

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-11-10^{*}

©2024 by Pablo González[†]

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` and `scontents` packages.

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	3	6.1.2	Keys for wrap and marks	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	4	6.2.1	Keys for <code>\anskey</code>	14
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	16
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	17
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	17
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	18
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	18
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	18
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	19
5	The <code>keyval</code> system	7	7	Full examples	20
5.1	Keys for <code>label</code> and <code>ref</code>	7	8	Tagged PDF examples	22
5.2	Keys for spaces	8	9	The way of non-enumerated lists	23
5.2.1	Vertical spaces	8	10	References	25
5.2.2	Horizontal spaces	9	11	Change history	25
5.3	Keys for add code	9	12	Index of Documentation	26
5.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	10	13	Implementation	28
5.5	Keys for <code>multicols</code>	10	14	Index of Implementation	146
5.6	Keys for <code>minipage</code>	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

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 multicols`, vertical alignment

^{*}This file describes a documentation for v1.0, last revised 2024-11-10.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.
The enumext package loads and requires multicol[3] and contents[4] packages, need to have a modern T_EX distribution such as T_EX Live or MiK_TE_X. It has been tested with the standard classes provided by L^AT_EX: book, report, article and letter on 10pt, 11pt and 12pt.

The minimum requirement is L^AT_EX release 2024-11-01.

1 Introduction

In the L^AT_EX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[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

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
- (a) $\alpha > \delta$

(b) L^AT_EX is cool?

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

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

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

3. True False

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
- * $(x - 1)^2$

* $3(x + y + z)$

(a) $\alpha > \delta$

(b) L^AT_EX is cool?
- * Yes

* Yes, dnf

* doesn’t exist for now :(

* xsim

* very good

* 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

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

2. Second type of questions

3. Third type of questions
- A) value

B) correct

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

D) value

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

5. Question with image on right side:
- A) value

B) value

C) value

D) correct

E) value
- A) value

B) value

C) value

D) correct

E) value

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 Internal counters

The package `enumext` uses internally 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 §5.5).

- 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 §5.6).

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem`[6] package to customize the references which is activated by the `ref` key (§5.1), the standard \LaTeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§6.1.1) when the key `save-ans` (§6.1) is active.

1.3.6 Support for \footnote

The `enumext*` and `keyans*` environments and the `mini-env` key use the `minipage` environment in their implementation but in a transparent way for the user, i.e. it is only used for typesetting and not directly. The `enumext` package provides an *internal implementation* for the command `\footnote` compatible with the `hyperref` package to work in the same way as if it were used anywhere in the document.

