```

(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.

```

4070 \cs_new_protected:Nn \__enumext_set_item_width:
4071 {
4072     \dim_set:Nn \itemwidth { \linewidth }
4073     \dim_compare:nT
4074     {
4075         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
4076     }
4077     {
4078         \dim_sub:Nn \itemwidth
4079         {
4080             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
4081         }
4082     }
4083 }

```

(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}{⟨arg one⟩}{⟨arg two⟩}\setcounter{enumX}`.

🔍 This is described in [processing order of legacysetupcode in the block templates](#) and we will apply the workaround provided by Frank Mittelbach.


```

4084 \cs_new_protected:Nn \__enumext_start_counter:
4085 {
4086   \setcounter { enumX \__enumext_level: }
4087   {
4088     \int_eval:n { \int_use:c { \__enumext_start_ \__enumext_level: _int } - 1 }
4089   }
4090 }

```

(End of definition for __enumext_start_counter:.)

enumext Now create the **enumext** environment based on **list** environment by levels.

```

4091 \NewDocumentEnvironment{enumext}{0}{}
4092 {
4093   \__enumext_safe_exec:
4094   \__enumext_parse_keys:n {#1}
4095   \__enumext_before_list:
4096   \__enumext_start_store_level:
4097   \__enumext_start_list:nn
4098   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
4099   {
4100     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
4101     \__enumext_before_keys_exec:
4102   }
4103   \__enumext_start_counter:
4104   \__enumext_set_item_width:
4105   \__enumext_after_args_exec:
4106 }
4107 {
4108   \__enumext_second_part:
4109 }

```

(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**.

```

4110 \__enumext_after_env:nn {enumext}
4111 {
4112   \__enumext_execute_after_env:
4113 }

```

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.

```

4114 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
4115 {
4116   \bool_if:NF \__enumext_store_active_bool
4117   {
4118     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
4119   }
4120   \int_incr:N \__enumext_keyans_level_int
4121   \bool_set_true:N \__enumext_keyans_env_bool
4122   \__enumext_keyans_name_and_start:
4123   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
4124   \bool_set_false:N \__enumext_store_active_bool
4125   \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
4126   {
4127     \msg_error:nn { enumext } { keyans-nested }
4128   }
4129   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4130   {
4131     \msg_error:nn { enumext } { keyans-wrong-level }
4132   }
4133 }

```

(End of definition for __enumext_keyans_safe_exec:.)

\\enumext_keyans_parse_keys:n

Parse [$\langle key = val \rangle$] for `keyans` environment.

```
4134 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
4135 {
4136   \keys_set:nn { enumext / keyans } {#1}
4137 }
```

(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:

```
4138 \cs_new_protected:Nn \__enumext_before_list_v:
```

\\enumext_keyans_multicols_stop:

```
4139 {
```

\\enumext_second_part_v:

```
4140   \__enumext_vspace_above_v:
```

```
4141   \__enumext_before_args_exec_v:
```

```
4142   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
```

```
4143   {
```

```
4144     \dim_set:Nn \l__enumext_minipage_left_v_dim
```

```
4145     {
```

```
4146       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
```

```
4147     }
```

```
4148     \bool_set_true:N \l__enumext_minipage_active_v_bool
```

```
4149     \int_gincr:N \g__enumext_minipage_stat_int
```

```
4150     \__enumext_keyans_minipage_add_space:
```

```
4151     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
```

```
4152   }
```

```
4153   \__enumext_keyans_multicols_start:
```

```
4154 }
```

```
4155 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
```

```
4156 {
```

```
4157   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
```

```
4158   {
```

```
4159     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
```

```
4160     {
```

```
4161       \dim_set:Nn \l__enumext_columns_sep_v_dim
```

```
4162       {
```

```
4163         (
```

```
4164           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
```

```
4165         ) / \l__enumext_columns_v_int
```

```
4166         - \l__enumext_listoffset_v_dim
```

```
4167       }
```

```
4168     }
```

```
4169     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
```

```
4170     \dim_zero:N \columnseprule % no rule here
```

```
4171     \bool_if:NF \l__enumext_minipage_active_v_bool
```

```
4172     {
```

```
4173       \skip_zero:N \multicolsep
```

```
4174       \__enumext_keyans_multi_addvspace:
```

```
4175     }
```

```
4176     \raggedcolumns
```

```
4177     \begin{multicols}{\l__enumext_columns_v_int }
```

```
4178   }
```

```
4179 }
```

```
4180 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
```

```
4181 {
```

```
4182   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
```

```
4183   {
```

```
4184     \__enumext_stop_list:
```

```
4185     \end{multicols}
```

```
4186     \__enumext_unskip_unkern:
```

```
4187     \__enumext_unskip_unkern:
```

```
4188     \par\addvspace{ \l__enumext_multicols_below_v_skip }
```

```
4189   }
```

```
4190   {
```

```
4191     \__enumext_stop_list:
```

```
4192   }
```

```
4193 }
```

```
4194 \cs_new_protected:Nn \__enumext_second_part_v:
```

```
4195 {
```

```
4196   \bool_if:NTF \l__enumext_minipage_active_v_bool
```

```
4197   {
```

```
4198     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
```

```
4199     {
```

```
4199     }
```

```

4200         \msg_warning:nn { enumext } { missing-miniright }
4201         \miniright
4202     }
4203     \int_gzero:N \g__enumext_minipage_stat_int
4204     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4205     \end__enumext_mini_page
4206     \par\addvspace{ \l__enumext_minipage_after_skip }
4207 }
4208 {
4209     \__enumext_keyans_multicols_stop:
4210 }
4211 \bool_set_false:N \l__enumext_keyans_env_bool
4212 \__enumext_after_stop_list_v:
4213 \__enumext_vspace_below_v:
4214 }

```

(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.

```

4215 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4216 {
4217     \dim_set:Nn \itemwidth { \linewidth }
4218     \dim_compare:nT
4219     {
4220         \l__enumext_listoffset_v_dim != \c_zero_dim
4221     }
4222     {
4223         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4224     }
4225 }

```

(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}.

```

4226 \cs_new_protected:Nn \__enumext_keyans_start_counter:
4227 {
4228     \setcounter { enumXv } { \int_eval:n { \int_use:c { \l__enumext_start_v_int } - 1 } }
4229 }

```

(End of definition for __enumext_keyans_start_counter:.)

keyans Now we define the environment **keyans** also based on lists.

```

4230 \NewDocumentEnvironment{keyans}{0}{}
4231 {
4232     \__enumext_keyans_safe_exec:
4233     \__enumext_keyans_parse_keys:n {#1}
4234     \__enumext_before_list_v:
4235     \__enumext_start_list:nn
4236     { \tl_use:N \l__enumext_label_v_tl }
4237     {
4238         \__enumext_list_arg_two_v:
4239         \__enumext_before_keys_exec_v:
4240     }
4241     \__enumext_keyans_start_counter:
4242     \__enumext_keyans_set_item_width:
4243     \__enumext_after_args_exec_v:
4244 }
4245 {
4246     \__enumext_check_starred_cmd:n { item }
4247     \__enumext_second_part_v:
4248 }

```

(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
4249 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4250 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {start-list-tags}
4251 {
4252   \tag_resume:n {#1}
4253   \tag_mc_end_push:
4254     \tag_struct_begin:n {tag=LI}
4255     \tag_struct_begin:n {tag=Lbl}
4256     \tag_mc_begin:n {tag=Lbl}
4257 }
4258 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4259 {
4260   \tag_mc_end:
4261   \tag_struct_end:n {tag=Lbl}
4262   \tag_struct_begin:n {tag=LBody}
4263   \tag_struct_begin:n {tag=text-unit}
4264   \tag_struct_begin:n {tag=text}
4265 }
4266 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4267 {
4268   \tag_struct_end:n {tag=text}
4269   \tag_struct_end:n {tag=text-unit}
4270   \tag_struct_end:n {tag=LBody}
4271   \tag_struct_end:n {tag=LI}
4272   \tag_mc_begin_pop:n {}
4273   \tag_suspend:n {#1}
4274 }
```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

4275 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
4276 {
4277   \IfDocumentMetadataT
4278   {
4279     \socket_assign_plugin:nn {tagsupport/__enumext/starred} {start-list-tags}
4280     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4281   }
4282 }
4283 \cs_new_protected_nopar:Nn __enumext_stop_start_list_tag:
4284 {
4285   \IfDocumentMetadataT
4286   {
4287     \socket_assign_plugin:nn {tagsupport/__enumext/starred} {stop-start-tags}
4288     \socket_use:nn {tagsupport/__enumext/starred} {}
4289   }
4290 }
4291 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
4292 {
4293   \IfDocumentMetadataT
4294   {
4295     \socket_assign_plugin:nn {tagsupport/__enumext/starred} {stop-list-tags}
4296     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4297   }
4298 }
```

(End of definition for `start-list-tags` and others.)

13.44.2 Socket for tagging support in keyanspic

start-list-tags
stop-start-tags
stop-list-tags
\\enumext_anspic_start_list_tag:
\\enumext_anspic_stop_start_list_tag:
\\enumext_anspic_stop_list_tag:

We will first define the necessary sockets and their behavior for `keyanspic` environment.

```

4299 \socket_new:nn {tagsupport/\\enumext/keyanspic}{ 0 }
4300 \socket_new_plug:nnn {tagsupport/\\enumext/keyanspic} {start-list-tags}
4301 {
4302   \tag_resume:n {keyanspic}
4303   \tag_mc_end_push:
4304     \tag_struct_begin:n {tag=LI}
4305     \tag_struct_begin:n {tag=Lbl}
4306     \tag_mc_begin:n {tag=Lbl}
4307 }
4308 \socket_new_plug:nnn {tagsupport/\\enumext/keyanspic} {stop-start-tags}
4309 {
4310   \tag_mc_end:
4311   \tag_struct_end:n {tag=Lbl}
4312   \tag_struct_begin:n {tag=LBody}
4313   \tag_struct_begin:n {tag=text-unit}
4314   \tag_struct_begin:n {tag=text}
4315   \tag_mc_begin:n {tag=text}
4316 }
4317 \socket_new_plug:nnn {tagsupport/\\enumext/keyanspic} {stop-list-tags}
4318 {
4319   \tag_mc_end:
4320   \tag_struct_end:n {tag=text}
4321   \tag_struct_end:n {tag=text-unit}
4322   \tag_struct_end:n {tag=LBody}
4323   \tag_struct_end:n {tag=LI}
4324   \tag_mc_begin_pop:n {}
4325   \tag_suspend:n {keyanspic}
4326 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

4327 \cs_new_protected_nopar:Nn \\enumext_anspic_start_list_tag:
4328 {
4329   \IfDocumentMetadataT
4330   {
4331     \socket_assign_plug:nn {tagsupport/\\enumext/keyanspic} {start-list-tags}
4332     \socket_use:n {tagsupport/\\enumext/keyanspic}
4333   }
4334 }
4335 \cs_new_protected_nopar:Nn \\enumext_anspic_stop_start_list_tag:
4336 {
4337   \IfDocumentMetadataT
4338   {
4339     \socket_assign_plug:nn {tagsupport/\\enumext/keyanspic} {stop-start-tags}
4340     \socket_use:n {tagsupport/\\enumext/keyanspic}
4341   }
4342 }
4343 \cs_new_protected_nopar:Nn \\enumext_anspic_stop_list_tag:
4344 {
4345   \IfDocumentMetadataT
4346   {
4347     \socket_assign_plug:nn {tagsupport/\\enumext/keyanspic} {stop-list-tags}
4348     \socket_use:n {tagsupport/\\enumext/keyanspic}
4349   }
4350 }

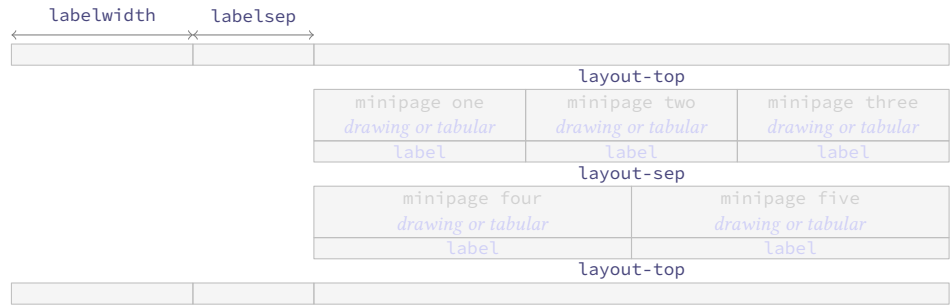
```

(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.

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?](#).

Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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 231 First we define the key that allows us to process the position of the <label> centered “above” or “below” which
label-sep 232 will be label-pos, the vertical separation of these from drawing or tabular will be handled with the key
layout-sty 233 label-sep. The “layout style” will be handled with the key layout-sty will take two values separated
layout-sep 234 by comma {<n° upper, n° lower>} and will determine the number of minipage environments in which all
layout-top 235 arguments of \anspic will be printed at the “upper” and “lower” within the environments separated by the
mark-ans 236 value of the key layout-sep. The vertical space “top” and “bottom” of the environment will be handled with
mark-pos 237 the key layout-top.
mark-sep 238
save-sep 239
wrap-opt 240
wrap-ans* 241
show-ans 242
show-pos 243

4351 \keys_define:nn { enumext / keyanspic }
4352 {
4353   label-pos .choice:,
4354   label-pos / above .code:n =
4355     \bool_set_true:N \l__enumext_anspic_label_above_bool
4356     \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4357   label-pos / below .code:n =
4358     \bool_set_false:N \l__enumext_anspic_label_above_bool
4359     \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4360   label-pos / unknown .code:n =
4361     \msg_error:nnee { enumext } { unknown-choice }
4362     { label-pos } { above,~ below } { \exp_not:n {#1} },
4363   label-pos .initial:n = below,
4364   label-pos .value_required:n = true,
4365   label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4366   label-sep .value_required:n = true,
4367   layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4368   layout-sty .value_required:n = true,
4369   layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4370   layout-sep .value_required:n = true,
4371   layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4372   layout-top .value_required:n = true,
4373   mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4374   mark-ans .value_required:n = true,
4375   mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4376   mark-pos .value_required:n = true,
4377   mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4378   mark-sep .value_required:n = true,
4379   save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4380   save-sep .value_required:n = true,
4381   wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4382   wrap-opt .value_required:n = true,
4383   wrap-ans* .code:n = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4384   wrap-ans* .value_required:n = true,
4385   show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4386   show-ans .value_required:n = true,
4387   show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4388   show-pos .value_required:n = true,
4389   unknown .code:n = {
4390     \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4391     \__enumext_keyans_unknown_keys:n {#1}
4392   },
4393 }

```

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

```

\__enumext_keyans_pic_safe_exec:
\__enumext_keyans_pic_parse_keys:n
\__enumext_keyans_pic_skip_abs:N
\__enumext_keyans_pic_arg_two:

```

The function `__enumext_keyans_pic_safe_exec`: check the nested level position inside the `enumext` environment.

```

4394 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4395 {
4396   \int_incr:N \l__enumext_keyans_pic_level_int
4397   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4398   {
4399     \msg_error:nn { enumext } { keyanspic-nested }
4400   }
4401   \__enumext_keyans_name_and_start:
4402 }

```

Parse [*key = val*] for `keyanspic` environment.

```

4403 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4404 {
4405   \tl_if_novalue:nF {#1}
4406   {
4407     \keys_set:nn { enumext / keyanspic } {#1}
4408   }
4409 }

```

The function `__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep` from `keyans` environment.

```

4410 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4411 {
4412   \dim_compare:nNnT { #1 } < { \c_zero_dim }
4413   {
4414     \skip_set:Nn #1 { -#1 }
4415   }
4416 }

```

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 `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to “false”, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```

4417 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4418 {
4419   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4420   \__enumext_list_arg_two_v:
4421   \__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 `\l__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*.

```

4422   \bool_if:NF \l__enumext_anspic_label_above_bool
4423   {
4424     \stepcounter { enumXv }
4425     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4426     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4427     {
4428       \box_ht_plus_dp:N \l__enumext_anspic_label_box
4429     }
4430     \skip_add:Nn \parsep
4431     {
4432       \l__enumext_anspic_label_htdp_dim
4433       + \box_dp:N \strutbox
4434       + \l__enumext_anspic_label_sep_skip
4435     }
4436   }

```

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.

```

4437   \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4438   \ignorespaces
4439   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4440   \dim_zero:N \listparindent
4441   \skip_zero:N \partopsep
4442   \skip_zero:N \itemsep
4443 }

```

(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.

```

4444 \NewDocumentEnvironment{keyanspic}{ o }
4445 {
4446   \__enumext_keyans_pic_safe_exec:
4447   \__enumext_keyans_pic_parse_keys:n {#1}
4448   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4449   \IfDocumentMetadataT
4450   {
4451     \tag_suspend:n {list}
4452   }
4453   \item[] \scan_stop:
4454   \RenewDocumentCommand \item {}
4455   {
4456     \msg_error:nn { enumext } { keyanspic-item-cmd }
4457   }
4458   \IfDocumentMetadataT
4459   {
4460     \tag_resume:n {keyanspic}
4461     \tag_tool:n {para/tagging=false}
4462     \tag_suspend:n {keyanspic}
4463   }
4464 }

```

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.

```

4465 {
4466   \IfDocumentMetadataT
4467   {
4468     \tag_resume:n {keyanspic}
4469     \tag_mc_end_push:
4470     \tag_struct_begin:n {tag=L,attribute=enumerate}
4471   }
4472   \__enumext_anspic_exec:
4473   \IfDocumentMetadataT
4474   {
4475     \tag_suspend:n {keyanspic}
4476   }
4477   \end{list}
4478   \IfDocumentMetadataT
4479   {
4480     \tag_struct_end:n {tag=L}
4481     \tag_mc_begin_pop:n {}
4482     \tag_struct_end:n {tag=L}
4483     \tag_mc_begin_pop:n {}
4484   }

```

Finally we check if `\anspic*` has been used, set the counter `enumXvi` to zero and apply our “adjusted” vertical space bottom.

```

4485   \__enumext_check_starred_cmd:n { anspic }
4486   \setcounter { enumXvi } { 0 }
4487   \bool_if:NTF \__enumext_anspic_label_above_bool
4488   {
4489     \par\addvspace{ 0.5\box_dp:N \strutbox }
4490   }
4491   {
4492     \par
4493     \addvspace
4494     {
4495       \dim_eval:n
4496       {
4497         \__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4498         + \__enumext_anspic_label_sep_skip + \__enumext_topsep_v_skip
4499       }
4500     }
4501   }
4502 }

```

(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.

```

4503 \NewDocumentCommand \anspic { s o +m }
4504 {
4505   \bool_if:NF \l__enumext_store_active_bool
4506   {
4507     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4508   }
4509   \int_compare:nNt { \l__enumext_level_int } > { 1 }
4510   {
4511     \msg_error:nn { enumext } { keyanspic-wrong-level }
4512   }
4513   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4514   {
4515     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4516   }
4517   \seq_put_right:Nn \l__enumext_anspic_args_seq
4518   {
4519     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4520   }
4521 }

```

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”.

```

4522 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4523 {
4524   \bool_if:NF \l__enumext_anspic_label_above_bool
4525   {
4526     \IfDocumentMetadataT
4527     {
4528       \tag_suspend:n {keyanspic}
4529     }
4530     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4531     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4532     {
4533       \box_ht_plus_dp:N \l__enumext_anspic_body_box
4534     }
4535     \IfDocumentMetadataT
4536     {
4537       \tag_resume:n {keyanspic}
4538     }
4539   }
4540 }

```

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.

```

4541 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4542 {
4543   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4544   {
4545     \bool_if:nTF { #1 }
4546     {
4547       \bool_set_true:N \l__enumext_item_wrap_key_bool
4548       \bool_set_true:N \l__enumext_wrap_label_v_bool
4549       \__enumext_keyans_save_item_opt:n { #2 }
4550       \__enumext_keyans_addto_prop:n { #2 }
4551       \__enumext_keyans_store_ref:
4552       \__enumext_keyans_addto_seq:n { #2 }

```

```

4553         \int_gincr:N \g__enumext_check_starred_cmd_int
4554         \__enumext_keyans_show_ans:
4555         \__enumext_keyans_show_pos:
4556         \makebox[ \l__enumext_labelwidth_v_dim ][c]
4557         {
4558             \tl_use:N \l__enumext_label_font_style_v_tl
4559             \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4560         }
4561         \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4562         \__enumext_keyans_show_item_opt:
4563     }
4564     {
4565         \bool_set_false:N \l__enumext_item_wrap_key_bool
4566         \tl_use:N \l__enumext_label_font_style_v_tl
4567         \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4568     }
4569 }
4570 }

```

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.

```

4571 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4572 {
4573     \stepcounter { enumXvi }
4574     \__enumext_anspic_body_dim:n { #3 }
4575     \bool_if:NTF \l__enumext_anspic_label_above_bool
4576     {
4577         \__enumext_anspic_label:nn { #1 } { #2 }
4578     }
4579     {
4580         \raisebox
4581         {
4582             -\dim_eval:n
4583             {
4584                 \l__enumext_anspic_label_htdp_dim
4585                 + \l__enumext_anspic_body_htdp_dim
4586                 + \box_dp:N \strutbox
4587                 + \l__enumext_anspic_label_sep_skip
4588             }
4589         }
4590         [ opt ] [ opt ]
4591         {
4592             \__enumext_anspic_label:nn { #1 } { #2 }
4593         }
4594     }
4595 }
4596 %

```

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.

```

4597 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4598 {
4599     \__enumext_anspic_start_list_tag:
4600     \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4601     \__enumext_anspic_stop_start_list_tag:
4602     \bool_if:NTF \l__enumext_anspic_label_above_bool
4603     {
4604         \\[\l__enumext_anspic_label_sep_skip] #3
4605     }
4606     {
4607         \\ #3
4608     }
4609     \__enumext_anspic_stop_list_tag:
4610 }

```

The value $\{ \langle n^{\circ} \text{upper}, n^{\circ} \text{lower} \rangle \}$ 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`.

```

4611 \cs_new_protected:Nn \__enumext_anspic_print:n
4612 {
4613     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4614 }

```

```
4615 \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.

```
4616 \cs_new_protected:Nn \__enumext_anspic_row:n
4617 {
4618   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4619   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4620   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4621   \int_step_inline:nnn
4622     { \l__enumext_anspic_above_int + 1 }
4623     { \l__enumext_anspic_below_int }
4624     {
4625       \IfDocumentMetadataT
4626       {
4627         \tag_suspend:n {minipage}
4628       }
4629       \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4630         \centering
4631         \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4632       \end{minipage}
4633       \IfDocumentMetadataT
4634       {
4635         \tag_resume:n {minipage}
4636       }
4637     }
4638   \par
4639 }
```

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*.

```
4640 \cs_new_protected:Nn \__enumext_anspic_exec:
4641 {
4642   \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4643   {
4644     \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4645   }
4646   {
4647     \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4648   }
4649 }
```

(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 \TeX list environments. The fundamental part of the code is adapted from the `shortlst` 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 \TeX 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

`__enumext_starred_columns_set_vii:`
`__enumext_starred_columns_set_viii:`

We set the default value for the *width of the box* containing the *⟨content⟩* of the items for `enumext*` environment.

```

4650 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4651 {
4652   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4653   {
4654     \dim_set:Nn \__enumext_columns_sep_vii_dim
4655     {
4656       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4657       / \__enumext_columns_vii_int
4658     }
4659   }
4660   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4661   \dim_set:Nn \__enumext_item_width_vii_dim
4662   {
4663     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4664     / \__enumext_columns_vii_int
4665     - \__enumext_labelwidth_vii_dim
4666     - \__enumext_labelsep_vii_dim
4667   }

```

When the key `rightmargin` is active we must adjust the values.