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

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

```
\IfDocumentMetadataTF{ }
{
  \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*`.

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

2.1 The environment enumext

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

Example with `columns=2`

1. This text is in the first level.
- A. This text is in the 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`[15] and `tasks`[16] 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.

- 🔴 The key `save-ans` that activate the “storage system” must NOT be passed through this command and 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` = { $\langle \backslash\alpha^* | \backslash\Alpha^* | \backslash\arabic^* | \backslash\roman^* | \backslash\Roman^* \rangle$ } default: *by levels*

Sets the $\langle label \rangle$ that will be printed at the *current level* and default value for `labelwidth` key. The default value for the first level of the environments `enumext` and `enumext*` are `\arabic*`, for second level are $\langle \backslash\alpha^* \rangle$, for third level are `\roman*`, and for fourth level are `\Alpha*`. For `keyans` and `keyans*` environments the default value is `\Alpha*`.

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

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

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

`labelwidth` = { $\langle rigid length \rangle$ } default: *by label*

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

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

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

`font` = { $\langle font commands \rangle$ } default: *empty*

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Keys for 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`.

5.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

`noitemsep` *\value forbidden* default: *not used*

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

`nosep` *\value forbidden* default: *not used*

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

`base-fix` *\value forbidden* default: *not used*

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

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

- The following $\langle keys \rangle$ 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 ‘*’ $\langle keys \rangle$ applies `\vspace*` so that \LaTeX does *not discard* this space at page break.

`above = { $\langle rubber length \mid rigid length \rangle$ }` 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 “discordable”.

`above* = { $\langle rubber length \mid rigid length \rangle$ }` 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 discordable”.

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

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

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

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

5.2.2 Horizontal spaces

`list-offset = { $\langle rigid length \rangle$ }` 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 = { $\langle rigid length \rangle$ }` 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 $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”.

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

`itemindent = { $\langle rigid length \rangle$ }` 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 = { $\langle rigid length \rangle$ }` 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 = { $\langle rigid length \rangle$ }` 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.

5.3 Keys for add code

The following $\langle keys \rangle$ should be used with “caution”, they are intended to inject $\{ \langle code \rangle \}$ 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 \LaTeX which is defined (simplified) as plain form `\list{ $\langle arg one \rangle$ }{ $\langle arg two \rangle$ }`. Using the `before*` key does not allow access to the `list` parameters defined by $[\langle key = val \rangle]$.

`before = { $\langle code \rangle$ }` default: *not used*

Execute $\{ \langle code \rangle \}$ “before” the environment starts. The $\{ \langle code \rangle \}$ must be passed between braces, is executed “after” performing all calculations related to the *list parameters* in the environment and the parameters sets by $[\langle key = val \rangle]$ that is, in the second argument of the list after setting all the parameters `\begin{list}{ $\langle arg one \rangle$ }{ $\langle arg two \rangle$ }{ $\langle code \rangle$ }`.

`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 [`key = val`] sets in the environment that is, before the arguments defining the 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` environment. 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.

5.4 Keys for start, series and resume

`start = {⟨integer | integer expression⟩}` default: `1`
 Sets the *start value* of the numbering on the current level. The {⟨integer expression⟩} must be passed between braces, internally is evaluated and pass to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start={\dimeval{100*⟨value⟩{chapter}}}` or `start={100*⟨value⟩{chapter}}`.

`start* = {⟨integer | string⟩}` default: *not used*
 Sets the *start value* of the numbering on the current level. Internally ⟨string⟩ is converted and passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

The following ⟨keys⟩ are “only” available for the `enumext*` environment and the “first level” of the `enumext` environment and are ignored if set when nested within each other.

`series = {⟨series name⟩}` default: *not used*
 Stores the *keys* of the *optional argument* of the “first level” of the environment in which it is executed in {⟨series name⟩} which is used as an argument in the key `resume`. The ⟨keys⟩ stored in {⟨series name⟩} are not cumulative and are overwritten if the same {⟨series name⟩} is used again.

`resume = {⟨series name⟩}` default: *not used*
 Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` key was executed. If passed *without value* this will only set *start value* continue the numbering from the last environment in which `series={⟨series name⟩}` or `resume={⟨series name⟩}` is not present and if the `save-ans` key is active 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.

`resume* ⟨value forbidden⟩` default: *not used*
 Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` or `resume={⟨series name⟩}` keys are NOT present, if the `save-ans` key is active 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.

- 🔴 For security reasons the `series` key will never save in {⟨series name⟩} the keys `series`, `resume`, `resume*`, `save-ans`, `save-key`, `start*` and `start`. When using the key `resume={⟨series name⟩}` it will have hierarchy in the ⟨keys⟩ that are saved in {⟨series name⟩}, in order to establish the value of a ⟨key⟩ already saved in {⟨series name⟩} it must be placed to the “right” of `resume={⟨series name⟩}`, the same thing happens with the `resume*` key, the exception is the `save-ans` key that must be placed on the “left” if you want to start the numbering with its value. 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*.

5.5 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`.

5.6 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`.

5.6.1 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 \TeX justification is maintained in the `minipage` on the “right side”.

5.6.2 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={\langle store name \rangle}]
  \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 = {⟨rigid length⟩}` default: `labelsep`
 Sets the *horizontal space* between the box containing the “symbol” defined by `mark-ans` key and the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` command and the *body* in `anskey*` environment.

`wrap-ans = {⟨code {#1} more code⟩}` default: `\fbox+\parbox{#1}`
 Wraps the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` and the ⟨*body*⟩ in `anskey*` environment referenced by {#1} when using the `show-ans` or `show-pos` keys. The {⟨*code*⟩} must be passed between braces and only affects how the *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* $\langle content \rangle$ passed to `\anskey` when “storing” in the *sequence* $\{\langle store name \rangle\}$ has the form `\item $\langle content \rangle$` , the following *keys* allow modifying the way in which it is “stored” in the *sequence*.

`break-col` $\langle value forbidden \rangle$ default: *not used*

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

`item-join` = $\{\langle columns \rangle\}$ default: *not set*

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

`item-star` $\langle value forbidden \rangle$ default: *not used*

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

`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*[$\langle symbol \rangle$] $\langle content \rangle$` . The *symbol* can be in text or math mode, for example `item-sym*={ $\$ast\$$ }` stores `\item*[$\$ast\$$] $\langle content \rangle$` .

`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*[$\langle symbol \rangle$][$\langle offset \rangle$] $\langle content \rangle$` .

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.
* <input type="text" value="first answer"/>
2. Text containing our instructions or questions.
(a) Question.
* <input type="text" value="second answer"/> | 3. Text containing our instructions or questions.
* <input type="text" value="third answer"/>
4. Text containing our instructions or questions.
* <input type="text" value="fourth answer"/> |
|--|---|

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 `scontents` package, `\begin{anskey*}` and `\end{anskey*}` must be in different lines, all *keys* must be passed separated by commas and “without separation” of the start of the environment.

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

6.3.1 Keys for `anskey*`

The `anskey*` environment uses the same *keys* as the `\anskey` command next to the $\langle keys \rangle$ inherited from package `scontents`. The environment is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

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

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}`%.

- For security reasons the keys `store-env`, `print-env` and `write-out` they have been left disabled. It is recommended that you review the `scontents`[4] documentation to understand how the keys described here work.

Example

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

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

  \item Text containing our instructions or questions.

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

  \item Text containing our instructions or questions.

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

  \item Text containing our instructions or questions.

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

- | | |
|---|---|
| ★ 5. Text containing our instructions or questions. | 7. Text containing our instructions or questions. |
| [5] First answer with verbatim | [7] third answer |
| 6. Text containing our instructions or questions. | 8. Text containing our instructions or questions. |
| (a) Question. | [8] fourth answer |
| [6] second answer | |

6.4 The environments `keyans` and `keyans*`

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

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

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*` [`<content>`]

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

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


Example

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

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

  \item Text containing a question and image.

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

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

6.5 The environment `keyanspic`

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

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

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

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

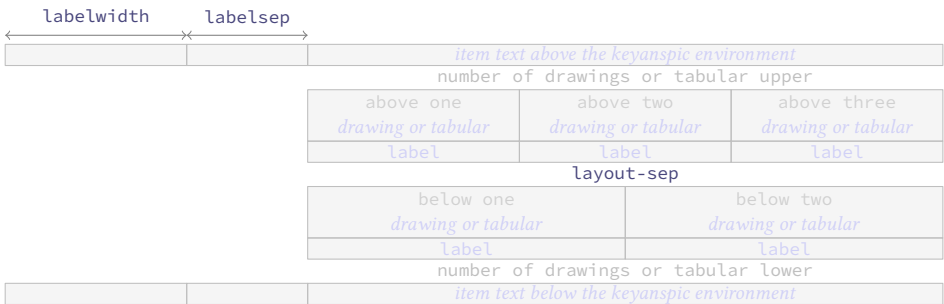


Figure 6: Representation of the `keyanspic` environment with `layout-sty={3, 2}` in `enumext`.

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

6.5.1 Keys for keyanspic

`label-pos = {⟨above | below⟩}` default: *below*

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

`label-sep = {⟨rubber length | rigid length⟩}` default: *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` `\anspic{⟨drawing or tabular⟩}`
`\anspic*` `\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.



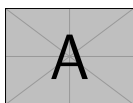
A)



B)



C)

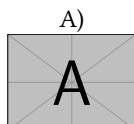


D)

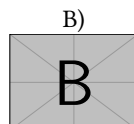


* E) [note]

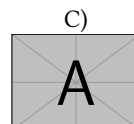
2. Question with images and labels above.



A)



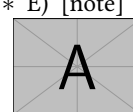
B)



C)



D)



* E) [note]

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



A)



B)



C)



D)



* E) [note]

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

6.6 Printing stored content

6.6.1 The command `\getkeyans`