```

4668   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4669   {
4670     \dim_sub:Nn \__enumext_item_width_vii_dim
4671     {
4672       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4673       / \__enumext_columns_vii_int
4674     }
4675     \dim_add:Nn \__enumext_columns_sep_vii_dim
4676     {
4677       \__enumext_rightmargin_vii_dim
4678     }
4679   }
4680 }

```

Same implementation for the `keyans*` environment.

```

4681 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4682 {
4683   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4684   {
4685     \dim_set:Nn \__enumext_columns_sep_viii_dim
4686     {
4687       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
4688       / \__enumext_columns_viii_int
4689     }
4690   }
4691   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
4692   \dim_set:Nn \__enumext_item_width_viii_dim
4693   {
4694     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
4695     / \__enumext_columns_viii_int
4696     - \__enumext_labelwidth_viii_dim
4697     - \__enumext_labelsep_viii_dim
4698   }
4699   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4700   {
4701     \dim_sub:Nn \__enumext_item_width_viii_dim
4702     {
4703       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_vii_int )
4704       / \__enumext_columns_viii_int
4705     }
4706     \dim_add:Nn \__enumext_columns_sep_viii_dim
4707     {
4708       \__enumext_rightmargin_viii_dim
4709     }
4710   }
4711 }

```

(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 *content* passed to `\item(<columns>)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4712 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4713 {
4714   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4715   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4716   {
4717     \msg_warning:nnee { enumext } { item-joined }
4718     { \int_use:N \l__enumext_joined_item_vii_int }
4719     { \int_use:N \l__enumext_columns_vii_int }
4720     \int_set:Nn \l__enumext_joined_item_vii_int
4721     {
4722       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4723     }
4724   }
4725   \int_compare:nNnT
4726   { \l__enumext_joined_item_vii_int }
4727   >
4728   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4729   {
4730     \msg_warning:nnee { enumext } { item-joined-columns }
4731     { \int_use:N \l__enumext_joined_item_vii_int }
4732     {
4733       \int_eval:n
4734       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4735     }
4736     \int_set:Nn \l__enumext_joined_item_vii_int
4737     {
4738       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4739     }
4740   }
4741   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4742   {
4743     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4744     \int_decr:N \l__enumext_joined_item_aux_vii_int
4745     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4746     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4747     \dim_set:Nn \l__enumext_joined_width_vii_dim
4748     {
4749       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4750       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4751         + \l__enumext_columns_sep_vii_dim
4752       ) * \l__enumext_joined_item_aux_vii_int
4753     }
4754     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4755   }
4756   {
4757     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4758     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4759   }
4760 }

```

Same implementation for the `keyans*` environment.

```

4761 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4762 {
4763   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4764   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4765   {
4766     \msg_warning:nnee { enumext } { item-joined }
4767     { \int_use:N \l__enumext_joined_item_viii_int }
4768     { \int_use:N \l__enumext_columns_viii_int }
4769     \int_set:Nn \l__enumext_joined_item_viii_int
4770     {
4771       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4772     }
4773   }
4774   \int_compare:nNnT
4775   { \l__enumext_joined_item_viii_int }
4776   >

```

```

4777 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4778 {
4779   \msg_warning:nnee { enumext } { item-joined-columns }
4780   { \int_use:N \l__enumext_joined_item_viii_int }
4781   {
4782     \int_eval:n
4783     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4784   }
4785   \int_set:Nn \l__enumext_joined_item_viii_int
4786   {
4787     \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4788   }
4789 }
4790 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4791 {
4792   \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4793   \int_decr:N \l__enumext_joined_item_aux_viii_int
4794   \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4795   \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4796   \dim_set:Nn \l__enumext_joined_width_viii_dim
4797   {
4798     \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4799     + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4800       + \l__enumext_columns_sep_viii_dim
4801       ) * \l__enumext_joined_item_aux_viii_int
4802   }
4803   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4804 }
4805 {
4806   \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4807   \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4808 }
4809 }

```

(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:`
`__enumext_stop_mini_vii:`

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_page` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

4810 \cs_new_protected:Nn \__enumext_start_mini_vii:
4811 {
4812   \dim_compare:nNnTF { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4813   {
4814     \dim_set:Nn \l__enumext_minipage_left_vii_dim
4815     {
4816       \linewidth
4817       - \l__enumext_minipage_right_vii_dim
4818       - \l__enumext_minipage_hsep_vii_dim
4819     }
4820     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4821     \dim_gset_eq:NN
4822     \g__enumext_minipage_right_vii_dim
4823     \l__enumext_minipage_right_vii_dim
4824     \__enumext_mini_addvspace_vii:
4825     \nointerlineskip\noindent
4826     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4827   }
4828 }

```

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 `__enumext_after_env:n` to execute the `minipage` on the “right side”. At this point we will execute the `__enumext_stop_list:` and `__enumext_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 `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.47).

```

4829 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4830 {
4831   \bool_if:NNTF \l__enumext_minipage_active_vii_bool

```



```

4832     {
4833       \__enumext_stop_list:
4834       \__enumext_stop_store_level_vii:
4835       \IfDocumentMetadataT { \tag_resume:n {enumext*} }
4836       \end__enumext_mini_page
4837       \hfill
4838       \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4839     }
4840     {
4841       \__enumext_stop_list:
4842       \__enumext_stop_store_level_vii:
4843     }
4844   }

```

(End of definition for `__enumext_start_mini_vii:` and `__enumext_stop_mini_vii:`)

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `minipage` environment on the “right side”. 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.

```

4845 \__enumext_after_env:nn {enumext*}
4846 {
4847   \bool_if:NT \g__enumext_minipage_active_vii_bool
4848   {
4849     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4850     \legacy_if_gset_false:n { @minipage }
4851     \skip_vertical:N \c_zero_skip
4852     \par\addvspace { \g__enumext_minipage_right_skip }
4853     \bool_if:NF \g__enumext_minipage_center_vii_bool
4854     {
4855       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4856       {
4857         \centering
4858       }
4859     }
4860     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4861     {
4862       \tl_use:N \g__enumext_miniright_code_vii_tl
4863     }
4864     \box_use_drop:N \l__enumext_miniright_code_vii_box
4865     \skip_vertical:N \c_zero_skip
4866     \__enumext_endminipage:
4867     \par\addvspace{ \g__enumext_minipage_after_skip }
4868   }
4869   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4870   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4871   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4872   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4873   \bool_gset_false:N \g__enumext_starred_bool
4874 }

```

`__enumext_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

```

4875 \cs_new_protected:Nn \__enumext_start_mini_viii:
4876 {
4877   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4878   {
4879     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4880     {
4881       \linewidth
4882       - \l__enumext_minipage_right_viii_dim
4883       - \l__enumext_minipage_hsep_viii_dim
4884     }
4885     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4886     \dim_gset_eq:NN
4887     \g__enumext_minipage_right_viii_dim
4888     \l__enumext_minipage_right_viii_dim
4889     \__enumext_mini_addvspace_viii:
4890     \nointerlineskip\noindent
4891     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4892   }

```

```

4893   }
4894   \cs_new_protected:Nn \__enumext_stop_mini_viii:
4895   {
4896     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4897     {
4898       \__enumext_stop_list:
4899       \IfDocumentMetadataT { \tag_resume:n {keyans*} }
4900       \end__enumext_mini_page
4901       \hfill
4902       \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4903     }
4904     {
4905       \__enumext_stop_list:
4906     }
4907   }
4908   \__enumext_after_env:nn {keyans*}
4909   {
4910     \bool_if:NT \g__enumext_minipage_active_viii_bool
4911     {
4912       \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4913       \par\addvspace { \g__enumext_minipage_right_skip }
4914       \bool_if:NF \g__enumext_minipage_center_viii_bool
4915       {
4916         \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4917         {
4918           \centering
4919         }
4920       }
4921       \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4922       {
4923         \tl_use:N \g__enumext_miniright_code_viii_tl
4924       }
4925       \box_use_drop:N \l__enumext_miniright_code_viii_box
4926       \end__enumext_mini_page
4927       \par\addvspace{ \g__enumext_minipage_after_skip }
4928     }
4929     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4930     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4931     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4932     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4933   }

```

(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.

```

4934 \NewDocumentEnvironment{enumext*}{o }
4935 {
4936   \__enumext_safe_exec_vii:
4937   \__enumext_parse_keys_vii:n {#1}
4938   \__enumext_before_list_vii:
4939   \__enumext_start_store_level_vii:
4940   \__enumext_start_list:nn { }
4941   {
4942     \__enumext_list_arg_two_vii:
4943     \__enumext_before_keys_exec_vii:
4944   }
4945   \setcounter { enumXvii } { \int_eval:n { \int_use:c { \l__enumext_start_vii_int } - 1 } }
4946   \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
4947   \__enumext_starred_columns_set_vii:
4948   \item[] \scan_stop:
4949   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4950   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4951   \ignorespaces
4952 }

```

```

4953 {
4954   \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4955   \__enumext_stop_item_tmp_vii:
4956   \__enumext_remove_extra_parsep_vii:
4957   \__enumext_after_list_vii:
4958 }

```

(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.

```

4959 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4960 {
4961   \__enumext_is_not_nested:
4962   \__enumext_internal_mini_page:
4963   \int_incr:N \l__enumext_level_h_int
4964   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4965   {
4966     \msg_error:nn { enumext } { nested }
4967   }
4968   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4969   {
4970     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4971   }
4972   \bool_set_true:N \l__enumext_starred_bool
4973   \bool_set_false:N \l__enumext_standar_bool
4974   \__enumext_is_on_first_level:
4975 }

```

(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.

```

4976 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4977 {
4978   \tl_if_novalue:nF {#1}
4979   {
4980     \str_clear:N \l__enumext_series_name_str
4981     \keys_set:nn { enumext / enumext* } {#1}
4982     \bool_if:NF \l__enumext_print_keyans_cmd_bool
4983     {
4984       \__enumext_parse_series:n {#1}
4985     }
4986     \__enumext_store_active_keys_vii:n {#1}
4987   }
4988 }

```

(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*`.

```

4989 \cs_new_protected:Nn \__enumext_before_list_vii:
4990 {
4991   \__enumext_vspace_above_vii:
4992   \__enumext_check_ans_active:
4993   \__enumext_before_args_exec_vii:
4994   \__enumext_start_mini_vii:
4995 }

```

(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 `__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

4996 \cs_new_protected:Nn \__enumext_after_list_vii:
4997 {
4998   \__enumext_stop_mini_vii:
4999   \__enumext_after_stop_list_vii:
5000   \__enumext_check_ans_key_hook:
5001   \__enumext_vspace_below_vii:
5002   \bool_set_false:N \__enumext_starred_bool
5003   \bool_if:NF \__enumext_print_keyans_cmd_bool
5004   {
5005     \__enumext_starred_save_counter:
5006   }
5007 }

```

(End of definition for `__enumext_after_list_vii:`.)

`__enumext_start_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`.

`__enumext_stop_store_level_vii:`

```

5008 \cs_new_protected:Nn \__enumext_start_store_level_vii:
5009 {
5010   \bool_if:NT \__enumext_store_active_bool
5011   {
5012     \int_compare:nNnT { \__enumext_level_int } > { 0 }
5013     {
5014       \__enumext_store_level_open_vii:
5015     }
5016   }
5017 }
5018 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
5019 {
5020   \bool_if:NT \__enumext_store_active_bool
5021   {
5022     \int_compare:nNnT { \__enumext_level_int } > { 0 }
5023     {
5024       \__enumext_store_level_close_vii:
5025     }
5026   }
5027 }

```

(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.

```

5028 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
5029 {
5030   \skip_horizontal:n
5031   {
5032     -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim
5033   }
5034   \ignorespaces
5035 }

```

(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`

First we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `__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`.

```

5036 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
5037 {

```

`__enumext_starred_item_vii_aux_ii:w`

`__enumext_starred_item_vii_aux_iii:w`

```

5038     \__enumext_stop_item_tmp_vii:
5039     \int_incr:N \__enumext_item_column_pos_vii_int
5040     \int_gincr:N \g__enumext_item_count_all_vii_int
5041     \__enumext_item_peek_args_vii:
5042 }

```

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).

```

5043 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
5044 {
5045     \peek_meaning:NTF (
5046     { \__enumext_joined_item_vii:w }
5047     { \__enumext_joined_item_vii:w (1) }
5048 }

```

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`.

```

5049 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
5050 {
5051     \__enumext_starred_joined_item_vii:n {#1}
5052     \peek_meaning_remove:NTF *
5053     { \__enumext_starred_item_vii:w }
5054     { \__enumext_standar_item_vii:w }
5055 }

```

The function `__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `__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 `__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 [__enumext_label_vii_tl]`.

```

5056 \cs_new_protected:Npn \__enumext_standar_item_vii:w
5057 {
5058     \bool_set_false:N \__enumext_item_starred_vii_bool
5059     \peek_meaning:NTF [
5060     {
5061         \bool_set_eq:NN \__enumext_wrap_label_vii_bool \__enumext_wrap_label_opt_vii_bool
5062         \__enumext_start_item_vii:w
5063     }
5064     {
5065         \bool_set_true:N \__enumext_wrap_label_vii_bool
5066         \legacy_if_set_true:n { @noitemarg }
5067         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5068     }
5069 }

```

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>]`.

```

5070 \cs_new_protected:Npn \__enumext_starred_item_vii:w
5071 {
5072     \bool_set_true:N \__enumext_item_starred_vii_bool
5073     \bool_set_true:N \__enumext_wrap_label_vii_bool
5074     \peek_meaning:NTF [
5075     { \__enumext_starred_item_vii_aux_i:w }
5076     { \__enumext_starred_item_vii_aux_ii:w }
5077 }
5078 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
5079 {
5080     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
5081     \__enumext_starred_item_vii_aux_ii:w
5082 }
5083 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
5084 {
5085     \peek_meaning:NTF [
5086     { \__enumext_starred_item_vii_aux_iii:w }
5087     {
5088         \dim_set_eq:NN \__enumext_item_symbol_sep_vii_dim \__enumext_labelsep_vii_dim
5089         \legacy_if_set_true:n { @noitemarg }

```

```

5090     \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5091   }
5092 }
5093 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
5094 {
5095   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
5096   \legacy_if_set_true:n { @noitemarg }
5097   \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5098 }

```

(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 `pdflatex` and not if you are running `lualatex`

```

5099 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
5100 {
5101   \legacy_if:nT { @noitemarg }
5102   {
5103     \legacy_if_set_false:n { @noitemarg }
5104     \legacy_if:nT { @nmbrrlist }
5105     {
5106       \IfDocumentMetadataT
5107       {
5108         \bool_if:NT \l__enumext_hyperref_bool
5109         {
5110           \legacy_if_set_true:n { @hyper@item }
5111         }
5112       }
5113       \refstepcounter{enumXvii}
5114       \bool_if:NT \l__enumext_check_answers_bool
5115       {
5116         \int_gincr:N \g__enumext_item_number_int
5117         \bool_set_true:N \l__enumext_item_number_bool
5118       }
5119     }
5120   }
5121   \bool_if:NT \l__enumext_item_starred_vii_bool
5122   {
5123     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
5124     {
5125       \tl_gset_eq:NN
5126       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
5127     }
5128     \mode_leave_vertical:
5129     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
5130     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5131     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
5132     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5133   }
5134   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
5135   {
5136     \tl_use:N \l__enumext_label_font_style_vii_tl
5137     \bool_if:NTF \l__enumext_wrap_label_vii_bool
5138     {
5139       \__enumext_wrapper_label_vii:n {#1}
5140     }
5141     { #1 }
5142   }
5143   \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
5144 }

```

(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
\__enumext_stop_item_vii:
```

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`.

```
5145 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5146 {
5147   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5148   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
5149   {
5150     \l__enumext_joined_width_vii_dim
5151     + \l__enumext_labelwidth_vii_dim
5152     + \l__enumext_labelsep_vii_dim
5153   }
```

Redefine the `\footnote` command.

```
5154 \__enumext_renew_footnote_starred:
```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```
5155 \__enumext_start_list_tag:n {enumext*}
5156 \__enumext_fake_make_label_vii:n {#1}
5157 \__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.

```
5158 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
5159 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5160 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5161 \__enumext_unskip_unkern:
5162 \__enumext_unskip_unkern:
5163 \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5164 \tl_use:N \l__enumext_fake_item_indent_vii_tl
5165 \tl_use:N \l__enumext_after_list_args_vii_tl
5166 }
```

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*.

```
5167 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5168 {
5169   \__enumext_endminipage:
5170   \__enumext_stop_list_tag:n {enumext*}
5171   \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*.

```
5172 \int_set:Nn \hbadness { 10000 }
5173 \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`.

```
5174 \int_compare:nNnTF
5175 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
5176 {
5177   \par\noindent
5178   \int_zero:N \l__enumext_item_column_pos_vii_int
5179 }
5180 {
5181   \skip_horizontal:N \l__enumext_columns_sep_vii_dim
5182 }
5183 }
```

(End of definition for `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:`)

`__enumext_remove_extra_parsep_vii:`

Remove the extra *vertical space* equal to `\parsep=\itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the 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)*.

```
5184 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
5185 {
5186   \int_compare:nNnT
5187   {
5188     \int_mod:nn
5189     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
5190   }
5191   =
5192   { 0 }
5193   {
5194     \para_end:
5195     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
5196     \skip_vertical:N \c_zero_skip
5197     \int_gzero:N \g__enumext_item_count_all_vii_int
5198   }
5199 }
```

(End of definition for `__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 `__enumext_after_env:nn`.

```
5200 \__enumext_after_env:nn {enumext*}
5201 {
5202   \__enumext_execute_after_env:
5203 }
```

13.48 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```
5204 \NewDocumentEnvironment{keyans*}{o}{
5205 {
5206   \__enumext_safe_exec_viii:
5207   \__enumext_parse_keys_viii:n {#1}
5208   \__enumext_before_list_viii:
5209   \__enumext_start_list:nn { }
5210   {
5211     \__enumext_list_arg_two_viii:
5212     \__enumext_before_keys_exec_viii:
5213   }
5214   \setcounter { enumxviii } { \int_eval:n { \int_use:c { \l__enumext_start_viii_int } - 1 } }
5215   \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
5216   \__enumext_starred_columns_set_viii:
5217   \item[] \scan_stop:
5218   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5219   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
5220   \ignorespaces
5221 }
5222 {
5223   \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
5224   \__enumext_stop_item_tmp_viii:
5225   \__enumext_remove_extra_parsep_viii:
5226   \__enumext_check_starred_cmd:n { item }
5227   \__enumext_after_list_viii:
5228 }
```

(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.

```
5229 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5230 {
```

```

5231 \bool_if:NF \l__enumext_store_active_bool
5232 {
5233   \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
5234 }
5235 \int_incr:N \l__enumext_keyans_level_h_int
5236 \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5237 {
5238   \msg_error:nn { enumext } { nested }
5239 }
5240 \__enumext_keyans_name_and_start:
5241 \bool_if:NT \l__enumext_starred_bool
5242 {
5243   \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5244 }
5245 \bool_set_true:N \l__enumext_starred_bool
5246 % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5247 \bool_set_false:N \l__enumext_store_active_bool
5248 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5249 {
5250   \msg_error:nn { enumext } { keyans-wrong-level }
5251 }
5252 }

```

(End of definition for `__enumext_safe_exec_viii:`)

```

\__enumext_parse_keys_viii:n Parse [key = val] for keyans*.
5253 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5254 {
5255   \tl_if_novalue:nF {#1}
5256   {
5257     \keys_set:nn { enumext / keyans* } {#1}
5258   }
5259 }

```

(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`.

```

5260 \cs_new_protected:Nn \__enumext_before_list_viii:
5261 {
5262   \__enumext_vspace_above_viii:
5263   \__enumext_before_args_exec_viii:
5264   \__enumext_start_mini_viii:
5265 }

```

(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.

```

5266 \cs_new_protected:Nn \__enumext_after_list_viii:
5267 {
5268   \__enumext_stop_mini_viii:
5269   \__enumext_after_stop_list_viii:
5270   \__enumext_vspace_below_viii:
5271 }

```

(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* (`\number`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\label` next to the `[\content]` if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[\content]`, `\item(\number)*` and `\item(\number)*[\content]` commands.

`__enumext_first_item_tmp_viii:` The `__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the “*first*” `\item` in the environment at the point of execution of this function, where it is equal to the `__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```

5272 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5273 {
5274     \skip_horizontal:n
5275     {
5276         -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
5277     }
5278     \ignorespaces
5279 }

```

(End of definition for `__enumext_first_item_tmp_viii:`)

`__enumext_start_item_tmp_viii:` 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 `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. `__enumext_item_peek_args_viii:` After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

`__enumext_item_peek_args_viii:`
`__enumext_joined_item_viii:w`
`__enumext_standar_item_viii:w`

```

5280 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5281 {
5282     \__enumext_stop_item_tmp_viii:
5283     \int_incr:N \l__enumext_item_column_pos_viii_int
5284     \int_gincr:N \g__enumext_item_count_all_viii_int
5285     \__enumext_item_peek_args_viii:
5286 }

```

The function `__enumext_item_peek_args_viii:` will handle the `\item(<number>)`. Look for the argument “(”, if it is present we will call the function `__enumext_joined_item_viii: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).

```

5287 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5288 {
5289     \peek_meaning:NTF (
5290     { \__enumext_joined_item_viii:w }
5291     { \__enumext_joined_item_viii:w (1) }
5292 }

```

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`.

```

5293 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5294 {
5295     \__enumext_starred_joined_item_viii:n {#1}
5296     \peek_meaning_remove:NTF *
5297     { \__enumext_starred_item_viii:w }
5298     { \__enumext_standar_item_viii:w }
5299 }

```

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[<custom>]` 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@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_viii:w [\l__enumext_label_viii_tl]`.

```

5300 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5301 {
5302     \bool_set_false:N \l__enumext_item_starred_viii_bool
5303     \bool_set_false:N \l__enumext_item_wrap_key_bool
5304     \peek_meaning:NTF [
5305     {
5306         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
5307         \__enumext_start_item_viii:w
5308     }
5309     {
5310         \bool_set_true:N \l__enumext_wrap_label_viii_bool
5311         \legacy_if_set_true:n { @noitemarg }
5312         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5313     }
5314 }

```

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

```
\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_keyans_starred_item_star:
```

The function `__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[\langle content \rangle]`.

```
5315 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5316 {
5317   \bool_set_true:N \l__enumext_item_starred_viii_bool
5318   \bool_set_true:N \l__enumext_item_wrap_key_bool
5319   \bool_set_true:N \l__enumext_wrap_label_viii_bool
5320   \peek_meaning:NTF [
5321     { \__enumext_starred_item_viii_aux_i:w }
5322     { \__enumext_starred_item_viii_aux_ii:w }
5323   ]
```

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`.

```
5324 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5325 {
5326   \tl_clear:N \l__enumext_store_current_label_tl
5327   \tl_if_no_value:nF { #1 }
5328   {
5329     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5330     {
5331       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
5332       \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5333     }
5334     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5335   }
5336   \__enumext_starred_item_viii_aux_ii:w
5337 }
5338 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5339 {
5340   \legacy_if_set_true:n { @noitemarg }
5341   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5342 }
```

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.

```
5343 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5344 {
5345   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5346   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5347   \__enumext_keyans_store_ref:
5348   \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
5349   \__enumext_keyans_addto_seq_link:
5350   \int_gincr:N \g__enumext_check_starred_cmd_int
5351   \dim_compare:nNt { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5352   {
5353     \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5354   }
5355   \bool_if:NT \l__enumext_show_answer_bool
5356   {
5357     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5358     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5359     \__enumext_print_keyans_box:NN
5360     \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5361   }
5362   \bool_if:NT \l__enumext_show_position_bool
5363   {
5364     \tl_set:Ne \l__enumext_mark_answer_sym_tl
5365     {
5366       \group_begin:
5367       \exp_not:N \normalfont
5368       \exp_not:N \footnotesize [ \int_eval:n
5369       {
5370         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5371       }
5372     }
```

```

5372         ]
5373         \group_end:
5374     }
5375     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5376     \__enumext_print_keyans_box:NN
5377         \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5378     }
5379 }

```

(End of definition for `__enumext_starred_item_viii:w` and others.)

```

\__enumext_keyans_wrapper_label_viii:n
\__enumext_fake_make_label_viii:n

```

The implementation at this is very similar to that of the `enumext*` environment.

```

5380 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5381 {
5382     \bool_lazy_all:nT
5383     {
5384         { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5385         { \bool_if_p:N \l__enumext_show_answer_bool }
5386         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5387         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5388     }
5389     {
5390         \cs_set_eq:NN
5391             \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5392     }
5393     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5394     {
5395         \__enumext_wrapper_label_viii:n {#1}
5396     }
5397     { #1 }
5398 }
5399 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5400 {
5401     \legacy_if:nT { @noitemarg }
5402     {
5403         \legacy_if_set_false:n { @noitemarg }
5404         \legacy_if:nT { @nmbrrlist }
5405         {
5406             \refstepcounter{enumXviii}
5407         }
5408     }
5409     \bool_if:NT \l__enumext_item_starred_viii_bool
5410     {
5411         \__enumext_keyans_starred_item_star:
5412     }
5413     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5414     {
5415         \tl_use:N \l__enumext_label_font_style_viii_tl
5416         \__enumext_keyans_wrapper_label_viii:n {#1}
5417     }
5418     \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5419 }

```

(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*`

```

\__enumext_start_item_viii:w
\__enumext_stop_item_viii:

```

The implementation at this is very similar to that of the `enumext*` environment.

```

5420 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5421 {
5422     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5423     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5424     {
5425         \l__enumext_joined_width_viii_dim
5426         + \l__enumext_labelwidth_viii_dim
5427         + \l__enumext_labelsep_viii_dim
5428     }
5429     \__enumext_renew_footnote_starred:
5430     \__enumext_start_list_tag:n {keyans*}
5431     \__enumext_fake_make_label_viii:n {#1}
5432     \__enumext_stop_start_list_tag:
5433     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }

```

```

5434 \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5435 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5436 \__enumext_unskip_unkern:
5437 \__enumext_unskip_unkern:
5438 \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5439 \tl_use:N \l__enumext_fake_item_indent_viii_tl
5440 \bool_if:NT \l__enumext_item_starred_viii_bool
5441 {
5442   \__enumext_keyans_show_item_opt_viii:
5443 }
5444 \tl_use:N \l__enumext_after_list_args_viii_tl
5445 }
5446 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5447 {
5448   \__enumext_endminipage:
5449   \__enumext_stop_list_tag:n {keyans*}
5450   \hbox_set_end:
5451   \int_set:Nn \hbadness { 10000 }
5452   \box_use_drop:N \l__enumext_item_text_viii_box
5453   \int_compare:nNnTF
5454   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5455   {
5456     \par\noindent
5457     \int_zero:N \l__enumext_item_column_pos_viii_int
5458   }
5459   {
5460     \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5461   }
5462 }

```

(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.

```

5463 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5464 {
5465   \int_compare:nNnT
5466   {
5467     \int_mod:nn
5468     { \g__enumext_item_count_all_viii_int }
5469     { \l__enumext_columns_viii_int }
5470   }
5471   =
5472   { 0 }
5473   {
5474     \para_end:
5475     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5476     \skip_vertical:N \c_zero_skip
5477     \int_gzero:N \g__enumext_item_count_all_viii_int
5478   }
5479 }

```

(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.

```

5480 \NewDocumentCommand \getkeyans { m }
5481 {
5482   \exp_args:Ne \__enumext_getkeyans_aux:n
5483   { \tl_to_str:e { \text_expand:n {#1} } }
5484 }

```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```

5485 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5486 {
5487   \str_if_in:nnTF {#1} { : }
5488   {
5489     \use:e
5490     {

```

```

5491         \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5492         { {##1} {##2} }
5493     }
5494     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5495 }
5496 { \msg_error:nnn { enumext } { missing-colon } {#1} }
5497 }

```

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*.

```

5498 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5499 {
5500     \prop_if_exist:cTF { g__enumext_#1_prop }
5501     {
5502         \prop_item:cn { g__enumext_#1_prop }{#2}
5503     }
5504     {
5505         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5506     }
5507 }

```

(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⟩]`.

```

5508 \keys_define:nn { enumext / print }
5509 {
5510     print* .code:n      = \keys_precompile:neN { enumext / enumext* }
5511                       { \__enumext_filter_save_key:n {#1} }
5512                       \l__enumext_print_keyans_starred_tl, % starred cmd
5513     print* .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5514                           rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*,
5515                           columns=2, first=\small, font=\small },
5516     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5517                       { \__enumext_filter_save_key:n {#1} }
5518                       \l__enumext_print_keyans_i_tl,
5519     print-1 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5520                           rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*,
5521                           columns=2, first=\small, font=\small },
5522     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5523                       { \__enumext_filter_save_key:n {#1} }
5524                       \l__enumext_print_keyans_ii_tl,
5525     print-2 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5526                           rightmargin=0pt, listparindent=0pt, nosep, label=(\alph*),
5527                           first=\small, font=\small },
5528     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5529                       { \__enumext_filter_save_key:n {#1} }
5530                       \l__enumext_print_keyans_iii_tl,
5531     print-3 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5532                           rightmargin=0pt, listparindent=0pt, nosep, label=\roman*,
5533                           first=\small, font=\small },
5534     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5535                       { \__enumext_filter_save_key:n {#1} }
5536                       \l__enumext_print_keyans_iv_tl,
5537     print-4 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5538                           rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*,
5539                           first=\small, font=\small },
5540     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5541                       { \__enumext_filter_save_key:n {#1} }
5542                       \l__enumext_print_keyans_vii_tl, % starred nested

```



```

5543     print-* .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5544                           rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*,
5545                           first=\small, font=\small },
5546   }

```

- 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`.

```

5547 \NewDocumentCommand \printkeyans { s O{} m }
5548 {
5549   \group_begin:
5550     \bool_set_true:N \__enumext_print_keyans_cmd_bool
5551     \tl_use:N \__enumext_print_keyans_i_tl
5552     \tl_use:N \__enumext_print_keyans_ii_tl
5553     \tl_use:N \__enumext_print_keyans_iii_tl
5554     \tl_use:N \__enumext_print_keyans_iv_tl
5555     \tl_use:N \__enumext_print_keyans_vii_tl
5556     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5557     \bool_set_false:N \__enumext_print_keyans_cmd_bool
5558   \group_end:
5559 }

```

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.

```

5560 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5561 {
5562   \seq_if_exist:cTF { g__enumext_#3_seq }
5563   {
5564     \seq_if_empty:cF { g__enumext_#3_seq }
5565     {

```

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.

```

5566     \bool_if:nTF {#1}
5567     {
5568       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5569       {
5570         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5571       }
5572       {
5573         \tl_use:N \__enumext_print_keyans_starred_tl
5574         \bool_set_true:N \__enumext_base_line_fix_bool
5575         \bool_set_true:N \__enumext_print_keyans_star_bool
5576         \begin{enumext*}[#2]
5577           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5578           \end{enumext*}
5579         \bool_set_false:N \__enumext_base_line_fix_bool
5580         \bool_set_false:N \__enumext_print_keyans_star_bool
5581       }
5582     }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “*first level*” then map the *sequence*.

```

5583     {
5584       \begin{enumext}[#2]
5585       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5586       \end{enumext}
5587     }
5588   }
5589   {
5590     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5591   }
5592 }
5593 }

```

(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.

```
5594 \cs_new:Npn \__enumext_filter_level:n #1
5595 {
5596   \use:e
5597   {
5598     \keyval_parse:NNn
5599       \__enumext_filter_level_key:n
5600       \__enumext_filter_level_pair:nn {#1}
5601   }
5602 }
```

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.

```
5603 \cs_new:Npn \__enumext_filter_level_key:n #1
5604 {
5605   \str_case:nnF {#1}
5606   {
5607     { resume* } {} { reset } {} { reset* } {}
5608   }
5609   { , { \exp_not:n {#1} } }
5610 }
```

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.

```
5611 \cs_new:Npn \__enumext_filter_level_pair:nn #1#2
5612 {
5613   \str_case:nnF {#1}
5614   {
5615     { series } {} { save-ans } {} { resume } {}
5616   }
5617   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5618 }
```

(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`.

```
5619 \keys_define:nn { enumext / meta-families }
5620 {
5621   enumext-1 .code:n = {
5622     \keys_set:ne { enumext / level-1 }
5623     {
5624       \__enumext_filter_level:n {#1}
5625     }
5626   },
5627   enumext-2 .code:n = {
5628     \keys_set:ne { enumext / level-2 }
5629     {
5630       \__enumext_filter_level:n {#1}
5631     }
5632   },
5633   enumext-3 .code:n = {
5634     \keys_set:ne { enumext / level-3 }
5635     {
5636       \__enumext_filter_level:n {#1}
5637     }
5638   },
5639   enumext-4 .code:n = {
5640     \keys_set:ne { enumext / level-4 }
5641     {
5642       \__enumext_filter_level:n {#1}
5643     }
5644   }
5645 }
```

```

5644     },
5645     enumext* .code:n = {
5646         \keys_set:ne { enumext / enumext* }
5647         {
5648             \__enumext_filter_level:n {#1}
5649         }
5650     },
5651     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} },
5652     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
5653     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } },
5654     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } },
5655     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } },
5656     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } },
5657     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } },
5658     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } },
5659     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
5660 }

```

We store them in the constant sequence `\c__enumext_all_families_seq` separated by commas.

```

5661 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5662 {
5663     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5664     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5665 }

```

`\setenumext` Now we define the user command `\setenumext`.

```

\__enumext_set_parse:n 5666 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
\__enumext_set_error:nn 5667 {
5668     \seq_clear:N \l__enumext_setkey_tmpa_seq
5669     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5670     \int_set:Nn \l__enumext_setkey_tmpa_int
5671     {
5672         \seq_count:N \l__enumext_setkey_tmpb_seq
5673     }
5674     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5675     {
5676         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5677         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5678         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5679         {
5680             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5681         }
5682     }
5683     {
5684         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5685     }
5686     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5687     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5688     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5689     {
5690         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5691     }
5692 }

```

Internal functions used by the `\setenumext` command.

```

5693 \cs_new_protected:Npn \__enumext_set_parse:n #1
5694 {
5695     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5696     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5697     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5698     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5699     {
5700         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5701         { \tl_trim_spaces:n {#1} }
5702     }
5703     { \__enumext_set_error:nn {#1} { } }
5704 }
5705 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5706 { \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
5707 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5708 {
5709   {enumext,1} = level-1,
5710   {enumext,2} = level-2,
5711   {enumext,3} = level-3,
5712   {enumext,4} = level-4,
5713   {enumext*} = enumext*
5714 }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5715 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5716 {
5717   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5718   { \msg_error:nn { enumext } { prohibited-unknown } }
5719   {
5720     \bool_if:nTF {#1}
5721     {
5722       \int_step_inline:nn { 4 }
5723       { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5724       __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5725     }
5726     { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
5727   }
5728 }
```

The internal functions `__enumext_add_meta_key:nnn` and `__enumext_def_meta_key:nnn` will check the *optional argument* and create the “meta-key”.

```
5729 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
5730 {
5731   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5732   \tl_replace_all:Nnn \l__enumext_meta_path_tl {~} {}
5733   \prop_get:NVNTF
5734   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5735   { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5736   {
5737     \msg_error:nnn { enumext } { unknown-set } {#1}
5738     \use_none:nn
5739   }
5740 }
5741 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
5742 {
5743   \bool_lazy_or:nnTF
5744   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5745   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5746   { \msg_error:nnn { enumext } { already-defined } {#2} }
5747   {
5748     \keys_define:nn { enumext / #1 }
5749     {
5750       #2 .meta:n = {#3},
5751       #2 .value_forbidden:n = true
5752     }
5753   }
5754 }
5755 \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`

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.

```
\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
5756 \keys_define:nn { enumext / foreach }
```

```

5757 {
5758   before .tl_set:N = \l__enumext_foreach_before_tl,
5759   before .value_required:n = true,
5760   after  .tl_set:N = \l__enumext_foreach_after_tl,
5761   after  .value_required:n = true,
5762   start  .int_set:N = \l__enumext_foreach_start_int,
5763   start  .value_required:n = true,
5764   stop   .int_set:N = \l__enumext_foreach_stop_int,
5765   stop   .value_required:n = true,
5766   step   .int_set:N = \l__enumext_foreach_step_int,
5767   step   .value_required:n = true,
5768   wrapper .cs_set_protected:Np = \l__enumext_foreach_wrapper:n #1,
5769   wrapper .value_required:n = true,
5770   sep     .tl_set:N = \l__enumext_foreach_sep_tl,
5771   sep     .value_required:n = true,
5772   unknown .code:n = { \l__enumext_parse_foreach_keys:n {#1} }
5773 }
5774 \keys_precompile:nnN { enumext / foreach }
5775 {
5776   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5777 }
5778 \l__enumext_foreach_default_keys_tl

```

Functions for handling unknown $\langle keys \rangle$.

```

5779 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:nn #1#2
5780 {
5781   \tl_if_blank:nTF {#2}
5782   {
5783     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5784   }
5785   {
5786     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5787   }
5788 }
5789 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:n #1
5790 {
5791   \exp_args:NV \l__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5792 }

```

We create the command.

```

5793 \NewDocumentCommand \foreachkeyans { +0{ } m }
5794 {
5795   \l__enumext_foreach_keyans:nn {#1} {#2}
5796 }

```

Finally the internal functions `\l__enumext_foreach_keyans:nn` and `\l__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5797 \cs_new_protected:Npn \l__enumext_foreach_keyans:nn #1 #2
5798 {
5799   \tl_use:N \l__enumext_foreach_default_keys_tl
5800   \keys_set:nn { enumext / foreach } {#1}
5801   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5802   \prop_if_exist:cF { g__enumext_#2_prop }
5803   {
5804     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5805   }
5806   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5807   {
5808     \int_set:Nn \l__enumext_foreach_stop_int
5809     { \prop_count:c { g__enumext_#2_prop } }
5810   }
5811   \seq_clear:N \l__enumext_foreach_print_seq
5812   \int_step_function:nnnN
5813   { \l__enumext_foreach_start_int }
5814   { \l__enumext_foreach_step_int }
5815   { \l__enumext_foreach_stop_int }
5816   \l__enumext_foreach_add_body:n
5817   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5818 }
5819 \cs_new_protected:Npn \l__enumext_foreach_add_body:n #1
5820 {
5821   \seq_put_right:Ne \l__enumext_foreach_print_seq

```

```

5822     {
5823         \exp_not:V \l__enumext_foreach_before_tl
5824         \__enumext_foreach_wrapper:n
5825         {
5826             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5827         }
5828         \exp_not:V \l__enumext_foreach_after_tl
5829     }
5830 }

```

(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.

```

5831 \msg_new:nnn { enumext } { package-load }
5832 {
5833     The~'#1'~package~is~already~loaded.
5834 }
5835 \msg_new:nnn { enumext } { package-not-load }
5836 {
5837     The~'#1'~package~will~be~loaded~as~a~dependency.
5838 }
5839 \msg_new:nnn { enumext } { package-load-foot }
5840 {
5841     The~'#1'~package~is~loaded~with~the~option~'#2'.
5842 }

```

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

```

5843 \msg_new:nnn { enumext } { counters }
5844 {
5845     The~counter~'#1'~is~already~defined~by~some~\\
5846     package~or~macro,~it~cannot~be~continued.
5847 }

```

Message used by `align` and `mark-pos` keys.

```

5848 \msg_new:nnn { enumext } { unknown-choice }
5849 {
5850     The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5851 }

```

Message used by reserved `anskey*` environment by `enumext` package.

```

5852 \msg_new:nnnn { enumext } { anskey-env-error }
5853 {
5854     The~environment~'#1'~is~reserved~by ~\\
5855     'enumext'~package,~It~is~already~defined.
5856 }
5857 {
5858     The~environment~'#1'~is~defined~internally ~
5859     for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
5860 }
5861 \msg_new:nnn { enumext } { anskey-env-nested }
5862 {
5863     The~#1~'#2'~can't~be~nested~\msg_line_context:.
5864 }

```

Message used in the creation of *prop list* by `enumext` package.

```

5865 \msg_new:nnn { enumext } { store-prop }
5866 {
5867     *~Package~enumext:~Creating ~
5868     \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5869 }
5870 \msg_new:nnn { enumext } { store-seq }
5871 {
5872     *~Package~enumext:~Creating ~
5873     \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5874 }
5875 \msg_new:nnn { enumext } { store-int }
5876 {
5877     *~Package~enumext:~Creating ~
5878     \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5879 }
5880 \msg_new:nnn { enumext } { prop-seq-int-hook }

```

```

5881 {
5882     *~Package~enumext:~Elements~in ~
5883     \c_backslash_str g__enumext_#1_prop~::~#2.\\
5884     *~Package~enumext:~Elements~in ~
5885     \c_backslash_str g__enumext_#1_seq~::~#3.\\
5886     *~Package~enumext:~Value~off ~
5887     \c_backslash_str g__enumext_resume_#1_int~::~#4.
5888 }
5889 \msg_new:nnn { enumext } { item-answer-hook }
5890 {
5891     *~Package~enumext:~Value~off ~
5892     \c_backslash_str g__enumext_item_number_int~::~#1.\\
5893     *~Package~enumext:~Value~off ~
5894     \c_backslash_str g__enumext_item_anskey_int~::~#2.\\
5895     *~Package~enumext:~Difference~item_number_int~-~item_anskey_int~::~#3.
5896 }

```

Message used by [*(key = val)*] system and `\setenumext` command.

```

5897 \msg_new:nnn { enumext } { invalid-key }
5898 {
5899     The~key~'#1'~is~not~know~the~level~#2.
5900 }
5901 \msg_new:nnn { enumext } { unknown-key-family }
5902 {
5903     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5904 }

```

Messages used in length calculation.

```

5905 \msg_new:nnn { enumext } { width-negative }
5906 {
5907     Ignoring~negative~value~'#1=#2'~\msg_line_context:.\
5908     The~key~'#1'~ accepts~values ~>::~opt.
5909 }
5910 \msg_new:nnn { enumext } { width-zero }
5911 {
5912     Invalid~'#1=#2'~\msg_line_context:.\
5913     The~key~'#1'~ accepts~values ~>::~opt.
5914 }

```

Messages used by `show-length` key in `enumext`.

```

5915 \msg_new:nnn { enumext } { list-lengths }
5916 {
5917     ****~Lengths~used~by~'#1'~enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\
5918     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5919     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5920     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5921     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5922     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5923     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5924     \__enumext_show_length:nnn { skip } { topsep } {#1}
5925     \__enumext_show_length:nnn { skip } { parsep } {#1}
5926     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5927     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5928     ****~
5929 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

5930 \msg_new:nnn { enumext } { list-lengths-not-nested }
5931 {
5932     ****~Lengths~used~by~'#1'~environment~\msg_line_context:~\c_space_tl ****\\
5933     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5934     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5935     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5936     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5937     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5938     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5939     \__enumext_show_length:nnn { skip } { topsep } {#1}
5940     \__enumext_show_length:nnn { skip } { parsep } {#1}
5941     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5942     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5943     ****~
5944 }

```


Messages used by `ref` key.

```
5945 \msg_new:nnn { enumext } { key-ref-empty }
5946 {
5947     Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5948 }
```

Messages used by `save-ans` key.

```
5949 \msg_new:nnn { enumext } { save-ans-empty }
5950 {
5951     Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5952 }
5953 \msg_new:nnn { enumext } { save-ans-log }
5954 {
5955     *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5956 }
5957 \msg_new:nnn { enumext } { save-ans-log-hook }
5958 {
5959     *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5960 }
5961 \msg_new:nnn { enumext } { save-ans-hook }
5962 {
5963     Stop~storing~for~'save-ans=#1'~\msg_line_context:.
5964 }
```

Messages used by the internal system to check answer used by `check-ans` key.

```
5965 \msg_new:nnn { enumext } { need-save-ans }
5966 {
5967     Key~'#1'~ works~only~with~the~'save-ans'~key~in~'#2'~ \msg_line_context:.
5968 }
5969 \msg_new:nnn { enumext } { items-same-answer }
5970 {
5971     *****\
5972     *~Package~enumext:~Checking~answers~in~'#1' ~
5973     for~\c_left_brace_str #2 \c_right_brace_str\
5974     *~started~#3~and~close~\msg_line_context: : ~
5975     'OK',~all~items~with~answer.\
5976     *****
5977 }
5978 \msg_new:nnn { enumext } { item-greater-answer }
5979 {
5980     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\
5981     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\
5982     Items~>~Answers.
5983 }
5984 \msg_new:nnn { enumext } { item-less-answer }
5985 {
5986     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\
5987     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\
5988     Items~<~Answers.
5989 }
```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```
5990 \msg_new:nnn { enumext } { missing-starred }
5991 {
5992     Missing~'\c_backslash_str #1'~#2.
5993 }
5994 \msg_new:nnn { enumext } { many-starred }
5995 {
5996     Many~'\c_backslash_str #1'~#2.
5997 }
```

Messages used by `\printkeyans*` command.

```
5998 \msg_new:nnn { enumext } { print-starred }
5999 {
6000     \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
6001     #2~environment~ \msg_line_context:.
6002 }
```

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

```
6003 \msg_new:nnn { enumext } { list-too-deep }
6004 {
6005     Too~deep~nesting ~for~'enumext'~\msg_line_context:~ \
6006     The~maximum ~level ~of ~nesting ~is~4.
6007 }
```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

6008 \msg_new:nnn { enumext } { anskey-unnumber-item }
6009 {
6010   Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
6011 }
6012 \msg_new:nnn { enumext } { anskey-already-stored }
6013 {
6014   Content~already~stored~for~this~\c_backslash_str item~\msg_line_context:.
6015 }
6016 \msg_new:nnn { enumext } { anskey-empty-arg }
6017 {
6018   Can't~store~empty~content~\msg_line_context:.
6019 }
6020 \msg_new:nnn { enumext } { anskey-wrong-place }
6021 {
6022   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6023   '\c_backslash_str #1'~works~in~the~environment~'#2'.
6024 }
6025 \msg_new:nnn { enumext } { anskey-nested }
6026 {
6027   The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
6028 }
6029 \msg_new:nnn { enumext } { anskey-math-mode }
6030 {
6031   #1~can't~work~in~math~mode~\msg_line_context:.
6032 }
6033 \msg_new:nnn { enumext } { anskey-env-wrong }
6034 {
6035   The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
6036 }
6037 \msg_new:nnn { enumext } { ansPIC-wrong-place }
6038 {
6039   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6040   '\c_backslash_str #1'~works~in~the~environment~'#2'.
6041 }
6042 \msg_new:nnn { enumext } { command-wrong-place }
6043 {
6044   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6045   '\c_backslash_str #1'~works~outside~the~environment~'#2'.
6046 }
6047 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
6048 {
6049   The~key~'#1'~is~unknown~by~environment~
6050   'anskey*~and~is~being~ignored.
6051 }
6052 {
6053   The~environment~'anskey*~does~not~have~a~key~called ~'#1'.\\
6054   Check~that~you~have~spelled~the~key~name~correctly.
6055 }
6056 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
6057 {
6058   The~key~'#1=#2'~is~unknown~by~environment ~
6059   'anskey*~and~is~being~ignored.
6060 }
6061 {
6062   The~environment~'anskey*~does~not~have~a~key~called ~'#1'.\\
6063   Check~that~you~have~spelled~the~key~name~correctly.
6064 }
6065 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
6066 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
6067 {
6068   The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6069   Check~that~you~have~spelled~the~key~name~correctly.
6070 }
6071 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
6072 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
6073 {
6074   The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6075   Check~that~you~have~spelled~the~key~name~correctly.
6076 }
6077 \msg_new:nnn { enumext } { overwrite-file }

```