```
\getkeyans <⟨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 [⟨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 {⟨code⟩} that will be executed ⟨before⟩ each {⟨content⟩} stored in *prop list* {⟨store name⟩}. The {⟨code⟩} must be passed between braces.

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

`wrapper = {⟨code #1⟩ more code}` default: *empty*
 Wraps the {⟨content⟩} stored in *prop list* {⟨store name⟩} referenced by {#1}. The {⟨code⟩} must be passed between braces. For example `\foreachkeyans[wrapper={\makebox[1em][l]{#1}}]{⟨store name⟩}`.

6.6.3 The command `\printkeyans`

```
\printkeyans {⟨store name⟩}
\printkeyans [⟨keys⟩]{⟨store name⟩}
\printkeyans * [⟨keys⟩]{⟨store name⟩}
```

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

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

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

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

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

- If we execute `\printkeyans*{⟨store name⟩}` and the *sequence* {⟨store name⟩} already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeyans*{⟨store name⟩}` and the *sequence* {⟨store name⟩} contains any `enumext` environments, they will start with the ⟨keys⟩ set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.
- If we execute `\printkeyans{⟨store name⟩}` and the *sequence* {⟨store name⟩} contains any environment `enumext*`, they will start with the ⟨keys⟩ set by default unless they are set in the *optional argument* or `save-key` is used to modify it.

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

If we need to set the ⟨keys⟩ for the environment `enumext` “saved” in the *sequence* {⟨store name⟩} we will use `\setenumext[⟨print , level⟩]{⟨keys⟩}` and if we need to set the ⟨keys⟩ for the environment `enumext*` “saved” in the *sequence* {⟨store name⟩} we will use `\setenumext[⟨print , *⟩]{⟨keys⟩}`.

Example

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

  \begin{enumext}[nosep]
    \item \LaTeX2e\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1]

2. True False

(a) ~~TeX~~ 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:

```
$ pdfdetach -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

Test keys

1. B, $x = 5$

✖ 4. E, A duck

✖

2. D


✖ 5. D, other note


✖


3. C, some note

✖


Example 4

A “simple worksheet” using ducks :) .

 Factor $x^2 - 2x + 1$


 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

(a) $\alpha > \delta$

(b) \LaTeX is cool?

 Related to Linux

(a) You use linux?

©2024 by Pablo González L


21 / 162

- (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> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>	<div>3</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>
<div>2</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>	<div>4</div> <p>Which choice best describes what happens in the passage?</p> <p>A) One character argues with another character who intrudes on her home.</p> <p>B) One character receives a surprising request from another character.</p> <p>C) One character reminisces about choices she has made over the years.</p> <p>D) One character criticizes another character for pursuing an unexpected course of action.</p>







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







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, scontents, unicode-math, geometry, graphicx, luamml and hyperref are required along with the line:

```
\DocumentMetadata
{
  lang = en-US, pdfversion = 2.0, pdfstandard = ua-2,
  testphase = {phase-III, math, title, table, firstaid},
}
```

- 🔗 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 <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. If we add <code>align=right</code> it will look like this:

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[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>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2024.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ $\text{\LaTeX}_{2\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The \LaTeX_3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The $\text{\LaTeX}_{2\epsilon}$ sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.
- [13] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2024.
- [14] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [15] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [16] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.
- [17] FISCHER, ULRICH. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2024.
- [18] The \LaTeX Project. “latex-lab – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2024.
- [19] MITTELBACH, FRANK. “ \LaTeX ’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.

11 Change history

- v1.0 (ctan), 2024-11-10**

- 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	10	item-pos*	14
Commands provide by enumext:		item-star	14
\anskey	11-14	item-sym*	14
\anspic	11-13, 16, 17	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	18	after	19
\getkeyans	13, 18	before	19
\item*	5-7, 11-13, 15, 16	sep	18
\item	5-7, 10, 11, 13, 15, 17	start	18
\miniright	11	step	18
\printkeyans	6, 12, 19	stop	18
\setenumextmeta	6	wrapper	19
\setenumext	5-7, 11, 13, 15, 19	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	14
enumXiii	4	force-eol	15
enumXii	4	item-join	14
enumXiv	4	item-pos*	14
enumXi	4	item-star	14
enumXviii	4	item-sym*	14
enumXvii	4	overwrite	14
enumXvi	4	write-env	14
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8, 9
anskey*	11-14, 23	after	10
enumext*	4-16, 19, 22, 23	align	7, 13, 22-24
enumext	4-17, 19, 22, 23	base-fix	8
keyans*	4-15, 23	before*	9, 10
keyanspic	4, 7, 8, 11-14, 16, 22, 23	before	9
keyans	4-17, 23	below*	9
Environments:		below	9
Verbatim	15	check-ans	12, 13
center	5	columns-sep	4, 10, 22
description	5, 23	columns	4, 9, 10, 22
enumerate	1, 3, 5, 24	first	10
figure	5	font	7, 12, 13
flushleft	5	item-pos*	5, 6
flushright	5	item-sym*	5, 6
itemize	5, 23	itemindent	8-10
list	3, 5, 9, 24	itemsep	8
minipage	3-5, 8-11, 22, 24	label-pos	17
multicols	3, 4, 10, 22	label-sep	17
quotation	5	labelsep	3-7, 9, 10, 22-24
quote	5	labelwidth	3, 4, 6, 7, 9, 10, 12, 13, 22-24
shortenenumerate	5	labelwith	5
tabbing	5	label	7, 8, 10, 15, 16, 22-24
table	5	labewdith	9
tasks	5	layout-sep	17
trivlist	5	layout-sty	16, 17
verbatim	5	layout-top	17
verse	5	list-indent	3, 9
		list-offset	3, 8, 9, 24

listparindent 9, 10

mark-ans* 12, 13, 16, 17

mark-ans 12, 13

mark-pos* 13, 16, 17

mark-pos 12

mark-ref 12

mark-sep* 13, 16, 17

mark-sep 12, 13

mini-env 4, 9, 11, 22

mini-right* 7, 11

mini-right 7, 11, 22

mini-sep 4, 11

mode-box 7

no-store 11–14, 23

noitemsep 8

nosep 8, 23

overwrite 14

parsep 8, 10, 17

partopsep 8

ref 4, 8, 22

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–19, 22, 23

save-key 10–12, 19

save-ref 4, 7, 12–14, 18, 22

save-sep 12, 16, 17, 22

series 7, 10, 11

show-ans 12, 13, 16, 17, 23

show-length 8

show-pos 12, 13, 16–18

start* 10

start 10

topsep 8, 9, 17

widest 7

wrap-ans* 13, 16, 17

wrap-ans 12, 13

wrap-label* 7, 24

wrap-label 7, 12, 13, 22, 24

wrap-opt 12, 13, 16, 17

write-env 14

L

\label 4

Labels provide by enumext:

 \Alph* 7, 8, 15

 \Roman* 7, 8

 \alph* 7, 8

 \arabic* 7, 8

 \roman* 7, 8

\labelsep 3, 7

\labelwidth 3, 7

\linewidth 11

\listparindent 9

P

Packages:

 enumerate 24

 enumext 1–5, 7, 12, 16, 22–25

 enumitem 3, 4, 24

 fancyvrb 15

 footnotehyper 5

 geometry 22

 graphicx 22

 hyperref 4, 5, 12–14, 22, 24

 l3keys 7

 l3prop 24

 l3seq 24

 luamml 22

 multicol 1, 2, 4, 22, 24

 scontents 1, 2, 14, 15, 22

 shortlst 5

 tasks 5

 task 6

 unicode-math 22

 xsim 2

\parsep 8

\partopsep 8

R

\raggedcolumns 4

\ref 4

\rightmargin 9

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}[2024-11-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage {enumext} {2024-11-10} {1.0} {Enumerate exercise sheets}
```

Finally check if the `multicol` and `scontents` packages 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}[2024-05-23]
14  }
15  \IfPackageLoadedTF { scontents }
16  {
17    \msg_info:nnn { enumext } { package-load } { scontents }
18  }
19  {
20    \msg_info:nnn { enumext } { package-not-load } { scontents }
21    \RequirePackage{scontents}
22  }
23 }
```

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.

```
\__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
```

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

```
24 \int_new:N \__enumext_level_int
25 \int_new:N \__enumext_level_h_int
26 \int_new:N \__enumext_anskey_level_int
27 \int_new:N \__enumext_keyans_level_int
28 \int_new:N \__enumext_keyans_level_h_int
29 \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_anskey_env_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).

```

30 \bool_new:N \l__enumext_starred_bool
31 \bool_new:N \g__enumext_starred_bool
32 \bool_new:N \l__enumext_starred_first_bool
33 \bool_new:N \l__enumext_standar_bool
34 \bool_new:N \g__enumext_standar_bool
35 \bool_new:N \l__enumext_standar_first_bool
36 \bool_new:N \l__enumext_anskey_env_bool
37 \bool_new:N \l__enumext_keyans_env_bool
38 \tl_new:N \g__enumext_start_line_tl
39 \tl_new:N \g__enumext_envir_name_tl
40 \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_counters:Nn` (§13.11) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.14).

```

41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_counter_#1_tl }
44 }
45 \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.)