```

6078 {
6079     Overwriting~file~'#1'.
6080 }
6081 \msg_new:nnn { enumext } { writing-file }
6082 {
6083     Writing~file~'#1'.
6084 }
6085 \msg_new:nnn { enumext } { not-writing }
6086 {
6087     File~'#1'~already~exists.~Not~writing.
6088 }

```

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

```

6089 \msg_new:nnn { enumext } { keyans-nested }
6090 {
6091     The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
6092 }
6093 \msg_new:nnn { enumext } { keyans-wrong-level }
6094 {
6095     Wrong~level~position~for~'keyans'~\msg_line_context:~ \\
6096     The~environment~'keyans'~can~only~be~in~the~first~level.
6097 }
6098 \msg_new:nnn { enumext } { wrong-place }
6099 {
6100     Wrong~place~for~'#1'~environment ~\msg_line_context:~ \\
6101     '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
6102 }
6103 \msg_new:nnn { enumext } { keyanspic-nested }
6104 {
6105     The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:~.
6106 }
6107 \msg_new:nnn { enumext } { keyanspic-wrong-level }
6108 {
6109     Wrong~level~position~for~'keyanspic'~\msg_line_context:~ \\
6110     The~environment~'keyans'~can~only~be~in~the~first~level.
6111 }
6112 \msg_new:nnn { enumext } { keyanspic-item-cmd }
6113 {
6114     Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
6115 }
6116 \msg_new:nnnn { enumext } { keyans-unknown-key }
6117 {
6118     The~key~'#1'~is~unknown~by~environment~
6119     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6120 }
6121 {
6122     The~environment~'\l__enumext_envir_name_tl'~does~not
6123     ~have~a~key~called ~'#1'.\\
6124     Check~that~you~have~spelled~the~key~name~correctly.
6125 }
6126 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
6127 {
6128     The~key~'#1=#2'~is~unknown~by~environment ~
6129     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6130 }
6131 {
6132     The~environment~'\l__enumext_envir_name_tl'~does~not
6133     ~have~a~key~called ~'#1'.\\
6134     Check~that~you~have~spelled~the~key~name~correctly.
6135 }

```

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

```

6136 \msg_new:nnnn { enumext } { starred-unknown-key }
6137 {
6138     The~key~'#1'~is~unknown~by~environment~
6139     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6140 }
6141 {
6142     The~environment~'\l__enumext_envir_name_tl'~does~not
6143     ~have~a~key~called ~'#1'.\\
6144     Check~that~you~have~spelled~the~key~name~correctly.
6145 }

```

```

6146 \msg_new:nnnn { enumext } { starred-unknown-key-value }
6147 {
6148   The~key~'#1=#2'~is~unknown~by~environment ~
6149   '\l__enumext_envir_name_tl'~and~is~being~ignored.
6150 }
6151 {
6152   The~environment~'\l__enumext_envir_name_tl'~does~not
6153   ~have~a~key~called ~'#1'.\\
6154   Check~that~you~have~spelled~the~key~name~correctly.
6155 }

```

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

```

6156 \msg_new:nnnn { enumext } { standar-unknown-key }
6157 {
6158   The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6159   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6160 }
6161 {
6162   The~environment~'\l__enumext_envir_name_tl'~does~not
6163   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6164   Check~that~you~have~spelled~the~key~name~correctly.
6165 }
6166 \msg_new:nnnn { enumext } { standar-unknown-key-value }
6167 {
6168   The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6169   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6170 }
6171 {
6172   The~environment~'\l__enumext_envir_name_tl'~does~not
6173   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6174   Check~that~you~have~spelled~the~key~name~correctly.
6175 }

```

Message used by unknown *⟨keys⟩* in `\foreachkeyans`.

```

6176 \msg_new:nnnn { enumext } { for-key-unknown }
6177 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
6178 {
6179   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6180   Check~that~you~have~spelled~the~key~name~correctly.
6181 }
6182 \msg_new:nnnn { enumext } { for-key-value-unknown }
6183 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
6184 {
6185   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6186   Check~that~you~have~spelled~the~key~name~correctly.
6187 }

```

Messages used by `\getkeyans` command.

```

6188 \msg_new:nnn { enumext } { undefined-storage-anskey }
6189 {
6190   Storage~named~'#1'~is~not~defined~\msg_line_context:.
6191 }

```

Messages used by `\miniright` command.

```

6192 \msg_new:nnn { enumext } { missing-miniright }
6193 {
6194   Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
6195   The~key~'mini-env'~need~'\c_backslash_str miniright'.
6196 }
6197 \msg_new:nnn { enumext } { wrong-miniright-place }
6198 {
6199   Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:~ \\
6200   Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
6201 }
6202 \msg_new:nnn { enumext } { wrong-miniright-use }
6203 {
6204   Wrong~use~for~'\c_backslash_str miniright'~\msg_line_context:~ \\
6205   '\c_backslash_str miniright'~need~a~key~'mini-env'.
6206 }
6207 \msg_new:nnn { enumext } { wrong-miniright-starred }
6208 {
6209   Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
6210 }

```

```

6211 \msg_new:nnn { enumext } { many-miniright-used }
6212 {
6213   Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
6214 }

```

Messages used by `\setenumextmeta` command.

```

6215 \msg_new:nnn { enumext } { unknown-set }
6216 {
6217   Argument~[#1]~is~unknown~by~ \c_backslash_str setenumextmeta~\msg_line_context:.
6218 }
6219 \msg_new:nnn { enumext } { already-defined }
6220 {
6221   The~key~'#1'~is~already~defined~\msg_line_context:.
6222 }
6223 \msg_new:nnn { enumext } { prohibited-unknown }
6224 {
6225   The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
6226 }

```

Messages used by `enumext*` and `keyans*` environments.

```

6227 \msg_new:nnn { enumext } { nested }
6228 {
6229   The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
6230 }
6231 \msg_new:nnn { enumext } { nested-horizontal }
6232 {
6233   The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_cont
6234 }
6235 \msg_new:nnn { enumext } { item-joined }
6236 {
6237   Items~joined~(#1)~>~#2 ~columns ~\msg_line_context:.
6238 }
6239 \msg_new:nnn { enumext } { item-joined-columns }
6240 {
6241   Not~space~to~join~items~(#1)~>~#2 ~\msg_line_context:.
6242 }

```

Messages used by `resume` key.

```

6243 \msg_new:nnn { enumext } { unknown-series-starred }
6244 {
6245   The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
6246   ~enumext*~environment~ \msg_line_context:.
6247 }
6248 \msg_new:nnn { enumext } { unknown-series-standar }
6249 {
6250   The~series~'#1'~for~the~resume~key~does~not~exist~at~level~\int_use:N \l__enumext_level_int
6251   \c_space_tl of~enumext~environment~ \msg_line_context:.
6252 }
6253 \msg_new:nnnn { enumext-reset } { invalid-clist }
6254 { The~argument~must~have~1~or~2~elements~separated~by~a~comma. }
6255 { Received:~'#1'. }
6256 \msg_new:nnn { enumext-reset } { invalid-single-arg-star }
6257 { The~single~argument~must~be~exactly~'enumext*'~when~using~a~'*. }
6258 \msg_new:nnnn { enumext-reset } { invalid-single-arg-no-star }
6259 { A~single~argument~is~not~allowed~without~a~'*. }
6260 { Received:~'#1'. }
6261 \msg_new:nnnn { enumext-reset } { out-of-range }
6262 { The~number~must~be~exactly~1,~2,~3~or~4. }
6263 { Received:~'#1'. }
6264 \msg_new:nnnn { enumext-reset } { invalid-package }
6265 { The~first~element~must~be~exactly~'enumext'. }
6266 { Received:~'#1'. }

```

13.55 Finish package

Finish package implementation.

```

6267 \file_input_stop:
6268 </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	
\+	220
\-	220
\\	228, 4604, 4607, 5845, 5854, 5859, 5883, 5885, 5892, 5894, 5907, 5912, 5917, 5932, 5971, 5973, 5975, 5980, 5981, 5986, 5987, 6005, 6022, 6039, 6044, 6053, 6062, 6068, 6074, 6095, 6100, 6109, 6123, 6133, 6143, 6153, 6163, 6173, 6179, 6185, 6194, 6199, 6204
A	
above	<u>1726</u>
above*	<u>1726</u>
\addvspace	1293, 1321, 1364, 1367, 1535, 1538, 1635, 1641, 1679, 1685, 1706, 1712, 4016, 4188, 4206, 4489, 4493, 4852, 4867, 4913, 4927
after	<u>1123</u>
align	<u>663</u>
\Alph	44, 48, 49
\Alph	605, 733, 777, 837, 5538
\alph	44, 48, 49
\alph	606, 731, 5526
\anskey	14, 88, 90, <u>3043</u>
anskey*	15, <u>3173</u>
\anspic	18, 116, 120, <u>4503</u>
\anspic*	81
\arabic	44
\arabic	604, 730, 776, 5514, 5520, 5544
B	
base-fix	<u>981</u>
\baselineskip	58
\baselineskip	997, 1008
before	<u>1123</u>
before*	<u>1123</u>
beginpenalty	<u>921</u>
below	<u>1726</u>
below*	<u>1726</u>
bool commands:	
\bool_gset_false:N	339, 340, 341, 4869, 4873, 4929
\bool_gset_true:N	249, 259, 1226, 2476, 2482, 4838, 4870, 4902, 4930
\bool_if:NTF	389, 399, 416, 490, 497, 506, 513, 527, 540, 1748, 1762, 1775, 1786, 1797, 1808, 1819, 1830, 1844, 1858, 1876, 1891, 1938, 2050, 2060, 2421, 2664, 2674, 2754, 2778, 2785, 2809, 2907, 2929, 2969, 2993, 2997, 3047, 3066, 3090, 3142, 3146, 3176, 3194, 3213, 3229, 3252, 3283, 3298, 3370, 3486, 3520, 3556, 3572, 3593, 3739, 3760, 3806, 3849, 3859, 3893, 3898, 3923, 3932, 3971, 3997, 4047, 4065, 4116, 4171, 4196, 4422, 4487, 4505, 4524, 4575, 4602, 4831, 4847, 4853, 4896, 4910, 4914, 4982, 5003, 5010, 5020, 5108, 5114, 5121, 5137, 5231, 5241, 5355, 5362, 5393, 5409, 5440
\bool_if:nTF	1686, 1713, 2240, 3542, 3718, 4545, 5566, 5720
\bool_if_p:N	268, 282, 991, 992, 1004, 1005, 1658, 1988, 1989, 2004, 2005, 2070, 2181, 2182, 2197, 2198, 2434, 2460, 2473, 2474, 2479, 2480, 2842, 2852, 2864, 2879, 2880, 2914, 2955, 2956, 3357, 3358, 3387, 3388, 3400, 3401, 3441, 3442, 3461, 3462, 3752, 3753, 3754, 3944, 3946, 3957, 5384, 5385, 5386
\bool_lazy_all:nTF	266, 280, 989, 2432, 2458, 2840, 2849, 2862, 2877, 3439, 3459, 3759, 3942, 3955, 5382
\bool_lazy_and:nnTF	245, 255, 1003, 1650, 1657, 1987, 2003, 2069, 2180, 2196, 2253, 2472, 2478, 2913, 2920, 2954, 3356
\bool_lazy_or:nnTF	2362, 2369, 3386, 3399, 5743
\bool_new:N	22, 23, 24, 25, 26, 27, 28, 47, 50, 60, 84, 89, 90, 95, 96, 99, 106, 121, 122, 134, 135, 142, 148, 149, 151, 155, 157, 158, 175, 187, 189
\bool_not_p:n	246, 256, 993, 1659, 2851, 2915, 2921, 3945, 3958
\bool_set_eq:NN	3495, 3699, 5061, 5306
\bool_set_false:N	396, 866, 1015, 2406, 2407, 2439, 2444, 2448, 2452, 2465, 3727, 3912, 4064, 4124, 4211, 4358, 4419, 4565, 4973, 5002, 5058, 5247, 5302, 5303, 5557, 5579, 5580
\bool_set_true:N	273, 287, 382, 385, 656, 1030, 1732, 1737, 1863, 1881, 2153, 2160, 2379, 2380, 2696, 2704, 3117, 3489, 3491, 3523, 3525, 3695, 3706, 3720, 3872, 3911, 3951, 3964, 4037, 4121, 4148, 4355, 4547, 4548, 4820, 4885, 4972, 5065, 5072, 5073, 5117, 5245, 5310, 5317, 5318, 5319, 5550, 5574, 5575
box commands:	
\box_dp:N	1581, 1582, 1585, 1592, 1605, 1613, 1619, 1627, 4433, 4439, 4489, 4586
\box_ht:N	1364, 1367, 1378, 1379, 1390, 1392, 1407, 1410, 1418, 1419, 1430, 1432, 1447, 1450, 1457, 1458, 1469, 1471, 1486, 1489, 1535, 1538, 1546, 1547, 1555, 1556, 1568, 1570
\box_ht_plus_dp:N	4428, 4497, 4533
\box_new:N	57, 144, 145, 182, 188
\box_use_drop:N	4864, 4925, 5173, 5452
\box_wd:N	612
break-col	<u>3013</u> , <u>3099</u>
C	
\c	872, 874, 886, 888
c@ internal commands:	
\c@_enumext_resume_i_int	<u>590</u>
\c@_enumext_resume_ii_int	<u>590</u>
\c@_enumext_resume_iii_int	<u>590</u>
\c@_enumext_resume_iv_int	<u>590</u>
\c@_enumext_resume_vii_int	<u>590</u>
\centering	1688, 1715, 4630, 4857, 4918
check-ans	<u>2398</u>
Document class:	
article	50
clist commands:	
\clist_const:Nn	194
\clist_map_function:nN	4613
\clist_map_inline:Nn	662, 920, 936, 1122, 1137, 1218, 1742
\clist_map_inline:nn	36, 45, 53, 65, 73, 86, 98, 137, 166, 193, 598, 640, 693, 713, 1035, 1056, 1232, 1852, 1872, 2280, 2287, 2302, 2346, 2412, 2591, 2661, 2693, 2837, 3292, 3614, 3629, 3676, 3835, 3838, 3840, 3867, 3879, 3882, 3884, 3903, 5696
\columnbreak	88

\columnbreak	2917
columns	1202
columns-sep	1202
\columnsep	109
\columnsep	3992, 4169
\columnseprule	109
\columnseprule	3995, 4170
Commands provide by enumext :	
\anskey	32, 33, 77, 78, 83–87, 89, 90, 95, 109, 129, 138, 140, 148
\anspic*	32, 34, 81, 84, 95, 96, 119, 120, 138, 140
\anspic	34, 85, 116, 120, 148
\foreachkeyans	143, 150
\getkeyans	84, 138, 150
\item*	32, 34, 81, 84, 85, 95, 96, 99, 103, 130, 131, 136, 138, 140
\item	99, 103, 124, 129, 130, 132, 135
\miniright	32, 55, 63, 64, 110, 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	2274, 2275
cs commands:	
\cs_generate_variant:Nn	199, 200, 614, 630, 878, 894, 2277, 2746, 2751, 2827, 3172, 3825, 4615, 5755
\cs_if_exist:NTF	575, 592
\cs_if_exist_p:N	3755, 5387
\cs_new:Nn	214
\cs_new:Npn	224, 2020, 2029, 2037, 2708, 2717, 2725, 5594, 5603, 5611
\cs_new_eq:NN	366, 367, 372, 373, 401, 402, 405, 406
\cs_new_protected:Nn	230, 238, 264, 295, 325, 331, 337, 343, 349, 357, 377, 424, 428, 446, 458, 476, 488, 504, 520, 533, 554, 753, 810, 857, 987, 1138, 1142, 1146, 1150, 1154, 1158, 1162, 1166, 1170, 1174, 1178, 1182, 1186, 1190, 1194, 1198, 1233, 1245, 1278, 1295, 1306, 1323, 1349, 1370, 1495, 1521, 1541, 1574, 1596, 1631, 1637, 1743, 1757, 1771, 1782, 1793, 1804, 1815, 1826, 1889, 1907, 1936, 1951, 1976, 2067, 2166, 2213, 2310, 2320, 2334, 2347, 2352, 2377, 2417, 2427, 2470, 2485, 2492, 2501, 2506, 2511, 2516, 2525, 2530, 2535, 2752, 2776, 2783, 2807, 2814, 2828, 3064, 3083, 3192, 3211, 3242, 3281, 3296, 3324, 3354, 3382, 3395, 3408, 3437, 3450, 3528, 3538, 3549, 3565, 3581, 3714, 3732, 3766, 3778, 3904, 3940, 3969, 3976, 4006, 4023, 4045, 4070, 4084, 4114, 4138, 4155, 4180, 4194, 4215, 4226, 4394, 4597, 4611, 4616, 4640, 4650, 4681, 4810, 4829, 4875, 4894, 4959, 4989, 4996, 5008, 5018, 5043, 5184, 5229, 5260, 5266, 5287, 5343, 5463
\cs_new_protected:Npn	201, 202, 206, 210, 409, 573, 599, 609, 615, 734, 778, 842, 864, 879, 1670, 1699, 2046, 2076, 2119, 2134, 2149, 2264, 2268, 2272, 2278, 2285, 2357, 2540, 2662, 2672, 2694, 2702, 2738, 2747, 2903, 2966, 2991, 3029, 3033, 3126, 3130, 3163, 3222,

3261, 3334, 3375, 3482, 3501, 3630, 3634, 3651, 3655, 3677, 3681, 3691, 3703, 3748, 3794, 3828, 3870, 3915, 4134, 4403, 4410, 4417, 4522, 4541, 4571, 4712, 4761, 4976, 5049, 5056, 5070, 5078, 5083, 5093, 5253, 5293, 5300, 5315, 5324, 5338, 5380, 5485, 5498, 5560, 5693, 5705, 5729, 5741, 5779, 5789, 5797, 5819	
\cs_new_protected_nopar:Nn	4283, 4327, 4335, 4343, 5028, 5036, 5167, 5272, 5280, 5446
\cs_new_protected_nopar:Npn	4275, 4291, 5099, 5145, 5399, 5420
\cs_set:Npn	1978, 2168, 2215, 2322, 2838, 2875, 5491
\cs_set_eq:NN	3758, 4949, 4950, 5147, 5218, 5219, 5390, 5422
\cs_set_protected:Nn	1061, 1077, 1090, 1102
\cs_set_protected:Npn	32, 39, 48, 58, 66, 81, 87, 130, 162, 173, 590, 631, 641, 663, 698, 714, 760, 895, 921, 937, 1017, 1040, 1114, 1123, 1202, 1219, 1726, 1837, 1853, 2292, 2338, 2398, 2557, 2592, 2680, 2830, 3285, 3603, 3619, 3665, 3826, 3868
\cs_to_str:N	601, 624

D

\d	220
\DeclareDocumentEnvironment	558
dim commands:	
\dim_abs:n	3799, 3804
\dim_add:Nn	3434, 4437, 4675, 4706
\dim_compare:nNnTF	1063, 1079, 1092, 1104, 1382, 1394, 1422, 1434, 1461, 1473, 1550, 1558, 1672, 1701, 2971, 2979, 3429, 3796, 3801, 3807, 3813, 3815, 3817, 3981, 4028, 4142, 4159, 4412, 4652, 4668, 4683, 4699, 4812, 4877, 5351
\dim_compare:nTF	2939, 4073, 4218
\dim_eval:n	997, 4495, 4582
\dim_gset_eq:NN	4821, 4886
\dim_gzero:N	4872, 4932
\dim_new:N	54, 61, 62, 63, 83, 126, 127, 139, 146, 147, 181, 183, 184, 190
\dim_set:Nn	612, 1031, 2973, 2981, 3416, 3420, 3425, 3431, 3518, 3799, 3804, 3806, 3809, 3810, 3814, 3816, 3819, 3820, 3822, 3984, 4031, 4072, 4144, 4161, 4217, 4426, 4531, 4618, 4654, 4661, 4685, 4692, 4747, 4796, 4814, 4879, 5095, 5353
\dim_set_eq:NN	721, 767, 834, 3513, 3837, 3881, 3992, 4169, 4754, 4757, 4758, 4803, 4806, 4807, 5088, 5159, 5434
\dim_sub:Nn	4078, 4223, 4670, 4701
\dim_use:N	1064, 1072, 1673, 1683, 2817, 2820, 2825, 2983, 3533, 3535, 3588, 3982, 3986, 3987, 3989, 4029, 4034, 4035, 4041, 4075, 4080
\dim_zero:N	3873, 3995, 4170, 4440
\dim_zero_new:N	572
\c_zero_dim	1066, 1080, 1093, 1105, 1673, 1701, 2941, 2971, 2979, 3416, 3429, 3796, 3801, 3807, 3814, 3982, 4029, 4075, 4142, 4159, 4220, 4412, 4652, 4668, 4683, 4699, 4812, 4877, 5351
\dimeval	2626

E

\end	2780, 2811, 4013, 4185, 4477, 4632, 5568, 5578, 5586
end internal commands:	
\end__enumext_mini_page	1681, 1708, 4056, 4205, 4836, 4900, 4926
\endlist	367
\endminipage	373
endpenalty	921


```

enumext ..... 5, 4091
enumext internal commands:
  \__enumext_add_meta_key:nnn .. 143, 5707, 5723,
    5724, 5726, 5729
  \__enumext_add_pre_parsep: . 56, 1243, 1245, 1245
  \__enumext_after_args_exec: 54, 1138, 1150, 4105
  \__enumext_after_args_exec_v: 1154, 1166, 4243
  \__enumext_after_args_exec_vii: .. 1170, 1194
  \__enumext_after_args_exec_viii: ..... 1198
  \__enumext_after_env:nn 94, 112, 125, 133, 206, 206,
    546, 550, 4110, 4845, 4908, 5200
  \__enumext_after_hyperref: ... 39, 375, 375, 377
  \l__enumext_after_list_args_v_tl ..... 1168
  \l__enumext_after_list_args_vii_tl 1196, 5165
  \l__enumext_after_list_args_viii_tl .. 1200,
    5444
  \__enumext_after_list_vii: 125, 129, 4957, 4996,
    4996
  \__enumext_after_list_viii: .. 134, 5227, 5266,
    5266
  \__enumext_after_stop_list: 54, 111, 1138, 1146,
    4061
  \__enumext_after_stop_list_v: 1154, 1162, 4212
  \l__enumext_after_stop_list_v_tl ..... 1164
  \__enumext_after_stop_list_vii: .. 129, 1170,
    1186, 4999
  \l__enumext_after_stop_list_vii_tl ... 1188
  \__enumext_after_stop_list_viii: . 1190, 5269
  \l__enumext_after_stop_list_viii_tl ... 1192
  \l__enumext_align_label_pos_v_str 3412, 3784
  \l__enumext_align_label_pos_X_str ..... 66
  \l__enumext_align_label_vii_str ..... 5134
  \l__enumext_align_label_viii_str ..... 5413
  \l__enumext_align_label_X_str ..... 173
  \c__enumext_all_envs_clist .. 194, 662, 920, 936,
    1122, 1137, 1218, 1742
  \c__enumext_all_families_seq .. 142, 5661, 5687
  \__enumext_anskey_env_file_if_writable:n 92,
    3140, 3140
  \__enumext_anskey_env_file_if_-
    writable:nTF ..... 3140, 3165
  \__enumext_anskey_env_file_write:nn 92, 3163,
    3172, 3227
  \l__enumext_anskey_env_force_eol_bool .. 94,
    3113, 3229
  \c__enumext_anskey_env_hidden_space_str 33,
    94, 109, 3233
  \l__enumext_anskey_env_overwrite_bool 3121,
    3146
  \__enumext_anskey_env_safe_inner: . 93, 3187,
    3192, 3211
  \__enumext_anskey_env_safe_inner:n .... 93
  \__enumext_anskey_env_safe_outer: . 93, 3175,
    3192, 3192
  \__enumext_anskey_env_unknown:n 92, 3124, 3126,
    3126
  \__enumext_anskey_env_unknown:nn . 3126, 3128,
    3130
  \l__enumext_anskey_level_int .. 16, 3085, 3086
  \__enumext_anskey_safe_inner: . 91, 3058, 3064,
    3083
  \__enumext_anskey_safe_inner:n ..... 90
  \__enumext_anskey_safe_outer: . 90, 3045, 3064,
    3064
  \__enumext_anskey_show_wrap_arg:n . 89, 2966,
    2966, 2995, 3010
  \__enumext_anskey_show_wrap_left:n 89, 2911,
    2991, 2991
  \__enumext_anskey_unknown:n 90, 3013, 3027, 3029
  \__enumext_anskey_unknown:nn . 3013, 3031, 3033
  \__enumext_anskey_wrapper:n ..... 2623, 2989
  \l__enumext_anspic_above_int . 138, 4619, 4620,
    4622
  \__enumext_anspic_args:nnn 120, 121, 4503, 4519,
    4597
  \l__enumext_anspic_args_seq 120-122, 138, 4517,
    4631, 4644
  \l__enumext_anspic_below_int . 138, 4619, 4620,
    4623
  \l__enumext_anspic_body_box ... 138, 4530, 4533
  \__enumext_anspic_body_dim:n .. 120, 4503, 4522,
    4574
  \l__enumext_anspic_body_htdp_dim .. 120, 138,
    4531, 4585
  __enumext_anspic_exec: ..... 4503
  \__enumext_anspic_exec: ... 119, 122, 4472, 4640
  \__enumext_anspic_label:nn 120, 4503, 4541, 4577,
    4592
  \l__enumext_anspic_label_above_bool ... 138,
    4355, 4358, 4422, 4487, 4524, 4575, 4602
  \l__enumext_anspic_label_box .. 138, 4425, 4428
  \l__enumext_anspic_label_htdp_dim . 118, 138,
    4426, 4432, 4497, 4584
  \__enumext_anspic_label_pos:nnn .. 121, 4503,
    4571, 4600
  \l__enumext_anspic_label_sep_skip 4365, 4434,
    4498, 4587, 4604
  \l__enumext_anspic_layout_style_tl 4367, 4642,
    4647
  \l__enumext_anspic_mini_pos_str .. 138, 4356,
    4359, 4629
  \l__enumext_anspic_mini_width_dim 138, 4543,
    4618, 4629
  \__enumext_anspic_print:n 121, 4503, 4611, 4615,
    4644, 4647
  \__enumext_anspic_row:n 121, 122, 4503, 4613, 4616
  \__enumext_anspic_start_list_tag: 4299, 4327,
    4599
  \__enumext_anspic_stop_list_tag: . 4299, 4343,
    4609
  \__enumext_anspic_stop_start_list_tag: 4299,
    4335, 4601
  \__enumext_at_begin_document:n .. 39, 202, 202,
    364, 370
  \l__enumext_base_line_fix_bool 51, 140, 983, 992,
    1015, 5574, 5579
  \__enumext_before_args_exec: 54, 110, 128, 1138,
    1138, 4026
  \__enumext_before_args_exec_v: 1154, 1154, 4141
  \__enumext_before_args_exec_vii: . 1170, 1170,
    4993
  \__enumext_before_args_exec_viii: 1174, 5263
  \__enumext_before_env:nn ..... 206, 210
  \__enumext_before_keys_exec: .. 54, 1138, 1142,
    4101
  \__enumext_before_keys_exec_v: 1154, 1158, 4239
  \__enumext_before_keys_exec_vii ..... 1170
  \__enumext_before_keys_exec_vii: . 1178, 4943

```

```

\__enumext_before_keys_exec_viii: 1182, 5212
\__enumext_before_list: .. 110, 4023, 4023, 4095
\__enumext_before_list_v: ... 4138, 4138, 4234
\__enumext_before_list_vii: ... 128, 4938, 4989,
4989
\__enumext_before_list_viii: . 134, 5208, 5260,
5260
\l__enumext_before_no_starred_key_v_tl 1160
\l__enumext_before_no_starred_key_vii-
tl ..... 1180
\l__enumext_before_no_starred_key_viii-
tl ..... 1184
\l__enumext_before_starred_key_v_tl ... 1156
\l__enumext_before_starred_key_vii_tl . 1172
\l__enumext_before_starred_key_viii_tl 1176
\__enumext_calc_hspace:NNNNNN 105, 3794, 3794,
3825, 3830, 3874
\__enumext_check_ans_active: 78, 110, 128, 2417,
2417, 4027, 4992
\g__enumext_check_ans_item_tl ..... 96
\g__enumext_check_ans_key_bool 79, 80, 148, 339,
2476, 2482, 3252
\l__enumext_check_ans_key_bool 79, 2402, 2407,
2473, 2479
\__enumext_check_ans_key_hook: .. 79, 111, 129,
2470, 2470, 4062, 5000
\__enumext_check_ans_level: . 78, 79, 2417, 2423,
2427
\__enumext_check_ans_log: 80, 94, 2516, 2516, 3256
\__enumext_check_ans_log_msg_greater: 2516,
2522, 2535
\__enumext_check_ans_log_msg_less: 2516, 2520,
2525
\__enumext_check_ans_log_msg_same_ok: 2516,
2521, 2530
\__enumext_check_ans_msg_greater: 2492, 2498,
2511
\__enumext_check_ans_msg_less: 2492, 2496, 2501
\__enumext_check_ans_msg_same_ok: 2492, 2497,
2506
\__enumext_check_ans_show: .. 80, 94, 2492, 2492,
3254
\l__enumext_check_answers_bool 77, 78, 90, 93, 99,
148, 2380, 2406, 2421, 2754, 2778, 2785, 2809, 3047,
3176, 3370, 3486, 3520, 5114
\__enumext_check_starred_cmd:n 37, 81, 96, 133,
2540, 2540, 4246, 4485, 5226
\g__enumext_check_starred_cmd_int . 103, 148,
2543, 2549, 2554, 3712, 4553, 5350
\l__enumext_check_start_line_env_tl . 37, 148,
302, 310, 318, 2546, 2552, 2555
\l__enumext_columns_sep_v_dim 4159, 4161, 4169
\l__enumext_columns_sep_vii_dim .. 4652, 4654,
4663, 4675, 4751, 5181
\l__enumext_columns_sep_viii_dim . 4683, 4685,
4694, 4706, 4800, 5460
\l__enumext_columns_v_int 1515, 1533, 1704, 4157,
4165, 4177, 4182
\l__enumext_columns_vii_int .. 4657, 4660, 4664,
4673, 4715, 4719, 4722, 4728, 4734, 4738, 5175, 5189
\l__enumext_columns_viii_int . 4688, 4691, 4695,
4704, 4764, 4768, 4771, 4777, 4783, 4787, 5454, 5469
\l__enumext_counter_i_tl ..... 32, 582
\l__enumext_counter_ii_tl ..... 32, 583
\l__enumext_counter_iii_tl ..... 32, 584
\l__enumext_counter_iv_tl ..... 32, 585
\g__enumext_counter_styles_tl . 31, 44, 54, 602,
620
\l__enumext_counter_v_tl ..... 32, 586
\l__enumext_counter_vi_tl ..... 32, 587
\l__enumext_counter_vii_tl ..... 32, 588
\l__enumext_counter_viii_tl ..... 32, 589
\l__enumext_current_widest_dim 31, 54, 626, 722,
768, 835
\__enumext_def_meta_key:nnn .. 143, 5707, 5735,
5741, 5755
\__enumext_default_item:n ... 3482, 3482, 3546
\__enumext_define_counter:Nn . 31, 573, 573, 582,
583, 584, 585, 586, 587, 588, 589
\__enumext_endminipage: . 39, 364, 373, 567, 4866,
5169, 5448
\g__enumext_envir_name_tl 37, 22, 274, 288, 347,
2350, 2355, 2365, 2504, 2509, 2514, 2528, 2533, 2538
\l__enumext_envir_name_tl . 36, 37, 102, 22, 244,
254, 301, 309, 317, 3624, 3647, 3671, 4390, 6119, 6122,
6129, 6132, 6139, 6142, 6149, 6152, 6158, 6162, 6168,
6172, 6229, 6233
\__enumext_execute_after_env: .. 38, 76, 80, 94,
3242, 3242, 4112, 5202
\__enumext_fake_item_indent: . 1061, 1061, 3858
\l__enumext_fake_item_indent_v_dim 1080, 1085
\l__enumext_fake_item_indent_v_tl 1082, 3696,
3700, 3707
\__enumext_fake_item_indent_vii: . 1061, 1090,
3892
\l__enumext_fake_item_indent_vii_dim . 1093,
1097
\l__enumext_fake_item_indent_vii_tl .. 1095,
5164
\__enumext_fake_item_indent_viii: 1061, 1102,
3897
\l__enumext_fake_item_indent_viii_dim 1105,
1109
\l__enumext_fake_item_indent_viii_tl . 1107,
5439
\l__enumext_fake_item_indent_X_tl ..... 87
\__enumext_fake_make_label_vii:n . 131, 5099,
5099, 5156
\__enumext_fake_make_label_viii:n 5380, 5399,
5431
\__enumext_filter_level:n 141, 5594, 5594, 5624,
5630, 5636, 5642, 5648
\__enumext_filter_level_key:n 141, 5594, 5599,
5603
\__enumext_filter_level_pair:nn .. 141, 5594,
5600, 5611
\__enumext_filter_save_key:n .. 84, 2669, 2677,
2700, 2706, 2708, 2708, 5511, 5517, 5523, 5529, 5535,
5541
\__enumext_filter_save_key_key:n .. 84, 2708,
2713, 2717
\__enumext_filter_save_key_pair:nn 84, 2708,
2714, 2725
\__enumext_filter_series:n 70, 2020, 2020, 2055,
2063, 2098, 2111
\__enumext_filter_series_key:n 70, 2020, 2025,
2029
\__enumext_filter_series_pair:nn .. 71, 2020,
2026, 2037

```

__enumext_first_item_tmp_vii: 127, 129, 4949, 5028, 5028
 __enumext_first_item_tmp_viii: .. 135, 5218, 5272, 5272
 \g__enumext_footnote_standar_arg_seq .. 167, 441, 452, 455
 \g__enumext_footnote_standar_int 167, 435, 438, 440, 443
 \g__enumext_footnote_standar_int_seq .. 167, 443, 448, 451, 456
 \g__enumext_footnote_starred_arg_seq .. 167, 471, 482, 485
 \g__enumext_footnote_starred_int 167, 465, 468, 470, 473
 \g__enumext_footnote_starred_int_seq .. 167, 473, 478, 481, 486
 __enumext_footnotes_key_bool 39
 \l__enumext_footnotes_key_bool 34, 39, 157, 385, 389, 396, 497, 513, 527, 540
 __enumext_footnotetext:nn .. 424, 424, 453, 483
 __enumext_foreach_add_body:n 144, 5756, 5816, 5819
 \l__enumext_foreach_after_tl 5760, 5828
 \l__enumext_foreach_before_tl 5758, 5823
 \g__enumext_foreach_default_keys_tl ... 143
 \l__enumext_foreach_default_keys_tl ... 116, 5778, 5799
 __enumext_foreach_keyans:nn . 144, 5756, 5795, 5797
 \l__enumext_foreach_name_prop_tl . 116, 5801, 5826
 \l__enumext_foreach_print_seq 116, 5811, 5817, 5821
 \l__enumext_foreach_sep_tl 5770, 5817
 \l__enumext_foreach_start_int 5762, 5813
 \l__enumext_foreach_step_int 5766, 5814
 \l__enumext_foreach_stop_int . 5764, 5806, 5808, 5815
 __enumext_foreach_wrapper:n 5768, 5824
 __enumext_getkeyans:nn .. 139, 5480, 5494, 5498
 __enumext_getkeyans_aux:n 138, 5480, 5482, 5485
 \l__enumext_hyperref_bool 34, 39, 157, 382, 399, 416, 2956, 3358, 5108
 __enumext_hypertarget:nn 39, 375, 401, 405, 421
 __enumext_if_is_int:n 218
 __enumext_if_is_int:nTF 218, 867, 881
 __enumext_internal_mini_page: 42, 108, 128, 554, 554, 3907, 4962
 __enumext_is_not_nested: . 31, 36, 108, 128, 238, 238, 3906, 4961
 __enumext_is_on_first_level: . 31, 37, 108, 128, 238, 264, 3913, 4974
 \g__enumext_item_anskey_int 90, 96, 148, 334, 361, 362, 2489, 2905, 3372
 __enumext_item_answer_diff: 80, 94, 2485, 2485, 3249
 \g__enumext_item_answer_diff_int 80, 148, 335, 2487, 2494, 2518
 \l__enumext_item_column_pos_vii_int 129, 4722, 4728, 4734, 4738, 4745, 5039, 5175, 5178
 \l__enumext_item_column_pos_viii_int .. 135, 4771, 4777, 4783, 4787, 4794, 5283, 5454, 5457
 \l__enumext_item_column_pos_X_int 173
 \g__enumext_item_count_all_vii_int 129, 4746, 5040, 5189, 5197
 \g__enumext_item_count_all_viii_int 135, 4795, 5284, 5468, 5477
 \g__enumext_item_count_all_X_int 173
 \g__enumext_item_number_bool 148
 \l__enumext_item_number_bool 79, 155, 2439, 2444, 2448, 2452, 2465, 3090, 3213, 3489, 3523, 5117
 \g__enumext_item_number_int .. 79, 148, 333, 360, 362, 2438, 2443, 2447, 2451, 2464, 2489, 3488, 3522, 5116
 __enumext_item_peek_args_vii: 129, 130, 5036, 5041, 5043
 __enumext_item_peek_args_viii: .. 135, 5280, 5285, 5287
 __enumext_item_starred_exec: . 100, 3501, 3528, 3570, 3591
 __enumext_item_starred_exec:nn .. 3501, 3501, 3544
 \l__enumext_item_starred_vii_bool 5058, 5072, 5121
 \l__enumext_item_starred_viii_bool 5302, 5317, 5409, 5440
 \l__enumext_item_starred_X_bool 173
 __enumext_item_std:w . 39, 99, 103, 364, 368, 3492, 3498, 3526, 3696, 3700, 3707
 \g__enumext_item_symbol_aux_tl . 99, 120, 3506, 3509, 3534, 3578, 3598
 \g__enumext_item_symbol_aux_vii_tl 5080, 5123, 5126, 5130, 5132
 \g__enumext_item_symbol_aux_X_tl 173
 \l__enumext_item_symbol_sep_vii_dim .. 5088, 5095, 5129, 5131
 \l__enumext_item_symbol_vii_tl 5126
 \l__enumext_item_text_vii_box 5148, 5173
 \l__enumext_item_text_viii_box ... 5423, 5452
 \l__enumext_item_text_X_box 173
 \l__enumext_item_width_vii_dim ... 4661, 4670, 4749, 4757, 4758
 \l__enumext_item_width_viii_dim .. 4692, 4701, 4798, 4806, 4807
 \l__enumext_item_width_X_dim 173
 \l__enumext_item_wrap_key_bool 104, 148, 3442, 3462, 3720, 3727, 3754, 4547, 4565, 5303, 5318, 5386
 \l__enumext_itemindent_X_dim 58
 \l__enumext_itemsep_i_skip ... 1376, 1383, 1386, 1388, 1395, 1399, 1402, 1404, 1544, 1551, 1553, 1554, 1559, 1563, 1565, 1566
 \l__enumext_itemsep_ii_skip .. 1416, 1423, 1426, 1428, 1435, 1439, 1442, 1444
 \l__enumext_itemsep_iii_skip . 1455, 1462, 1465, 1467, 1474, 1478, 1481, 1483
 \l__enumext_itemsep_vii_skip 5195
 \l__enumext_itemsep_viii_skip 5475
 \l__enumext_joined_item_aux_vii_int .. 4743, 4744, 4745, 4746, 4752
 \l__enumext_joined_item_aux_viii_int . 4792, 4793, 4794, 4795, 4801
 \l__enumext_joined_item_aux_X_int 173
 __enumext_joined_item_vii:w .. 130, 5036, 5046, 5047, 5049
 \l__enumext_joined_item_vii_int .. 4714, 4715, 4718, 4720, 4726, 4731, 4736, 4741, 4743, 4749
 __enumext_joined_item_viii:w . 135, 5280, 5290, 5291, 5293
 \l__enumext_joined_item_viii_int . 4763, 4764, 4767, 4769, 4775, 4780, 4785, 4790, 4792, 4798

```

\l__enumext_joined_item_X_int . . . . . 173
\l__enumext_joined_width_vii_dim . 4747, 4754,
    4757, 5150, 5158
\l__enumext_joined_width_viii_dim 4796, 4803,
    4806, 5425, 5433
\l__enumext_joined_width_X_dim . . . . . 173
\__enumext_keyans_addto_prop:n 95, 3261, 3261,
    3709, 4550
\__enumext_keyans_addto_seq:n . 96, 3334, 3334,
    3711, 4552
\__enumext_keyans_addto_seq_link: 3334, 3352,
    3354, 5349
\__enumext_keyans_default_item:n . 103, 3691,
    3691, 3728
\l__enumext_keyans_env_bool 22, 3945, 3958, 4121,
    4211
\__enumext_keyans_fake_item_indent: . . 1061,
    1077, 3848
\l__enumext_keyans_level_h_int . . 133, 16, 795,
    819, 3074, 3202, 3312, 4968, 5235, 5236
\l__enumext_keyans_level_int . . 16, 1664, 3070,
    3198, 3307, 3452, 4120, 4125, 4513
\__enumext_keyans_make_label: 104, 3732, 3732,
    3846
\__enumext_keyans_make_label_box: 3732, 3736,
    3741, 3778
\__enumext_keyans_make_label_std: 3732, 3744,
    3766
\__enumext_keyans_mini_right_cmd:n 64, 1666,
    1699, 1699
\__enumext_keyans_mini_set_vskip: . . . . . 61
\__enumext_keyans_minipage_add_space: 1495,
    1521, 4150
\__enumext_keyans_minipage_set_skip: . 1495,
    1495, 1523
\__enumext_keyans_multi_addvspace: 1295, 1306,
    4174
\__enumext_keyans_multi_set_vskip: 57, 1295,
    1295, 1308
\__enumext_keyans_multicols_start: 4138, 4153,
    4155
\__enumext_keyans_multicols_stop: 1703, 4138,
    4180, 4209
\__enumext_keyans_name_and_start: 31, 37, 133,
    295, 295, 4122, 4401, 5240
\__enumext_keyans_parse_keys:n 4134, 4134, 4233
\__enumext_keyans_pic_arg_two: 118, 4394, 4417,
    4448
\l__enumext_keyans_pic_level_int . . 16, 1645,
    3078, 3206, 3264, 3302, 3337, 4396, 4397
\__enumext_keyans_pic_parse_keys:n 4394, 4403,
    4447
\__enumext_keyans_pic_safe_exec: . 117, 4394,
    4394, 4446
\__enumext_keyans_pic_skip_abs:N . 118, 4394,
    4410, 4421
\__enumext_keyans_pos_mark_set: 97, 3408, 3408,
    3445, 3477
\__enumext_keyans_pre_itemsep_skip: . . 1495,
    1514, 1541
\__enumext_keyans_redefine_item: . 104, 3714,
    3714, 3845
\__enumext_keyans_ref: . . . . . 48, 842, 857, 3847
\__enumext_keyans_ref:n . . . . . 48, 839, 842, 842
\__enumext_keyans_safe_exec: . 4114, 4114, 4232
\__enumext_keyans_save_item_opt:n . . 97, 103,
    3375, 3375, 3705, 4549
\__enumext_keyans_set_item_width: 114, 4215,
    4215, 4242
\__enumext_keyans_show_ans: 98, 3408, 3437, 3771,
    3786, 4554
\__enumext_keyans_show_item_opt: 97, 103, 3375,
    3382, 3708, 4562
\__enumext_keyans_show_item_opt_viii: . . 97,
    3375, 3395, 5442
\__enumext_keyans_show_pos: 98, 3408, 3450, 3772,
    3787, 4555
\__enumext_keyans_starred_item:n . 103, 3703,
    3703, 3723
\__enumext_keyans_starred_item_star: . . 136,
    5315, 5343, 5411
\__enumext_keyans_start_counter: . 4226, 4226,
    4241
\__enumext_keyans_store_ref: . . 95, 3281, 3281,
    3710, 4551, 5347
\__enumext_keyans_store_ref_aux_i: 95, 3281,
    3293, 3296
\__enumext_keyans_store_ref_aux_ii: 96, 3281,
    3322, 3324
\__enumext_keyans_unknown_keys:n . 3619, 3625,
    3630, 4391
\__enumext_keyans_unknown_keys:nn 3619, 3632,
    3634
\__enumext_keyans_wraper_label:n . . . . . 104
\__enumext_keyans_wraper_label_viii:n 5380,
    5380, 5416
\__enumext_keyans_wrapper_item_v:n 3755, 3758
\__enumext_keyans_wrapper_item_viii:n 5387,
    5391
\__enumext_keyans_wrapper_label:n 3732, 3748,
    3774, 3789, 4559
\__enumext_keyans_wrapper_opt_v:n . . . . 3390
\__enumext_keyans_wrapper_opt_viii:n . . 3403
\l__enumext_label_copy_i_tl . . 2871, 3300, 3305,
    3310, 3315
\l__enumext_label_copy_v_tl . . . . . 3310
\l__enumext_label_copy_vi_tl . . . . . 3305
\l__enumext_label_copy_vii_tl 2847, 2858, 2887,
    3300
\l__enumext_label_copy_viii_tl . . . . . 3315
\l__enumext_label_copy_X_tl . . . . . 159
\l__enumext_label_fill_left_v_tl . . . . . 3770
\l__enumext_label_fill_left_X_tl . . . . . 87
\l__enumext_label_fill_right_v_tl . . . . 3775
\l__enumext_label_fill_right_X_tl . . . . . 87
\l__enumext_label_font_style_v_tl 3773, 3788,
    4558, 4566
\l__enumext_label_font_style_vii_tl . . 5136
\l__enumext_label_font_style_viii_tl . . 5415
\l__enumext_label_i_tl . . . . . 714
\l__enumext_label_ii_tl . . . . . 714
\l__enumext_label_iii_tl . . . . . 714
\l__enumext_label_iv_tl . . . . . 714
\__enumext_label_style:Nnn 31, 44, 615, 615, 630,
    719, 765, 830, 832
\l__enumext_label_v_tl 96, 827, 3269, 3342, 3411,
    4236, 4425
\l__enumext_label_vi_tl 96, 827, 3266, 3339, 4559,
    4567