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
  \l__enumext_renew_the_count_X_tl

```

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

```

46 \tl_const:Nn \c__enumext_counter_style_tl
47 { { { arabic } { roman } { Roman } { alph } { Alph } } }
48 \tl_new:N \l__enumext_ref_key_arg_tl
49 \tl_new:N \l__enumext_ref_the_count_tl
50 \cs_set_protected:Npn \__enumext_tmp:n #1
51 {
52   \tl_new:c { l__enumext_renew_the_count_#1_tl }
53   \tl_new:c { l__enumext_the_counter_#1_tl }
54   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
55 }
56 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
  \l_enumext_resume_active_bool
  \g__enumext_starred_series_tl
  \g__enumext_standar_series_tl

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.25).

```

57 \int_new:N \g__enumext_resume_int
58 \int_new:N \g__enumext_resume_vii_int
59 \tl_new:N \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N \g__enumext_standar_series_tl
62 \tl_new:N \g__enumext_starred_series_tl

```

(End of definition for `\g__enumext_resume_int` 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.

```

63 \dim_new:N \l__enumext_current_widest_dim
64 \tl_new:N \g__enumext_counter_styles_tl
65 \tl_new:N \g__enumext_widest_label_tl
66 \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 (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.38.1).

```

67 \cs_set_protected:Npn \__enumext_tmp:n #1
68 {
69   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
70   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
71   \dim_new:c { \l__enumext_leftmargin_#1_dim }
72   \dim_new:c { \l__enumext_itemindent_#1_dim }
73 }
74 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

```

Internal variables used by `columns` key (§13.22) and `align` key (§13.13).

```

75 \cs_set_protected:Npn \__enumext_tmp:n #1
76 {
77   \skip_new:c { \l__enumext_multicols_above_#1_skip }
78   \skip_new:c { \l__enumext_multicols_below_#1_skip }
79   \skip_new:c { \g__enumext_multicols_right_#1_skip }
80   \str_new:c { \l__enumext_align_label_pos_#1_str }
81 }
82 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

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

```

Internal variables used by `\miniright` command (§13.23.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.21, §13.23).

```

83 \int_new:N \g__enumext_minipage_stat_int
84 \skip_new:N \l__enumext_minipage_temp_skip
85 \skip_new:N \l__enumext_minipage_left_skip
86 \skip_new:N \l__enumext_minipage_right_skip
87 \skip_new:N \l__enumext_minipage_after_skip
88 \skip_new:N \g__enumext_minipage_right_skip
89 \skip_new:N \g__enumext_minipage_after_skip
90 \cs_set_protected:Npn \__enumext_tmp:n #1
91 {
92   \dim_new:c { \l__enumext_minipage_left_#1_dim }
93   \bool_new:c { \l__enumext_minipage_active_#1_bool }
94 }
95 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_wrap_label_X_bool
\l__enumext_wrap_label_opt_X_bool
\l__enumext_start_X_int
\l__enumext_fake_item_indent_X_tl
\l__enumext_label_fill_left_X_tl
\l__enumext_label_fill_right_X_tl
\l__enumext_vspace_a_star_X_bool
\l__enumext_vspace_b_star_X_bool

```

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.18.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.20).

```

96 \cs_set_protected:Npn \__enumext_tmp:n #1
97 {
98   \bool_new:c { \l__enumext_wrap_label_#1_bool }
99   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
100   \int_new:c { \l__enumext_start_#1_int }
101   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
103   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
104   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
105   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
106 }
107 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_store_active_bool
\l__enumext_store_name_tl
\g__enumext_store_name_tl
\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_store_anskey_opt_tl
\l__enumext_store_current_label_tl
\l__enumext_store_current_opt_arg_tl
\l__enumext_store_current_label_tmp_tl

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.26.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.30) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the $\langle body \rangle$ and the $\langle keys \rangle$ of the environment `anskey*` (§13.31).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.37) and `\anspic*` (§13.42.2) for the `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_current_label_tmp_tl` is a temporary variable used by `keyans`, `keyans*` and `keyanspic` at various points.

```

108 \bool_new:N \l__enumext_store_active_bool
109 \tl_new:N \l__enumext_store_name_tl
110 \tl_new:N \g__enumext_store_name_tl
111 \tl_new:N \l__enumext_store_anskey_arg_tl
112 \tl_new:N \l__enumext_store_anskey_env_tl
113 \tl_new:N \l__enumext_store_anskey_opt_tl
114 \tl_new:N \l__enumext_store_current_label_tl
115 \tl_new:N \l__enumext_store_current_opt_arg_tl
116 \tl_new:N \l__enumext_store_current_label_tmp_tl

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§13.48).