```

```

\l__enumext_label_vii_tl . 760, 5067, 5090, 5097
\l__enumext_label_viii_tl 760, 5312, 5341, 5345
\l__enumext_label_width_by_box . . 54, 611, 612
\__enumext_label_width_by_box:Nn 44, 609, 609,
614, 626, 891, 3410
\l__enumext_labelsep_v_dim . . . 3431, 4164, 4437,
4561
\l__enumext_labelsep_vii_dim . 2973, 4656, 4666,
4750, 5032, 5088, 5143, 5152
\l__enumext_labelsep_viii_dim 4687, 4697, 4799,
5276, 5353, 5418, 5427
\l__enumext_labelwidth_v_dim . 835, 3421, 3426,
3447, 3479, 3784, 4164, 4437, 4556
\l__enumext_labelwidth_vii_dim . . . 2976, 4656,
4665, 4750, 5032, 5134, 5151
\l__enumext_labelwidth_viii_dim . . 4687, 4696,
4799, 5276, 5360, 5377, 5413, 5426
\l__enumext_leftmargin_tmp_v_bool . 118, 4419
\l__enumext_leftmargin_tmp_X_bool . . . . . 58
\l__enumext_leftmargin_tmp_X_dim . . . . . 58
\l__enumext_leftmargin_X_dim . . . . . 58
\__enumext_level: 214, 214, 744, 746, 755, 757, 1064,
1068, 1072, 1140, 1144, 1148, 1152, 1235, 1237, 1239,
1241, 1283, 1285, 1287, 1289, 1293, 1327, 1333, 1338,
1340, 1343, 1346, 1359, 1362, 1673, 1677, 1683, 1746,
1748, 1750, 1753, 1760, 1762, 1764, 1767, 1912, 1913,
1922, 1928, 1931, 1932, 2050, 2052, 2054, 2095, 2097,
2100, 2103, 2123, 2127, 2153, 2664, 2666, 2668, 2696,
2697, 2699, 2756, 2764, 2768, 2772, 2983, 2987, 3491,
3492, 3496, 3497, 3498, 3506, 3514, 3515, 3518, 3525,
3526, 3530, 3533, 3535, 3569, 3571, 3572, 3574, 3577,
3588, 3589, 3592, 3593, 3595, 3951, 3964, 3971, 3979,
3982, 3984, 3986, 3987, 3988, 3989, 3992, 3997, 4003,
4009, 4016, 4029, 4031, 4034, 4035, 4037, 4041, 4047,
4075, 4080, 4086, 4088, 4098, 4100
\l__enumext_level_h_int 128, 16, 247, 270, 283, 781,
812, 1652, 2001, 2058, 2085, 2106, 2136, 2158, 2194,
2229, 2435, 2455, 2866, 3959, 4963, 4964
\l__enumext_level_int . 108, 16, 216, 257, 269, 284,
556, 1247, 1372, 1651, 1894, 1985, 1995, 1998, 2048,
2080, 2092, 2121, 2126, 2151, 2178, 2188, 2191, 2221,
2225, 2227, 2312, 2314, 2316, 2329, 2331, 2429, 2461,
2843, 2853, 2859, 2865, 2872, 2881, 2886, 3244, 3670,
3862, 3908, 3909, 3920, 3931, 3949, 3962, 3993, 4129,
4509, 5012, 5022, 5248, 6159, 6163, 6169, 6173, 6250
\__enumext_list_arg_two_i: . . . . . 3826
\__enumext_list_arg_two_ii: . . . . . 3826
\__enumext_list_arg_two_iii: . . . . . 3826
\__enumext_list_arg_two_iv: . . . . . 3826
\__enumext_list_arg_two_v: 104, 3826, 4238, 4420
\__enumext_list_arg_two_vii: . . . . 3868, 4942
\__enumext_list_arg_two_viii: . . . . 3868, 5211
\l__enumext_listoffset_v_dim . 4166, 4220, 4223
\l__enumext_listparindent_vii_dim 5159, 5163
\l__enumext_listparindent_viii_dim 5434, 5438
\__enumext_log_answer_vars: . 38, 349, 357, 3251
\__enumext_log_global_vars: . 38, 349, 349, 3250
\__enumext_make_label: . . . 100, 3549, 3549, 3856
\__enumext_make_label_box: . . . 3549, 3553, 3558,
3581
\__enumext_make_label_std: . . . 3549, 3561, 3565
\l__enumext_mark_answer_sym_tl 86, 2608, 2822,
2999, 3433, 5357, 5364
\l__enumext_mark_answer_sym_v_tl . 3433, 3465
\l__enumext_mark_answer_sym_viii_tl . . . 5357
\l__enumext_mark_position_str 120, 2614, 2615,
2616, 2820, 3435, 5358, 5375
\l__enumext_mark_position_v_str . . 120, 3435
\l__enumext_mark_position_viii_str 120, 5358,
5375
\l__enumext_mark_ref_sym_tl . . 2596, 2961, 3366
\l__enumext_mark_sep_tmpa_dim 120, 3411, 3421,
3426
\l__enumext_mark_sep_tmpb_dim 120, 3416, 3420,
3425, 3434
\l__enumext_mark_sym_sep_dim . 2611, 2971, 2973,
2976, 2979, 2981
\l__enumext_mark_sym_sep_v_dim . . . 3429, 3431,
3434, 3447, 3479
\l__enumext_mark_sym_sep_viii_dim 5351, 5353,
5360, 5377
\l__enumext_meta_path_tl . 116, 5731, 5732, 5734,
5735
\c__enumext_meta_paths_prop . . . . . 143, 5707
\__enumext_mini_addvspace_vii: 63, 1631, 1631,
4824
\__enumext_mini_addvspace_viii: 63, 1631, 1637,
4889
__enumext_mini_env* . . . . . 554
\__enumext_mini_page 1683, 1710, 4041, 4151, 4826,
4891, 4912
\__enumext_mini_right_cmd:n 64, 1668, 1670, 1670
\__enumext_mini_set_vskip_vii: 62, 1574, 1574,
1633
\__enumext_mini_set_vskip_viii: 62, 1574, 1596,
1639
\__enumext_minipage:w 39, 364, 372, 561, 4849, 5158,
5433
\l__enumext_minipage_active_v_bool 4148, 4171,
4196
\g__enumext_minipage_active_vii_bool . . 125,
4838, 4847, 4869
\l__enumext_minipage_active_vii_bool . 4820,
4831
\g__enumext_minipage_active_viii_bool 4902,
4910, 4929
\l__enumext_minipage_active_viii_bool 4885,
4896
\g__enumext_minipage_active_X_bool . . . 173
\l__enumext_minipage_active_X_bool . . . . 74
\__enumext_minipage_add_space: . 59, 110, 1323,
1349, 4039
\g__enumext_minipage_after_skip 74, 1578, 1590,
4867, 4927
\l__enumext_minipage_after_skip . . 58, 111, 74,
1336, 1376, 1378, 1383, 1386, 1390, 1395, 1399, 1402,
1406, 1418, 1423, 1426, 1430, 1435, 1439, 1442, 1446,
1457, 1462, 1465, 1469, 1474, 1478, 1481, 1485, 1497,
1511, 1544, 1546, 1551, 1553, 1555, 1559, 1563, 1565,
1567, 1598, 1611, 1625, 1679, 1706, 4206
\g__enumext_minipage_center_vii_bool . 4853,
4870
\g__enumext_minipage_center_viii_bool 4914,
4930
\g__enumext_minipage_center_X_bool . . . 173
\l__enumext_minipage_hsep_v_dim . . . . . 4146
\l__enumext_minipage_hsep_vii_dim . . . . 4818
\l__enumext_minipage_hsep_viii_dim . . . 4883

```


`\l__enumext_minipage_left_skip` [74](#), [1498](#), [1576](#), [1581](#), [1585](#), [1599](#), [1603](#), [1617](#), [1635](#), [1641](#)
`\l__enumext_minipage_left_v_dim` .. [4144](#), [4151](#)
`\l__enumext_minipage_left_vii_dim` [4814](#), [4826](#)
`\l__enumext_minipage_left_viii_dim` [4879](#), [4891](#)
`\l__enumext_minipage_left_X_dim` [74](#)
`\g__enumext_minipage_right_skip` [74](#), [1577](#), [1582](#), [1586](#), [4852](#), [4913](#)
`\l__enumext_minipage_right_skip` . [58](#), [74](#), [1325](#), [1331](#), [1336](#), [1338](#), [1340](#), [1499](#), [1500](#), [1506](#), [1511](#), [1512](#), [1513](#), [1518](#), [1600](#), [1607](#), [1621](#), [1685](#), [1712](#)
`\l__enumext_minipage_right_v_dim` . [1701](#), [1710](#), [4142](#), [4146](#)
`\g__enumext_minipage_right_vii_dim` [125](#), [4822](#), [4849](#), [4872](#)
`\l__enumext_minipage_right_vii_dim` [125](#), [4812](#), [4817](#), [4823](#)
`\g__enumext_minipage_right_viii_dim` .. [4887](#), [4912](#), [4932](#)
`\l__enumext_minipage_right_viii_dim` .. [4877](#), [4882](#), [4888](#)
`\g__enumext_minipage_right_X_dim` [173](#)
`\g__enumext_minipage_right_X_skip` [173](#)
`__enumext_minipage_set_skip`: . [58](#), [1323](#), [1323](#), [1351](#)
`\g__enumext_minipage_stat_int` .. [110](#), [74](#), [1690](#), [1717](#), [4038](#), [4049](#), [4054](#), [4149](#), [4198](#), [4203](#)
`\l__enumext_minipage_temp_skip` [74](#), [1397](#), [1407](#), [1410](#), [1437](#), [1447](#), [1450](#), [1476](#), [1486](#), [1489](#), [1561](#), [1568](#), [1570](#)
`\l__enumext_miniright_code_vii_box` [4860](#), [4864](#)
`\g__enumext_miniright_code_vii_tl` [126](#), [4855](#), [4862](#), [4871](#)
`\l__enumext_miniright_code_viii_box` .. [4921](#), [4925](#)
`\g__enumext_miniright_code_viii_tl` [4916](#), [4923](#), [4931](#)
`\l__enumext_miniright_code_X_box` [173](#)
`\l__enumext_mode_box_bool` [635](#), [3556](#), [3739](#)
`__enumext_multi_addvspace`: [57](#), [110](#), [1278](#), [1278](#), [4000](#)
`__enumext_multi_set_vskip`: [56](#), [1233](#), [1233](#), [1280](#)
`\l__enumext_multicols_above_ii_skip` ... [1252](#)
`\l__enumext_multicols_above_iii_skip` .. [1261](#)
`\l__enumext_multicols_above_iv_skip` ... [1270](#)
`\l__enumext_multicols_above_v_skip` [1297](#), [1311](#), [1321](#), [1512](#)
`\l__enumext_multicols_above_X_skip` [66](#)
`\l__enumext_multicols_below_ii_skip` .. [1379](#), [1388](#), [1392](#), [1404](#), [1409](#)
`\l__enumext_multicols_below_iii_skip` . [1419](#), [1428](#), [1432](#), [1444](#), [1449](#)
`\l__enumext_multicols_below_iv_skip` .. [1458](#), [1467](#), [1471](#), [1483](#), [1488](#)
`\l__enumext_multicols_below_v_skip` [1301](#), [1315](#), [1513](#), [1547](#), [1554](#), [1556](#), [1566](#), [1569](#), [4188](#)
`\l__enumext_multicols_below_X_skip` [66](#)
`\g__enumext_multicols_right_X_skip` [66](#)
`__enumext_multicols_start`: [109](#), [110](#), [3976](#), [3976](#), [4043](#)
`__enumext_multicols_stop`: [110](#), [1675](#), [4006](#), [4006](#), [4059](#)
`__enumext_nested_base_line_fix`: [51](#), [108](#), [981](#), [987](#), [3927](#)
`__enumext_newlabel:nn` [34](#), [40](#), [87](#), [409](#), [409](#), [2897](#), [3328](#)
`\l__enumext_newlabel_arg_one_tl` [34](#), [40](#), [87](#), [95](#), [159](#), [2890](#), [2898](#), [2960](#), [3317](#), [3329](#), [3364](#)
`\l__enumext_newlabel_arg_two_tl` [34](#), [40](#), [86](#), [159](#), [2846](#), [2856](#), [2869](#), [2884](#), [2899](#), [3304](#), [3309](#), [3314](#), [3330](#)
`__enumext_parse_foreach_keys:n` .. [5756](#), [5772](#), [5789](#)
`__enumext_parse_foreach_keys:nn` . [5756](#), [5779](#), [5791](#)
`__enumext_parse_keys:n` [51](#), [71](#), [3915](#), [3915](#), [4094](#)
`__enumext_parse_keys_vii:n` [71](#), [4937](#), [4976](#), [4976](#)
`__enumext_parse_keys_viii:n` . [5207](#), [5253](#), [5253](#)
`__enumext_parse_save_key:n` [84](#), [2689](#), [2694](#), [2694](#)
`__enumext_parse_save_key_vii:n` [84](#), [2684](#), [2694](#), [2702](#)
`__enumext_parse_series:n` [67](#), [71](#), [108](#), [128](#), [2076](#), [2076](#), [3925](#), [3934](#), [4984](#)
`__enumext_parse_store_keys:n` [108](#)
`\l__enumext_parsep_i_skip` [1250](#), [1254](#)
`\l__enumext_parsep_ii_skip` [1259](#), [1263](#)
`\l__enumext_parsep_iii_skip` [1268](#), [1272](#)
`\l__enumext_parsep_vii_skip` [5160](#)
`\l__enumext_parsep_viii_skip` [5435](#)
`\l__enumext_partopsep_v_skip` . [1313](#), [1317](#), [1508](#), [1531](#)
`\l__enumext_partopsep_viii_skip` [1609](#)
`__enumext_phantomsection`: [39](#), [375](#), [402](#), [406](#), [422](#)
`__enumext_pre_itemsep_skip`: [58](#), [59](#), [1341](#), [1370](#), [1370](#)
`__enumext_print_footnote`: .. [424](#), [446](#), [510](#), [515](#)
`__enumext_print_footnote_mini`: [424](#), [476](#), [537](#), [542](#)
`__enumext_print_footnote_standar`: [488](#), [504](#), [568](#)
`__enumext_print_footnote_starred`: [488](#), [533](#), [548](#), [552](#)
`__enumext_print_keyans_box:NN` [86](#), [2814](#), [2814](#), [2827](#), [2975](#), [2986](#), [3446](#), [3478](#), [5359](#), [5376](#)
`\l__enumext_print_keyans_cmd_bool` [120](#), [1844](#), [1858](#), [1876](#), [3923](#), [3932](#), [4065](#), [4982](#), [5003](#), [5550](#), [5557](#)
`\l__enumext_print_keyans_i_tl` [5518](#), [5551](#)
`\l__enumext_print_keyans_ii_tl` ... [5524](#), [5552](#)
`\l__enumext_print_keyans_iii_tl` .. [5530](#), [5553](#)
`\l__enumext_print_keyans_iv_tl` ... [5536](#), [5554](#)
`\l__enumext_print_keyans_star_bool` . [51](#), [140](#), [120](#), [993](#), [1005](#), [5575](#), [5580](#)
`\l__enumext_print_keyans_starred_tl` [139](#), [140](#), [120](#), [5512](#), [5573](#)
`\l__enumext_print_keyans_vii_tl` [139](#), [5542](#), [5555](#)
`\l__enumext_print_keyans_X_tl` [120](#)
`__enumext_printkeyans:nnn` [140](#), [5547](#), [5556](#), [5560](#)
`__enumext_redefine_item`: [100](#), [3538](#), [3538](#), [3855](#)
`\l__enumext_ref_key_arg_t` [46](#)
`\l__enumext_ref_key_arg_tl` [37](#), [736](#), [737](#), [749](#), [780](#), [783](#), [791](#), [797](#), [805](#), [844](#), [845](#), [853](#)
`\l__enumext_ref_the_count_tl` . [46](#), [37](#), [742](#), [748](#), [788](#), [791](#), [802](#), [805](#), [850](#), [853](#)
`__enumext_register_default_label_wd:Nn` [599](#), [599](#), [604](#), [605](#), [606](#), [607](#), [608](#)
`__enumext_remove_extra_parsep_vii`: .. [4956](#), [5184](#), [5184](#)
`__enumext_remove_extra_parsep_viii`: . [5225](#), [5463](#), [5463](#)

```

\l__enumext_renew_counter_v_tl . 851, 859, 861
\l__enumext_renew_counter_vii_tl 789, 814, 816
\l__enumext_renew_counter_viii_tl . 803, 821,
823
\l__enumext_renew_counter_X_tl . . . . . 37
\__enumext_renew_footnote: . . 424, 428, 494, 499
\__enumext_renew_footnote_mini: 424, 458, 524,
529
\__enumext_renew_footnote_standar: 488, 488,
560
\__enumext_renew_footnote_starred: 488, 520,
5154, 5429
\__enumext_reset_count_resume:nn . 2238, 2266,
2270, 2272, 2277, 2282, 2289
\__enumext_reset_count_resume_all:n . . 2238,
2242, 2278
\__enumext_reset_count_resume_levels:n 2238,
2247, 2285
\__enumext_reset_global_bool: . . 325, 328, 337
\__enumext_reset_global_int: . . . 325, 327, 331
\__enumext_reset_global_tl: . . . . 325, 329, 343
\__enumext_reset_global_vars: . 38, 94, 325, 325,
3258
\__enumext_resume:n . . . . . 72, 1866, 2119, 2119
\l__enumext_resume_count_bool . . 46, 866, 1863,
1881, 2070
\__enumext_resume_counter: 69, 1976, 1976, 2073,
2219, 2233
\__enumext_resume_integer_series: . 73, 2149,
2156, 2163, 2166
\__enumext_resume_last_counter: 71, 2046, 2067,
2083, 2088
\__enumext_resume_save_counter: . . . 111, 129
\__enumext_resume_series:n . . 72, 73, 2125, 2140,
2149, 2149
\l__enumext_resume_series_vii_bool 2060, 2160
\l__enumext_resume_series_X_bool . . . . . 46
\__enumext_resume_star: . . . 74, 1846, 2213, 2213
\__enumext_resume_vii:n . . . 72, 1884, 2119, 2134
\l__enumext_rightmargin_vii_dim . . 4668, 4672,
4677
\l__enumext_rightmargin_viii_dim . 4699, 4703,
4708
\__enumext_safe_exec: . . 42, 108, 3904, 3904, 4093
\__enumext_safe_exec_vii: . 42, 4936, 4959, 4959
\__enumext_safe_exec_viii: 133, 5206, 5229, 5229
\__enumext_save_last_keys:n 71, 2046, 2046, 2082,
2087
\g__enumext_save_last_keys_vii_tl 2062, 2063,
2231, 2234
\g__enumext_save_last_keys_X_tl . . . . . 46
\__enumext_scan_tokens:n . . . 94, 201, 201, 3239
\__enumext_second_part: . . 111, 4045, 4045, 4108
\__enumext_second_part_v: . . . 4138, 4194, 4247
\l__enumext_series_name_str . 68, 108, 128, 1841,
1909, 1912, 1917, 1953, 1956, 1960, 2078, 2095, 2097,
2100, 2103, 2108, 2110, 2112, 2114, 3919, 4980
\l__enumext_series_name_tl 68, 73, 46, 1860, 1861,
1878, 1879, 1915, 1928, 1931, 1958, 1969, 1972, 2071,
2154, 2155, 2161, 2162, 2170, 2174, 2208
\__enumext_set_error:nn . . . . . 5666, 5703, 5705
\__enumext_set_item_width: 111, 4070, 4070, 4104
\__enumext_set_parse:n . . . . . 5666, 5677, 5693
\l__enumext_setkey_tmpa_int . . . 111, 5670, 5674
\l__enumext_setkey_tmpa_seq . . 111, 5668, 5678,
5684, 5686, 5688, 5700
\l__enumext_setkey_tmpa_tl . . . . 111, 5676, 5680
\l__enumext_setkey_tmpb_seq . . 111, 5669, 5672,
5676, 5677
\l__enumext_setkey_tmpb_tl 111, 5695, 5697, 5698
\l__enumext_show_answer_bool . 2583, 2602, 2993,
3387, 3400, 3441, 3753, 5355, 5385
\__enumext_show_length:nnn . . 53, 224, 224, 5918,
5919, 5920, 5921, 5922, 5923, 5924, 5925, 5926, 5927,
5933, 5934, 5935, 5936, 5937, 5938, 5939, 5940, 5941,
5942
\l__enumext_show_pos_tmp_int . 120, 3454, 3457,
3472
\l__enumext_show_position_bool . . . 2586, 2605,
2997, 3388, 3401, 3461, 5362
\g__enumext_standar_bool 36, 108, 22, 246, 249, 268,
340, 490, 506, 1891, 2460, 2474, 2851, 2864, 2879,
3946
\l__enumext_standar_bool 108, 111, 22, 1659, 2852,
3911, 4064, 4973
\l__enumext_standar_first_bool 37, 108, 22, 273,
1988, 2181, 2363, 2370
\__enumext_standar_item_vii:w . 130, 5036, 5054,
5056
\__enumext_standar_item_viii:w 135, 5280, 5298,
5300
\__enumext_standar_ref: . . . . 46, 734, 753, 3857
\__enumext_standar_ref:n . . . . . 726, 734, 734
\__enumext_standar_save_counter: . . 68, 1889,
1889, 4067
\__enumext_standar_save_counter_aux: . 1889,
1893, 1904, 1907
\__enumext_standar_unknown_keys:n 3665, 3672,
3677
\__enumext_standar_unknown_keys:nn 3665, 3679,
3681
\__enumext_standard_ref:n . . . . . 46
\__enumext_standard_reset:nn . 2238, 2256, 2264
\__enumext_standard_reset_key: 76, 2296, 2310,
2310
\__enumext_standard_reset_key_star: 76, 2298,
2310, 2320
\g__enumext_starred_bool 36, 128, 22, 256, 259, 282,
341, 1658, 1938, 2434, 2480, 2842, 3298, 4873
\l__enumext_starred_bool 128, 129, 133, 22, 2880,
2915, 2921, 2969, 3912, 4972, 5002, 5241, 5245
\__enumext_starred_columns_set_vii: . . 4650,
4650, 4947
\__enumext_starred_columns_set_viii: . 4650,
4681, 5216
\l__enumext_starred_first_bool 37, 128, 22, 287,
991, 1004, 2004, 2197, 2363, 2370
\__enumext_starred_item_vii:w . 130, 5036, 5053,
5070
\__enumext_starred_item_vii_aux_i:w . . 5036,
5075, 5078
\__enumext_starred_item_vii_aux_ii:w . 5036,
5076, 5081, 5083
\__enumext_starred_item_vii_aux_iii:w 5036,
5086, 5093
\__enumext_starred_item_viii:w 135, 136, 5297,
5315, 5315
\__enumext_starred_item_viii_aux_i:w . . 136,
5315, 5321, 5324