```

117 \tl_new:N \l__enumext_setkey_tmpa_tl
118 \tl_new:N \l__enumext_setkey_tmpb_tl
119 \int_new:N \l__enumext_setkey_tmpa_int
120 \seq_new:N \l__enumext_setkey_tmpa_seq
121 \seq_new:N \l__enumext_setkey_tmpb_seq

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

```

122 \tl_new:N \l__enumext_meta_path_tl
123 \seq_new:N \l__enumext_foreach_print_seq
124 \tl_new:N \l__enumext_foreach_name_prop_tl
125 \tl_new:N \g__enumext_foreach_default_keys_tl

```

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

```

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

```

Internal variables used by command `\printkeyans` (§13.47), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.3) and “*storing structure*”.

```

126 \tl_new:N \l__enumext_print_keyans_starred_tl
127 \bool_new:N \l__enumext_print_keyans_star_bool
128 \str_new:N \l__enumext_mark_position_str
129 \str_new:N \l__enumext_mark_position_v_str
130 \str_new:N \l__enumext_mark_position_viii_str
131 \dim_new:N \l__enumext_mark_sep_tmpa_dim
132 \dim_new:N \l__enumext_mark_sep_tmpb_dim
133 \int_new:N \l__enumext_show_pos_tmp_int
134 \tl_new:N \g__enumext_item_symbol_aux_tl
135 \cs_set_protected:Npn \l__enumext_tmp:n #1
136 {
137   \tl_new:c { \l__enumext_print_keyans_#1_tl }
138   \tl_new:c { \l__enumext_store_save_key_#1_tl }
139   \bool_new:c { \l__enumext_store_save_key_#1_bool }
140   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
141 }
142 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__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 (§13.42.1).

```
143 \seq_new:N \l__enumext_anspic_args_seq
144 \dim_new:N \l__enumext_anspic_mini_width_dim
145 \int_new:N \l__enumext_anspic_above_int
146 \int_new:N \l__enumext_anspic_below_int
147 \bool_new:N \l__enumext_anspic_label_above_bool
148 \str_new:N \l__enumext_anspic_mini_pos_str
149 \box_new:N \l__enumext_anspic_label_box
150 \box_new:N \l__enumext_anspic_body_box
151 \dim_new:N \l__enumext_anspic_label_htdp_dim
152 \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 (§13.26.3) 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.

```
153 \bool_new:N \l__enumext_check_answers_bool
154 \bool_new:N \g__enumext_check_ans_key_bool
155 \tl_new:N \l__enumext_check_start_line_env_tl
156 \bool_new:N \l__enumext_item_wrap_key_bool
157 \int_new:N \g__enumext_check_starred_cmd_int
158 \int_new:N \g__enumext_item_anskey_int
159 \int_new:N \g__enumext_item_number_int
160 \bool_new:N \l__enumext_item_number_bool
161 \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**.

```
162 \bool_new:N \l__enumext_hyperref_bool
163 \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 (§13.27). 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.

```
164 \tl_new:N \l__enumext_newlabel_arg_one_tl
165 \tl_new:N \l__enumext_newlabel_arg_two_tl
166 \tl_new:N \l__enumext_write_aux_file_tl
167 \cs_set_protected:Npn \__enumext_tmp:n #1
168 {
169   \tl_new:c { \l__enumext_label_copy_#1_tl }
170 }
171 \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).

```
172 \int_new:N \g__enumext_footnote_standar_int
173 \int_new:N \g__enumext_footnote_starred_int
174 \seq_new:N \g__enumext_footnote_standar_arg_seq
175 \seq_new:N \g__enumext_footnote_starred_arg_seq
176 \seq_new:N \g__enumext_footnote_standar_int_seq
177 \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.

```
178 \cs_set_protected:Npn \__enumext_tmp:n #1
179 {
180   \bool_new:c { \l__enumext_item_starred_#1_bool }
181   \int_new:c { \l__enumext_item_column_pos_#1_int }
182   \int_new:c { \g__enumext_item_count_all_#1_int }
183   \int_new:c { \l__enumext_joined_item_#1_int }
```

```

184 \int_new:c { \__enumext_joined_item_aux_#1_int }
185 \int_new:c { \__enumext_tmpa_#1_int }
186 \dim_new:c { \__enumext_tmpa_#1_dim }
187 \box_new:c { \__enumext_item_text_#1_box }
188 \dim_new:c { \__enumext_joined_width_#1_dim }
189 \dim_new:c { \__enumext_item_width_#1_dim }
190 \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
191 \str_new:c { \__enumext_align_label_#1_str }
192 \bool_new:c { g__enumext_minipage_active_#1_bool }
193 \box_new:c { \__enumext_miniright_code_#1_box }
194 \bool_new:c { g__enumext_minipage_center_#1_bool }
195 \dim_new:c { g__enumext_minipage_right_#1_dim }
196 \skip_new:c { g__enumext_minipage_right_#1_skip }
197 }
198 \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`.