```



```

\__enumext_starred_item_viii_aux_ii:w . 136,
  5315, 5322, 5336, 5338
\__enumext_starred_joined_item_vii:n 124, 130,
  4712, 4712, 5051
\__enumext_starred_joined_item_viii:n . 124,
  135, 4712, 4761, 5295
\__enumext_starred_ref: . . . . 47, 778, 810, 3889
\__enumext_starred_ref:n . . . . 47, 772, 778, 778
\__enumext_starred_reset:n . . 2238, 2251, 2268
\__enumext_starred_reset_key: . 76, 2305, 2307,
  2310, 2334
\__enumext_starred_save_counter: . . 68, 1889,
  1936, 5005
\__enumext_starred_save_counter_aux: . 1889,
  1940, 1948, 1951
\__enumext_starred_unknown_keys:n 3644, 3648,
  3651
\__enumext_starred_unknown_keys:nn 3644, 3653,
  3655
\__enumext_start_counter: . . . 4084, 4084, 4103
\__enumext_start_from:NNn 48, 864, 864, 878, 900,
  906
\l__enumext_start_i_int . . . . . 1991, 2184
\__enumext_start_item_tmp_vii: 127, 4950, 5036,
  5036
\__enumext_start_item_tmp_viii: . . 5219, 5280,
  5280
\__enumext_start_item_vii:w 130, 132, 5062, 5067,
  5090, 5097, 5145, 5145
\__enumext_start_item_viii:w . . 135, 5307, 5312,
  5341, 5420, 5420
\g__enumext_start_line_tl 37, 22, 275, 289, 346,
  2504, 2509, 2514, 2528, 2533, 2538
\__enumext_start_list:nn 39, 105, 364, 366, 4097,
  4235, 4940, 5209
\__enumext_start_list_tag:n . . 4249, 4275, 5155,
  5430
\__enumext_start_mini_vii: 128, 4810, 4810, 4994
\__enumext_start_mini_viii: . . 134, 4875, 4875,
  5264
\__enumext_start_save_ans_msg: . . 76, 77, 2347,
  2347, 2372
\__enumext_start_store_level: . 109, 3940, 3940,
  4096
\__enumext_start_store_level_vii: 129, 4939,
  5008, 5008
\l__enumext_start_vii_int 2007, 2013, 2200, 2206
\l__enumext_start_X_int . . . . . 87
\__enumext_stop_item_tmp_vii: . . 127, 129, 132,
  4949, 4955, 5038, 5147
\__enumext_stop_item_tmp_viii: 135, 5218, 5224,
  5282, 5422
\__enumext_stop_item_vii: 132, 5145, 5147, 5167
\__enumext_stop_item_viii: . . 5420, 5422, 5446
\__enumext_stop_list: 39, 125, 129, 364, 367, 4011,
  4019, 4184, 4191, 4833, 4841, 4898, 4905
\__enumext_stop_list_tag:n . . 4249, 4291, 5170,
  5449
\__enumext_stop_mini_vii: 125, 129, 4810, 4829,
  4998
\__enumext_stop_mini_viii: 134, 4875, 4894, 5268
\__enumext_stop_save_ans_msg: . 76, 2347, 2352,
  3248
\__enumext_stop_start_list_tag: . . 4249, 4283,
  5157, 5432
\__enumext_stop_store_level: . . 109, 110, 3969,
  3969, 4012, 4020
\__enumext_stop_store_level_vii: . . 125, 129,
  4834, 4842, 5008, 5018
\l__enumext_store_active_bool . 32, 77, 99, 1989,
  2005, 2182, 2198, 2379, 3066, 3194, 3944, 3957, 4116,
  4124, 4505, 5010, 5020, 5231, 5247
\__enumext_store_active_keys:n . 83, 108, 2662,
  2662, 3937
\__enumext_store_active_keys_vii:n . 83, 128,
  2662, 2672, 4986
\__enumext_store_addto_prop:n 84, 95, 2738, 2738,
  2746, 2906, 3279, 5346
\__enumext_store_addto_seq:n 85, 96, 2747, 2747,
  2751, 2758, 2772, 2780, 2789, 2803, 2811, 2964, 3369
\__enumext_store_anskey_arg:n . . 88, 90, 93, 94,
  2903, 2903, 3059, 3237
\l__enumext_store_anskey_arg_tl . . 33, 88, 104,
  2912, 2917, 2919, 2924, 2931, 2934, 2944, 2949, 2952,
  2958, 2964
\__enumext_store_anskey_env:n . 94, 3188, 3192,
  3222
\l__enumext_store_anskey_env_tl . . 33, 94, 104,
  3224, 3226, 3228, 3231, 3239
\__enumext_store_anskey_safe_outer: . . 91, 93
\l__enumext_store_columns_break_bool . 2914,
  3015, 3101
\l__enumext_store_current_label_tl 32, 95, 96,
  136, 99, 3263, 3266, 3269, 3275, 3277, 3279, 3336,
  3339, 3342, 3348, 3350, 3360, 3369, 5326, 5331, 5332,
  5345, 5346, 5348
\l__enumext_store_current_opt_arg_tl . 32, 97,
  136, 99, 3379, 3384, 3391, 3397, 3404, 5334
\__enumext_store_internal_ref: . . 86, 88, 2828,
  2828, 2909
\l__enumext_store_item_join_int . . 2922, 2926,
  3018, 3104
\l__enumext_store_item_star_bool . 2929, 3020,
  3106
\l__enumext_store_item_symbol_sep_dim 2941,
  2946, 3025, 3111
\l__enumext_store_item_symbol_tl . 2932, 2936,
  3023, 3109
\l__enumext_store_keyans_item_opt_sep_v_-
  tl . . . . . 3273, 3275, 3346, 3348
\l__enumext_store_keyans_item_opt_sep_-
  viii_tl . . . . . 5329, 5331
\__enumext_store_level_close: . 85, 2752, 2776,
  3973
\__enumext_store_level_close_vii: . 85, 2783,
  2807, 5024
\__enumext_store_level_open: 85, 109, 2752, 2752,
  3952, 3965
\__enumext_store_level_open_vii: . . 85, 2783,
  2783, 5014
\g__enumext_store_name_tl . 32, 77, 99, 345, 352,
  353, 354, 355, 2355, 2381, 2503, 2508, 2513, 2527,
  2532, 2537, 3246
\l__enumext_store_name_tl . 32, 77, 78, 99, 1896,
  1899, 1919, 1941, 1944, 1962, 1993, 2009, 2186, 2202,
  2350, 2359, 2360, 2381, 2382, 2384, 2385, 2387, 2389,
  2390, 2392, 2394, 2395, 2419, 2740, 2742, 2749, 2892,
  2893, 3005, 3319, 3320, 3471, 5370
\l__enumext_store_ref_key_bool 88, 2599, 2907,

```

[2955](#), [3283](#), [3357](#)
`\l__enumext_store_save_key_vii_bool` .. [2674](#),
[2704](#)
`\l__enumext_store_save_key_vii_tl` [2676](#), [2677](#),
[2705](#), [2706](#), [2787](#), [2795](#), [2799](#), [2803](#)
`\l__enumext_store_save_key_X_bool` .. [83](#), [120](#)
`\l__enumext_store_save_key_X_tl` [83](#), [120](#)
`\l__enumext_store_upper_level_X_bool` .. [120](#)
`__enumext_storing_exec`: .. [77](#), [2357](#), [2373](#), [2377](#)
`__enumext_storing_set:n` [76](#), [77](#), [2342](#), [2357](#), [2357](#)
`\l__enumext_the_counter_v_tl` [850](#)
`\l__enumext_the_counter_vii_tl` [788](#)
`\l__enumext_the_counter_viii_tl` [802](#)
`\l__enumext_the_counter_X_tl` [37](#)
`__enumext_tmp:n` [32](#), [36](#), [39](#), [45](#), [48](#), [53](#), [58](#), [65](#), [66](#), [73](#),
[81](#), [86](#), [87](#), [98](#), [130](#), [137](#), [162](#), [166](#), [173](#), [193](#), [590](#), [598](#),
[631](#), [640](#), [1837](#), [1852](#), [1853](#), [1872](#), [1978](#), [1995](#), [1998](#),
[2168](#), [2188](#), [2191](#), [2215](#), [2227](#), [2292](#), [2302](#), [2322](#), [2331](#),
[2338](#), [2346](#), [2398](#), [2416](#), [2592](#), [2661](#), [2680](#), [2693](#), [2830](#),
[2837](#), [2838](#), [2859](#), [2872](#), [2875](#), [2886](#), [3285](#), [3292](#), [3619](#),
[3629](#), [3665](#), [3676](#), [3826](#), [3867](#), [3868](#), [3903](#)
`__enumext_tmp:nn` [641](#), [662](#), [663](#), [697](#), [698](#), [713](#), [895](#),
[920](#), [921](#), [936](#), [1017](#), [1039](#), [1040](#), [1060](#), [1114](#), [1122](#),
[1123](#), [1137](#), [1202](#), [1218](#), [1219](#), [1232](#), [1726](#), [1742](#), [2557](#),
[2591](#), [3603](#), [3618](#)
`__enumext_tmp:nnn` [714](#), [730](#), [731](#), [732](#), [733](#), [760](#), [776](#),
[777](#)
`__enumext_tmp:nnnnnn` [937](#), [962](#), [965](#), [968](#), [970](#), [972](#),
[975](#), [978](#)
`__enumext_tmp:w` [5491](#), [5494](#)
`\l__enumext_tmpa_vii_int` [4660](#), [4663](#), [4672](#), [4703](#)
`\l__enumext_tmpa_viii_int` [4691](#), [4694](#)
`\l__enumext_tmpa_X_dim` [173](#)
`\l__enumext_tmpa_X_int` [173](#)
`\l__enumext_topsep_v_skip` [1299](#), [1303](#), [1502](#), [4498](#)
`\l__enumext_topsep_vii_skip` .. [1579](#), [1588](#), [1592](#)
`\l__enumext_topsep_viii_skip` . [1601](#), [1623](#), [1627](#)
`__enumext_unskip_unkern`: .. [36](#), [230](#), [230](#), [1352](#),
[1524](#), [4014](#), [4015](#), [4055](#), [4186](#), [4187](#), [4204](#), [5161](#), [5162](#),
[5436](#), [5437](#)
`\l__enumext_vspace_a_star_v_bool` [1775](#)
`\l__enumext_vspace_a_star_vii_bool` ... [1797](#)
`\l__enumext_vspace_a_star_viii_bool` ... [1808](#)
`\l__enumext_vspace_a_star_X_bool` [87](#)
`__enumext_vspace_above`: [65](#), [110](#), [1743](#), [1743](#), [4025](#)
`__enumext_vspace_above_v`: . [66](#), [1771](#), [1771](#), [4140](#)
`\l__enumext_vspace_above_v_skip` .. [1773](#), [1777](#),
[1779](#)
`__enumext_vspace_above_vii`: [66](#), [128](#), [1793](#), [1793](#),
[4991](#)
`\l__enumext_vspace_above_vii_skip` [1795](#), [1799](#),
[1801](#)
`__enumext_vspace_above_viii`: . [66](#), [1793](#), [1804](#),
[5262](#)
`\l__enumext_vspace_above_viii_skip` [1806](#), [1810](#),
[1812](#)
`\l__enumext_vspace_b_star_v_bool` [1786](#)
`\l__enumext_vspace_b_star_vii_bool` ... [1819](#)
`\l__enumext_vspace_b_star_viii_bool` ... [1830](#)
`\l__enumext_vspace_b_star_X_bool` [87](#)
`__enumext_vspace_below`: [66](#), [111](#), [1757](#), [1757](#), [4063](#)
`__enumext_vspace_below_v`: . [66](#), [1782](#), [1782](#), [4213](#)
`\l__enumext_vspace_below_v_skip` .. [1784](#), [1788](#),
[1790](#)
`__enumext_vspace_below_vii`: [67](#), [129](#), [1815](#), [1815](#),
[5001](#)
`\l__enumext_vspace_below_vii_skip` [1817](#), [1821](#),
[1823](#)
`__enumext_vspace_below_viii`: . [67](#), [1815](#), [1826](#),
[5270](#)
`\l__enumext_vspace_below_viii_skip` [1828](#), [1832](#),
[1834](#)
`__enumext_widest_from:nnnn` .. [49](#), [879](#), [879](#), [894](#),
[913](#)
`\g__enumext_widest_label_tl` [31](#), [44](#), [54](#), [619](#), [623](#),
[627](#)
`\l__enumext_wrap_label_opt_v_bool` [3699](#)
`\l__enumext_wrap_label_opt_vii_bool` [130](#), [5061](#)
`\l__enumext_wrap_label_opt_viii_bool` .. [135](#),
[5306](#)
`\l__enumext_wrap_label_opt_X_bool` [87](#)
`\l__enumext_wrap_label_v_bool` [3695](#), [3699](#), [3706](#),
[3752](#), [3760](#), [4548](#)
`\l__enumext_wrap_label_vii_bool` .. [130](#), [5061](#),
[5065](#), [5073](#), [5137](#)
`\l__enumext_wrap_label_viii_bool` . [135](#), [5306](#),
[5310](#), [5319](#), [5384](#), [5393](#)
`\l__enumext_wrap_label_X_bool` [87](#)
`__enumext_wrapper_label_v:n` . [3758](#), [3762](#), [4567](#)
`__enumext_wrapper_label_vii:n` [5139](#)
`__enumext_wrapper_label_viii:n` .. [5391](#), [5395](#)
`\l__enumext_write_anskey_env_bool` .. [33](#), [104](#),
[3117](#), [3142](#)
`\l__enumext_write_anskey_env_file_iow` .. [33](#),
[104](#), [3167](#), [3168](#), [3169](#)
`\l__enumext_write_anskey_env_file_name_-`
`tl` [33](#), [104](#), [3118](#), [3228](#)
`\l__enumext_write_aux_file_tl` . [34](#), [87](#), [96](#), [159](#),
[2895](#), [2901](#), [3326](#), [3332](#)
enumext* [5](#), [4934](#)
enumXi [573](#)
enumXii [573](#)
enumXiii [573](#)
enumXiv [573](#)
enumXv [573](#)
enumXvi [573](#)
enumXvii [573](#)
enumXviii [573](#)
Environments provide by enumext:
anskey* [30](#), [32](#), [33](#), [35](#), [77](#), [82](#), [83](#), [87](#), [89](#), [92](#), [109](#), [129](#), [138](#),
[140](#), [145](#), [148](#)
enumext* [30](#), [31](#), [34-36](#), [40-44](#), [47](#), [49](#), [50](#), [52](#), [53](#), [55](#), [62](#),
[63](#), [66-68](#), [71](#), [72](#), [74-79](#), [82-85](#), [87](#), [88](#), [90](#), [94](#), [95](#), [101](#),
[102](#), [105](#), [107-109](#), [115](#), [123-126](#), [129](#), [131-134](#),
[137-141](#), [143](#), [146](#), [149](#), [151](#)
enumext . [30](#), [31](#), [35](#), [36](#), [40-44](#), [46-58](#), [61](#), [63-68](#), [71](#), [72](#),
[74-79](#), [82-85](#), [87](#), [88](#), [90](#), [94](#), [95](#), [99-101](#), [103](#), [105](#), [109](#),
[112](#), [113](#), [117](#), [122](#), [125](#), [128](#), [129](#), [131](#), [133](#), [139-141](#),
[143](#), [146](#), [147](#), [150](#)
keyans* [30-32](#), [34-37](#), [40-43](#), [47-50](#), [52](#), [53](#), [55](#), [62](#), [63](#), [66](#),
[67](#), [77](#), [78](#), [81](#), [82](#), [84](#), [93](#), [95](#), [97](#), [102](#), [105](#), [107](#), [115](#), [123](#),
[124](#), [133](#), [134](#), [146](#), [149](#), [151](#)
keyanspic .. [30-32](#), [34](#), [37](#), [43](#), [48](#), [77](#), [78](#), [81](#), [84](#), [85](#), [93](#),
[95-97](#), [102](#), [115-121](#), [149](#)
keyans [30-32](#), [34](#), [36](#), [37](#), [40](#), [41](#), [43](#), [44](#), [48-50](#), [52](#), [53](#), [55](#),
[57](#), [61](#), [63-66](#), [77](#), [78](#), [81](#), [82](#), [84](#), [85](#), [93](#), [95-98](#), [102-105](#),
[112-114](#), [116](#), [118](#), [121](#), [125](#), [134](#), [146](#), [149](#)

Environments:

center	122
description	101, 122
enumerate	122
flushleft	122
flushright	122
itemize	122
list	35, 38, 39, 50, 90, 101, 105, 110, 112, 115, 116, 118, 119, 122, 125
lrbox	132
minipage	35, 38, 39, 41, 42, 55, 58, 59, 116, 117, 120, 122, 125, 126, 132
multicols	56–59, 64, 109–111
quotation	122
quote	122
tabbing	122
trivlist	122
verbatim	122
verse	122

exp commands:

\exp_after:wN	5494
\exp_args:Ne	1980, 2126, 2172, 2220, 3236, 3930, 5482
\exp_args:NV	3031, 3128, 3632, 3653, 3679, 5791
\exp_not:N	43, 622, 748, 791, 805, 853, 1070, 1073, 1084, 1085, 1086, 1097, 1098, 1109, 1110, 2960, 3002, 3003, 3362, 3468, 3469, 5367, 5368, 5491
\exp_not:n	277, 291, 304, 312, 320, 688, 708, 748, 749, 791, 805, 853, 1071, 2035, 2044, 2570, 2619, 2723, 2736, 2898, 2926, 2936, 2946, 2960, 2961, 3329, 3364, 3366, 4362, 5609, 5617, 5823, 5828

F

\fbox	2626
\fboxrule	2626
\fboxsep	2626
file commands:	
\file_if_exist:nTF	3144
\file_input_stop:	6267
first	1123
font	641
\footnote	40
\footnote	40, 430, 460
\footnotemark	440, 470
\footnotesize	3003, 3469, 5368
\footnotetext	426
force-eol	3099
\foreachkeyans	19, 143, 5756

G

\getkeyans	19, 138, 5480
group commands:	
\group_begin:	3001, 3046, 3467, 5366, 5549
\group_end:	3008, 3062, 3475, 5373, 5558

H

\hbadness	5172, 5451
hbox commands:	
\hbox_overlap_left:n	2818, 3534, 5130
\hbox_set:Nn	611, 4425
\hbox_set_end:	5171, 5450
\hbox_set_to_wd:Nnw	5148, 5423
\hfill	671, 676, 682, 683, 1682, 1709, 2960, 3362, 4837, 4901
hook commands:	
\hook_gput_code:nnn	5, 204, 208, 212, 375
\hook_gset_rule:nnnn	376
\hyperlink	89, 96

\hyperlink	2960, 3362
\hypertarget	39
\hypertarget	401

I

\IfDocumentMetadataT	4277, 4285, 4293, 4329, 4337, 4345, 4449, 4458, 4466, 4473, 4478, 4526, 4535, 4625, 4633, 4835, 4899, 4946, 4954, 5106, 5215, 5223
\IfDocumentMetadataTF	492, 508, 522, 535, 3551, 3734
\IfHyperBoolean	383
\IfPackageLoadedT	379
\IfPackageLoadedTF	7, 391
\ignorespaces	1073, 1086, 1098, 1110, 4438, 4951, 5034, 5067, 5090, 5097, 5143, 5163, 5220, 5278, 5312, 5341, 5418, 5438
\inputlineno	277, 291, 304, 312, 320
int commands:	
\int_add:Nn	4745, 4794
\int_case:nn	1247, 1372, 2429, 2455, 2494, 2518
\int_case:nnTF	232
\int_compare:nNnTF	556, 781, 795, 812, 819, 1342, 1361, 1515, 1533, 1645, 1664, 1676, 1704, 1894, 1985, 2001, 2048, 2058, 2080, 2085, 2092, 2106, 2121, 2136, 2151, 2158, 2178, 2194, 2225, 2229, 2312, 2329, 2542, 2548, 3070, 3074, 3078, 3086, 3198, 3202, 3206, 3244, 3264, 3302, 3307, 3312, 3337, 3452, 3909, 3920, 3949, 3962, 3978, 3993, 4008, 4049, 4125, 4129, 4157, 4182, 4198, 4397, 4509, 4513, 4715, 4725, 4741, 4764, 4774, 4790, 4964, 4968, 5012, 5022, 5174, 5186, 5236, 5248, 5453, 5465, 5674, 5806
\int_compare_p:nNn	247, 257, 269, 270, 283, 284, 1651, 1652, 2254, 2255, 2435, 2461, 2843, 2853, 2865, 2866, 2881, 2922, 3959
\int_decr:N	4744, 4793
\int_eval:n	362, 908, 2258, 2742, 2893, 3003, 3320, 3469, 4088, 4228, 4733, 4782, 4945, 5214, 5368
\int_from_alph:n	873, 887
\int_from_roman:n	875, 889
\int_gadd:Nn	4746, 4795
\int_gdecr:N	2438, 2443, 2447, 2451, 2464
\int_gincr:N	2905, 3372, 3488, 3522, 3712, 4038, 4149, 4553, 5040, 5116, 5284, 5350
\int_gset:Nn	438, 468, 2487
\int_gset_eq:NN	435, 465, 1898, 1911, 1921, 1930, 1943, 1955, 1964, 1971
\int_gzero:N	333, 334, 335, 1690, 1717, 2316, 2326, 2336, 2554, 4054, 4203, 5197, 5477
\int_if_exist:NnTF	1896, 1927, 1941, 1969, 2099, 2112, 2170, 2314, 2324, 2392
\int_incr:N	3085, 3454, 3908, 4120, 4396, 4963, 5039, 5235, 5283
\int_mod:nn	5188, 5467
\int_new:N	16, 17, 18, 19, 20, 21, 74, 91, 113, 128, 140, 141, 152, 153, 154, 156, 167, 168, 176, 177, 178, 179, 180, 2102, 2114, 2395
\int_set:Nn	869, 873, 875, 1980, 1991, 2007, 2013, 2172, 2184, 2200, 2206, 3670, 4619, 4620, 4660, 4691, 4714, 4720, 4736, 4763, 4769, 4785, 5172, 5451, 5670, 5808
\int_set_eq:NN	3841, 3885, 4743, 4792
\int_sign:n	2489
\int_step_function:nnN	1995, 1998, 2188, 2191, 2227, 2331, 2859, 2872, 2886
\int_step_function:nnnN	5812
\int_step_inline:nn	5722
\int_step_inline:nnn	4621

\int_to_roman:n	216, 1980, 1982, 2170, 2172, 2174, 2217, 2222, 2266, 2314, 2316, 2324, 2326, 2839, 2876
\int_use:N	355, 360, 361, 1343, 1362, 1677, 1982, 1993, 2009, 2015, 2126, 2174, 2186, 2202, 2208, 2221, 3862, 3931, 3979, 3988, 4003, 4009, 4088, 4228, 4718, 4719, 4731, 4767, 4768, 4780, 4945, 5214, 6159, 6163, 6169, 6173, 6250
\int_zero:N	3457, 5178, 5457
iow commands:	
\iow_char:N	3225, 3226
\iow_close:N	3169
\iow_new:N	108
\iow_now:Nn	3168
\iow_open:Nn	3167
\item	99, 103, 129, 132, 134, 137, 368, 2760, 2766, 2791, 2797, 2919, 3339, 3342, 3540, 3716, 4453, 4454, 4948, 4950, 5217, 5219, 5348
\item*	5, 17, 81, 3714
item-join	3013, 3099
item-pos*	3013, 3099, 3603
item-star	3013, 3099
item-sym*	3013, 3099, 3603
\itemindent	106
\itemindent	105
itemindent	1017
\itemsep	4442
\itemwidth	572, 2626, 4072, 4078, 4217, 4223, 4754, 4758, 4803, 4807

K

keyans	16, 4230
keyans*	16, 5204
keyanspic	17, 4444

Keys for \anskey provide by enumext:

break-col	88, 90
force-eol	91
item-join	88, 90
item-pos*	88, 90
item-star	88, 90
item-sym*	88, 90
overwrite	91
write-env	91

Keys for anskey* provide by enumext:

break-col	88, 90
force-eol	91
item-join	88, 90
item-pos*	88, 90
item-star	88, 90
item-sym*	88, 90
overwrite	91
write-env	91

Keys for environments provide by enumext:

above*	32, 51, 65, 66, 110, 128
above	32, 51, 65, 66, 110, 128, 134
after	53, 54, 111, 129, 134
align	32, 45, 97, 99, 100, 104, 131, 145
base-fix	51, 70, 84, 108
before*	53, 54, 110, 128, 134
before	53, 54
below*	32, 65-67, 111, 129
below	32, 65-67, 111, 129, 134
check-ans	34, 35, 37, 76-81, 84, 94, 96, 111, 112, 129, 133, 147
columns-sep	55, 109, 132
columns	32, 55, 65, 109

first	53, 54, 132
font	44, 100, 104, 120, 131
item-pos*	99, 101
item-sym*	33, 99, 101
itemindent	32, 52, 99, 103-105, 132
itemsep	50, 107, 132
label-pos	117, 118, 120, 121
label-sep	117
labelsep	44, 106, 131
labelwidth	43, 44, 46-49, 106, 131
label	31, 43, 44, 46, 48, 49, 118, 122
layout-sep	117
layout-sty	117, 121, 122
layout-top	117
lisparindent	107
list-indent	31, 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, 97, 98
mark-sep	33, 82, 84, 136
mini-env	32, 40-42, 55, 64, 65, 84, 110, 122, 125, 126, 128, 129, 134
mini-right*	32, 35, 55, 84, 126, 128, 129
mini-right	32, 35, 55, 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
noitemsep	50
nosep	50
overwrite	33, 92
parindent	107
parsep	50, 107, 118, 132
partopsep	50
ref	31, 46-48, 105, 147
reset*	70, 75, 76, 84, 141
reset	70, 75, 76, 84, 141
resume*	31, 43, 67, 69-71, 74, 76, 77, 84, 111, 129, 141
resume	31, 38, 43, 49, 67-74, 76, 77, 84, 111, 129, 141, 151
rightmargin	52, 123
save-ans	32, 38, 68, 69, 71, 73, 76-78, 80, 83-85, 90, 91, 93-96, 103, 112, 120, 131, 133, 134, 136, 138, 139, 141, 147
save-key	33, 71, 83, 84, 108, 128
save-pos	84
save-ref	34, 40, 82, 84, 86, 88, 89, 95, 96, 103, 136
save-sep	81, 84, 95, 136
series	31, 67, 68, 71, 73, 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, 48, 49, 71
start	32, 36, 48, 49, 71
store-key	83
topsep	50, 51, 118
widest	31, 36, 49
wrap-ans*	34, 81, 84, 104, 120
wrap-ans	43, 82, 84, 86, 89
wrap-label*	32, 44, 99, 100, 103, 104, 130, 131, 135
wrap-label	32, 44, 99, 100, 103, 104, 118, 120, 130, 131, 135

wrap-opt	81, 84, 97, 103, 120
wrap-sep	89
write-env	33, 92
keys commands:	
\keys_define:nn	633, 643, 665, 700, 716, 762, 827, 897, 923, 939, 981, 1019, 1042, 1116, 1125, 1204, 1221, 1728, 1839, 1855, 1873, 2294, 2303, 2340, 2400, 2559, 2594, 2682, 2687, 3013, 3099, 3605, 3621, 3644, 3667, 4351, 5508, 5619, 5748, 5756
\keys_if_exist_p:nn	5744, 5745
\l_keys_key_str	90, 92, 3031, 3128, 3632, 3653, 3679, 5791, 5903
\keys_precompile:nnN	140, 199, 199, 5510, 5516, 5522, 5528, 5534, 5540, 5774
\keys_set:nn	657, 998, 1010, 1227, 1733, 1738, 2126, 2141, 2220, 2234, 2630, 2631, 2635, 2636, 2640, 2641, 2645, 2646, 2650, 2651, 2655, 2656, 3051, 3180, 3922, 3930, 4136, 4369, 4371, 4373, 4375, 4377, 4379, 4381, 4383, 4385, 4387, 4407, 4981, 5257, 5622, 5628, 5634, 5640, 5646, 5651, 5652, 5653, 5654, 5655, 5656, 5657, 5658, 5690, 5800
keyval commands:	
\keyval_parse:NNn	2024, 2712, 5598

L

label	714, 760, 827
label-pos	4351
label-sep	4351
Labels provide by enumext:	
\Alph*	43, 44
\Roman*	43, 44
\alph*	43, 44
\arabic*	43, 44
\roman*	43, 44
labelsep	641
\labelwidth	44
labelwidth	641
\lastnodetype	232
layout-sep	4351
layout-sty	4351
layout-top	4351
\leftmargin	106
\leftmargin	105, 4437
legacy commands:	
\legacy_if:nTF	5101, 5104, 5401, 5404
\legacy_if_gset_false:n	562, 4850
\legacy_if_set_false:n	5103, 5403
\legacy_if_set_true:n	5066, 5089, 5096, 5110, 5311, 5340
\linewidth	110
\linewidth	4033, 4072, 4146, 4217, 4618, 4663, 4694, 4816, 4881
\list	366
list-indent	1017
list-offset	1017
\listparindent	4440
listparindent	1017

M

\makebox	122
\makebox	2820, 3587, 3784, 4543, 4556, 5134, 5413
\makelabel	99, 100, 104, 122
\makelabel	99, 103, 3567, 3583, 3768, 3780
mark-ans	2592, 4351
mark-ans*	2557, 2592

mark-pos	2592, 4351
mark-pos*	2557, 2592
mark-ref	2592
mark-sep	2592, 4351
mark-sep*	2557, 2592
midpenalty	921
mini-env	1202
mini-sep	1202
\minipage	372
\miniright	12, 63, 1643, 1694, 1721, 4052, 4201
mode commands:	
\mode_if_math:TF	3094, 3217
\mode_if_vertical:TF	1281, 1309, 1329, 1353, 1504, 1525
\mode_leave_vertical:	996, 1007, 1070, 1084, 2816, 3532, 5128
mode-box	631
msg commands:	
\msg_error:nn	1696, 1723, 3055, 3088, 3092, 3184, 3215, 4127, 4131, 4399, 4456, 4511, 4966, 5238, 5250, 5659, 5718
\msg_error:nnn	739, 785, 799, 847, 1647, 1654, 1661, 1692, 1719, 2130, 2145, 2258, 2365, 3037, 3096, 3134, 3196, 3200, 3204, 3208, 3219, 3638, 3659, 3685, 4970, 5243, 5496, 5505, 5591, 5706, 5737, 5746, 5783, 5804
\msg_error:nnnn	3040, 3068, 3072, 3076, 3080, 3137, 3641, 3662, 3688, 4118, 4507, 4515, 5233, 5570, 5786
\msg_error:nnnnn	687, 707, 2569, 2618, 4361
\msg_fatal:nn	3910
\msg_fatal:nnn	576, 593
\msg_info:nnn	9, 12, 381, 393
\msg_line_context:	5863, 5868, 5873, 5878, 5907, 5912, 5917, 5932, 5947, 5951, 5955, 5959, 5963, 5967, 5974, 5981, 5987, 6001, 6005, 6010, 6014, 6018, 6022, 6027, 6031, 6035, 6039, 6044, 6091, 6095, 6100, 6105, 6109, 6114, 6190, 6194, 6199, 6204, 6209, 6213, 6217, 6221, 6225, 6229, 6233, 6237, 6241, 6246, 6251
\msg_log:nnn	2384, 2389, 2394
\msg_log:nnnnn	359, 2527, 2532, 2537
\msg_log:nnnnnn	351
\msg_new:nnn	5831, 5835, 5839, 5843, 5848, 5861, 5865, 5870, 5875, 5880, 5889, 5897, 5901, 5905, 5910, 5915, 5930, 5945, 5949, 5953, 5957, 5961, 5965, 5969, 5978, 5984, 5990, 5994, 5998, 6003, 6008, 6012, 6016, 6020, 6025, 6029, 6033, 6037, 6042, 6077, 6081, 6085, 6089, 6093, 6098, 6103, 6107, 6112, 6188, 6192, 6197, 6202, 6207, 6211, 6215, 6219, 6223, 6227, 6231, 6235, 6239, 6243, 6248, 6256
\msg_new:nnnn	5852, 6047, 6056, 6065, 6071, 6116, 6126, 6136, 6146, 6156, 6166, 6176, 6182, 6253, 6258, 6261, 6264
\msg_term:nnnn	2349, 2354, 3851, 3861, 3894, 3899
\msg_term:nnnnn	2508
\msg_warning:nn	4051, 4200
\msg_warning:nnn	3148, 3152, 3157
\msg_warning:nnnn	2545, 2551, 3798, 3803, 4717, 4730, 4766, 4779
\msg_warning:nnnnn	2503, 2513
\multicolsep	109
\multicolsep	1346, 1518, 3999, 4173

N

\NeedsTeXFormat	3
\NewCommandCopy	368
\newcounter	579, 595

\NewDocumentCommand	1643, 2238, 3043, 4503, 5480, 5547, 5666, 5715, 5793
\NewDocumentEnvironment	3173, 4091, 4230, 4444, 4934, 5204
\newlabel	40
\newlabel	413
no-store	2398
\noindent	4040, 4825, 4890, 5177, 5456
\nointerlineskip	1355, 1358, 1527, 1530, 1684, 1711, 4825, 4890
noitemsep	937
\nopagebreak	1292, 1320, 1355, 1358, 1527, 1530, 1634, 1640
\normalfont	3002, 3468, 5367
nosep	937
O	
\obeyedline	3225, 3226
overwrite	3099
P	
Packages:	
caption	126
enumext	30, 43, 46, 76, 81, 101, 106, 117, 145
enumitem	43
expl3	122
footnotehyper	39, 41, 42
hyperref	34, 35, 39, 40, 89, 96, 131, 145
latex-lab-block	39
ltxcmd	39, 91
ltsockets	115
lua-visual-debug	58
multicol	30, 145
scontents	91
shortlst	122, 127, 132
tagpdf	115
\par	1292, 1320, 1358, 1530, 1634, 1640, 1679, 1684, 1706, 1711, 2968, 4016, 4188, 4206, 4489, 4492, 4638, 4852, 4867, 4913, 4927, 5177, 5456
para commands:	
\para_end:	5194, 5474
\parbox	2626
\parindent	5159, 5434
\parsep	56, 118
\parsep	997, 3886, 4421, 4430
parsep	937
\parskip	5160, 5435
\partopsep	3887, 4204, 4441
partopsep	937
peek commands:	
\peek_meaning:Ntf	5045, 5059, 5074, 5085, 5289, 5304, 5320
\peek_meaning_remove:Ntf	5052, 5296
\peek_remove_spaces:n	3721
\phantomsection	39
\phantomsection	402
prg commands:	
\prg_do_nothing:	406
\prg_new_protected_conditional:Npnn	218, 3140
\prg_replicate:nn	227
\prg_return_false:	222, 3153, 3161
\prg_return_true:	221, 3149, 3158
\printkeyans	20, 139, 5547
prop commands:	
\prop_const_from_keyval:Nn	5707
\prop_count:N	353, 2742, 2893, 3005, 3320, 3471, 5370, 5809

\prop_get:NnNtf	5733
\prop_gput_if_not_in:Nnn	2740
\prop_if_exist:Ntf	2382, 5500, 5802
\prop_item:Nn	5502, 5826
\prop_new:N	2385
\ProvidesExplPackage	4
R	
\raggedcolumns	4002, 4176
\raisebox	4580
\ref	86, 95
ref	714, 760, 827
\refstepcounter	5113, 5406
regex commands:	
\regex_if_match:nnTF	220, 872, 874, 886, 888
\renewcommand	748, 791, 805, 853
\RenewDocumentCommand	430, 460, 1694, 1721, 3225, 3540, 3567, 3583, 3716, 3768, 3780, 4454
\RequirePackage	13
reset	2292
reset*	2292
\resetenumext	11, 74, 2238
resume	1837
resume*	1837
rightmargin	1017
\Roman	44, 48, 49
\Roman	607
\roman	44, 48, 49
\roman	608, 732, 5532
S	
save-ans	2338
save-key	2680
save-ref	2592
save-sep	2557, 2592, 4351
scan commands:	
\scan_stop:	4453, 4948, 5217, 5491, 5494
seq commands:	
\seq_clear:N	5668, 5811
\seq_const_from_clist:Nn	5661
\seq_count:N	354, 4644, 5672
\seq_gclear:N	455, 456, 485, 486
\seq_gput_right:Nn	441, 442, 471, 472, 2749
\seq_if_empty:Ntf	448, 478, 5564, 5686
\seq_if_exist:Ntf	2387, 5562
\seq_if_in:NnTF	5568
\seq_item:Nn	4631
\seq_map_function:NN	5677
\seq_map_inline:Nn	5577, 5585, 5687, 5688
\seq_map_pairwise_function:NNN	450, 480
\seq_new:N	114, 115, 117, 138, 169, 170, 171, 172, 2390
\seq_pop_left:NN	5676
\seq_put_right:Nn	4517, 5684, 5700, 5821
\seq_set_from_clist:Nn	5669
\seq_set_map_e:NNn	5678
\seq_use:Nn	199, 200, 5817
series	1837
\setcounter	883, 887, 889, 4086, 4228, 4486, 4945, 5214
\setenumext	6, 141, 5666
\setenumextmeta	6, 143, 5707
show-ans	2557, 2592, 4351
show-length	1114
show-pos	2557, 2592, 4351

skip commands:

<code>\skip_add:Nn</code>	1252, 1261, 1270, 1283, 1287, 1311, 1315, 1331, 1389, 1391, 1405, 1408, 1429, 1431, 1445, 1448, 1468, 1470, 1484, 1487, 1506, 1555, 1556, 1567, 1569, 4430, 4439
<code>\skip_gset:Nn</code>	1582, 1586, 1590
<code>\skip_gzero_new:N</code>	1577, 1578
<code>\skip_horizontal:N</code>	1085, 1097, 1109, 5131, 5143, 5181, 5418, 5460
<code>\skip_horizontal:n</code>	1071, 2817, 2825, 3533, 3535, 4561, 5030, 5129, 5163, 5274, 5438
<code>\skip_if_eq:nnTF</code>	1250, 1259, 1268, 1375, 1415, 1455, 1543, 1579, 1601, 1745, 1759, 1773, 1784, 1795, 1806, 1817, 1828
<code>\skip_new:N</code>	68, 69, 70, 75, 76, 77, 78, 79, 80, 191
<code>\skip_set:Nn</code>	1235, 1239, 1297, 1301, 1325, 1378, 1379, 1397, 1418, 1419, 1437, 1457, 1458, 1476, 1500, 1546, 1547, 1561, 1581, 1585, 1603, 1607, 1611, 1617, 1621, 1625, 4414
<code>\skip_set_eq:NN</code>	1336, 1337, 1339, 1346, 1511, 1512, 1513, 1518, 3839, 3883, 3886, 5160, 5435
<code>\skip_sub:Nn</code>	1385, 1387, 1401, 1403, 1425, 1427, 1441, 1443, 1464, 1466, 1480, 1482, 1553, 1554, 1565, 1566
<code>\skip_use:N</code>	1237, 1241, 1285, 1289, 1293, 1313, 1317, 1327, 1333, 1746, 1750, 1753, 1760, 1764, 1767, 4016
<code>\skip_vertical:N</code>	563, 566, 1009, 4851, 4865, 5196, 5476
<code>\skip_vertical:n</code>	1008, 5195, 5475
<code>\skip_zero:N</code>	1345, 1359, 1497, 1498, 1499, 1517, 1531, 3887, 3999, 4173, 4441, 4442
<code>\skip_zero_new:N</code>	1576, 1598, 1599, 1600
<code>\c_zero_skip</code>	563, 566, 1009, 1250, 1259, 1268, 1416, 1455, 1579, 1601, 1746, 1760, 1773, 1784, 1795, 1806, 1817, 1828, 4851, 4865, 5196, 5476
<code>\small</code>	5515, 5521, 5527, 5533, 5539, 5545
<code>\smash</code>	3585, 3782
socket commands:	
<code>\socket_assign_plug:nn</code>	4279, 4287, 4295, 4331, 4339, 4347
<code>\socket_new:nn</code>	4249, 4299
<code>\socket_new_plug:nnn</code>	4250, 4258, 4266, 4300, 4308, 4317
<code>\socket_use:n</code>	4332, 4340, 4348
<code>\socket_use:nn</code>	4280, 4288, 4296
<code>start</code>	895
<code>start*</code>	895
<code>start-list-tags</code>	4249, 4299
<code>\stepcounter</code>	434, 464, 4424, 4573
<code>stop-list-tags</code>	4249, 4299
<code>stop-start-tags</code>	4249, 4299
str commands:	
<code>\c_backslash_str</code>	3096, 5868, 5873, 5878, 5883, 5885, 5887, 5892, 5894, 5992, 5996, 6000, 6010, 6014, 6022, 6023, 6027, 6039, 6040, 6044, 6045, 6066, 6068, 6072, 6074, 6114, 6177, 6179, 6183, 6185, 6194, 6195, 6199, 6204, 6205, 6209, 6213, 6217
<code>\c_circumflex_str</code>	110
<code>\c_colon_str</code>	2892, 3319, 5491
<code>\c_left_brace_str</code>	5973, 5980, 5986
<code>\c_percent_str</code>	110
<code>\c_right_brace_str</code>	5973, 5980, 5986
<code>\str_case:nn</code>	240, 297, 3412
<code>\str_case:nnTF</code>	2031, 2039, 2719, 2727, 5605, 5613
<code>\str_clear:N</code>	3919, 4980
<code>\str_const:Nn</code>	109

<code>\str_count:n</code>	227
<code>\str_if_empty:NTF</code>	1909, 1917, 1953, 1960, 2078
<code>\str_if_eq:nnTF</code>	2250, 3843, 3890, 5717
<code>\str_if_in:nnTF</code>	5487
<code>\str_new:N</code>	71, 123, 124, 125, 143, 186
<code>\str_set:Nn</code>	672, 678, 684, 703, 704, 705, 2565, 2566, 2567, 2614, 2615, 2616, 4356, 4359
<code>\str_set_eq:NN</code>	3435, 5358, 5375
<code>\str_use:N</code>	3589
<code>\strut</code>	3585, 3782
<code>\strutbox</code>	1364, 1367, 1378, 1379, 1390, 1392, 1407, 1410, 1418, 1419, 1430, 1432, 1447, 1450, 1457, 1458, 1469, 1471, 1486, 1489, 1535, 1538, 1546, 1547, 1555, 1556, 1568, 1570, 1581, 1582, 1585, 1592, 1605, 1613, 1619, 1627, 4433, 4439, 4489, 4497, 4586

T

tag commands:

<code>\tag_mc_begin:n</code>	4256, 4306, 4315
<code>\tag_mc_begin_pop:n</code>	4272, 4324, 4481, 4483
<code>\tag_mc_end:</code>	4260, 4310, 4319
<code>\tag_mc_end_push:</code>	4253, 4303, 4469
<code>\tag_resume:n</code>	4252, 4302, 4460, 4468, 4537, 4635, 4835, 4899
<code>\tag_struct_begin:n</code>	4254, 4255, 4262, 4263, 4264, 4304, 4305, 4312, 4313, 4314, 4470
<code>\tag_struct_end:n</code>	4261, 4268, 4269, 4270, 4271, 4311, 4320, 4321, 4322, 4323, 4480, 4482, 4954, 5223
<code>\tag_suspend:n</code>	4273, 4325, 4451, 4462, 4475, 4528, 4627, 4946, 5215
<code>\tag_tool:n</code>	4461

TeX and L^AT_EX 2_ε commands:

<code>\@auxout</code>	411
<code>\@currentenv</code>	240, 297
<code>\protected@write</code>	411

tex commands:

<code>\tex_scantokens:D</code>	201
--------------------------------	-----

text commands:

<code>\text_expand:n</code>	5483
<code>\textasteriskcentered</code>	2562, 2609
<code>\textborn</code>	3609
<code>\textreferencemark</code>	2597
<code>\thepage</code>	417

tl commands:

<code>\c_space_tl</code>	3391, 3404, 5917, 5932, 5955, 5959, 6158, 6159, 6168, 6169, 6229, 6233, 6251
<code>\tl_clear:N</code>	670, 677, 2154, 2161, 2555, 2666, 2676, 2697, 2705, 2912, 3263, 3336, 5326
<code>\tl_clear_new:N</code>	617
<code>\tl_const:Nn</code>	601
<code>\tl_gclear:N</code>	345, 346, 347, 2052, 2062, 3578, 3598, 4871, 4931, 5132
<code>\tl_gclear_new:N</code>	2094, 2108
<code>\tl_gput_right:Nn</code>	602
<code>\tl_greplace_all:Nnn</code>	623
<code>\tl_gset:Nn</code>	274, 275, 288, 289, 2053, 2063, 2096, 2109, 2381, 3509, 5080
<code>\tl_gset_eq:NN</code>	619, 3505, 5125
<code>\tl_if_blank:nTF</code>	3035, 3053, 3132, 3182, 3636, 3657, 3683, 5123, 5781
<code>\tl_if_empty:NTF</code>	737, 755, 783, 797, 814, 821, 845, 859, 1861, 1879, 1915, 1919, 1958, 1962, 2217, 2231, 2360, 2419, 2756, 2787, 2932, 3246, 3273, 3346, 3384, 3397, 3530, 4642, 5329, 5698
<code>\tl_if_empty_p:N</code>	2071
<code>\tl_if_exist:NTF</code>	2123, 2138

`\tl_if_novalue:nTF` 432, 462, 2245, 3049, 3178, 3271, 3344, 3377, 3484, 3503, 3511, 3693, 3917, 4405, 4978, 5255, 5327

`\tl_map_inline:Nn` 620

`\tl_new:N` 29, 30, 31, 34, 37, 38, 41, 42, 46, 51, 55, 56, 92, 93, 94, 100, 101, 102, 103, 104, 105, 107, 111, 112, 116, 118, 119, 120, 129, 132, 133, 150, 159, 160, 161, 164, 185

`\tl_put_left:Nn` 2764, 2795, 2917, 4855, 4916, 5345, 5348

`\tl_put_right:Nn` . 618, 851, 2768, 2799, 2846, 2856, 2869, 2884, 2890, 2895, 2919, 2924, 2931, 2934, 2944, 2949, 2952, 2958, 3231, 3266, 3269, 3275, 3277, 3304, 3309, 3314, 3317, 3326, 3339, 3342, 3348, 3350, 3360, 5331, 5332

`\tl_remove_all:Nn` 5697

`\tl_remove_once:Nn` 2834, 3289

`\tl_replace_all:Nnn` 622, 3226, 5732

`\tl_reverse:N` 2833, 2835, 3288, 3290

`\tl_set:Nn` . 43, 244, 254, 301, 302, 309, 310, 317, 318, 578, 671, 676, 682, 683, 736, 746, 780, 789, 803, 844, 1068, 1082, 1095, 1107, 1860, 1878, 2155, 2162, 2359, 2667, 2677, 2698, 2706, 2999, 3118, 3224, 3379, 3465, 3624, 3647, 3671, 4390, 5334, 5364, 5695, 5731, 5801

`\tl_set_eq:NN` .. 628, 742, 788, 802, 850, 2832, 3287, 3300, 3433, 5357

`\tl_to_str:n` 2123, 2127, 2138, 2142, 5483

`\tl_trim_spaces:n` ... 618, 5684, 5695, 5701, 5717

`\tl_use:N` 624, 627, 757, 816, 823, 861, 1140, 1144, 1148, 1152, 1156, 1160, 1164, 1168, 1172, 1176, 1180, 1184, 1188, 1192, 1196, 1200, 2822, 2839, 2847, 2858, 2871, 2876, 2887, 3492, 3498, 3526, 3569, 3571, 3577, 3592, 3696, 3700, 3707, 3770, 3773, 3775, 3788, 4098, 4236, 4558, 4566, 4862, 4923, 5136, 5164, 5165, 5415, 5439,

5444, 5551, 5552, 5553, 5554, 5555, 5573, 5680, 5799

token commands:

`\token_to_str:N` 413

`\topsep` 4204, 4439

`topsep` 937

`\topskip` 1345, 1517

U

`\unkern` 235

`unknown` 3013, 3099, 3619, 3644, 3665

`\unskip` 234

use commands:

`\use:N` 228, 3574, 3595, 4100

`\use:n` 2022, 2710, 5489, 5596

`\use_none:nn` 405, 5738

`\usecounter` 3842, 3888

V

`\value` 1899, 1944, 1956, 1972

vbox commands:

`\vbox_set:Nn` 4530

`\vbox_set_top:Nn` 4860, 4921

`\vspace` 997, 1750, 1753, 1764, 1767, 1777, 1779, 1788, 1790, 1799, 1801, 1810, 1812, 1821, 1823, 1832, 1834

W

`widest` 895

`wrap-ans` 2592

`wrap-ans*` 2557, 2592, 4351

`wrap-label` 641

`wrap-label*` 641

`wrap-opt` 2557, 2592, 4351

`write-env` 3099