```

199 \clist_const:Nn \c__enumext_all_envs_clist
200 {
201   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
202   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
203 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

`\seq_use:NV`

```

204 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
205 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

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

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

```

206 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
207 {
208   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
209 }

```

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

`__enumext_after_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.

`__enumext_before_env:nn`

```

210 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
211 {
212   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
213 }
214 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
215 {
216   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
217 }

```

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

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

```

218 \cs_new:Nn \__enumext_level:
219 {
220   \int_to_roman:n { \__enumext_level_int }
221 }

```

(End of definition for `__enumext_level:`.)


```

\__enumext_if_is_int:nT
\__enumext_if_is_int:nF
\__enumext_if_is_int:nTF

```

A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function argument is an integer expression?](#).

```

222 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
223 {
224   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
225   { \prg_return_true: }
226   { \prg_return_false: }
227 }

```

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

```
\__enumext_regex_counter_style:
```

The internal function `__enumext_regex_counter_style:` replace the ‘*’ with the actual counter of the running level and is used by the `ref` key. It loops through the defined counter styles in `\c__enumext_counter_style_tl` and replace ‘*’ by real command, for example, looking for `\arabic*` and replacing that by `\arabic{<counter>}` defined on the current level.

```

228 \cs_new_protected:Nn \__enumext_regex_counter_style:
229 {
230   \tl_map_inline:Nn \c__enumext_counter_style_tl
231   {
232     \regex_replace_once:nnN { \c{##1}\* }
233     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
234   }
235 }

```

(End of definition for `__enumext_regex_counter_style:.`)

```
\__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.

```

236 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
237 {
238   * ~ #2
239   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
240   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
241 }

```

(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.

```

242 \cs_new_protected:Nn \__enumext_unskip_unkern:
243 {
244   \int_case:nnT { \lastnodetype }
245   {
246     { 11 }{ \unskip }
247     { 12 }{ \unkern }
248   }
249 }

```

(End of definition for `__enumext_unskip_unkern:.`)

13.5.1 Utilities for environments and levels

```

\__enumext_is_not_nested:
\__enumext_is_on_first_level:

```

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`.

```

250 \cs_new_protected:Nn \__enumext_is_not_nested:
251 {
252   \str_case:en { \@currenvir }
253   {
254     {enumext}
255     {
256       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
257       \bool_lazy_and:nnT
258       { \bool_not_p:n { \g__enumext_standar_bool } }
259       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
260       {
261         \bool_gset_true:N \g__enumext_standar_bool
262       }
263     }
264     {enumext*}

```

```

265     {
266       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
267       \bool_lazy_and:nnT
268         { \bool_not_p:n { \g__enumext_starred_bool } }
269         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
270       {
271         \bool_gset_true:N \g__enumext_starred_bool
272       }
273     }
274   }
275 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) 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.

```

276 \cs_new_protected:Nn \__enumext_is_on_first_level:
277 {
278   \bool_lazy_all:nT
279   {
280     { \bool_if_p:N \g__enumext_standar_bool }
281     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
282     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
283   }
284   {
285     \bool_set_true:N \l__enumext_standar_first_bool
286     \bool_set_true:N \l__enumext_anskey_env_bool
287     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
288     \tl_gset:Nn \g__enumext_start_line_tl
289       {
290         on ~ line ~ \exp_not:V \inputlineno
291       }
292   }
293   \bool_lazy_all:nT
294   {
295     { \bool_if_p:N \g__enumext_starred_bool }
296     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
297     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
298   }
299   {
300     \bool_set_true:N \l__enumext_starred_first_bool
301     \bool_set_true:N \l__enumext_anskey_env_bool
302     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
303     \tl_gset:Nn \g__enumext_start_line_tl
304       {
305         on ~ line ~ \exp_not:V \inputlineno
306       }
307   }
308 }

```

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

`__enumext_keyans_name_and_start:`

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

```

309 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
310 {
311   \str_case:en { \@currenvir }
312   {
313     {keyans}
314     {
315       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
316       \tl_set:Nn \l__enumext_check_start_line_env_tl
317         {
318           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
319         }
320     }
321     {keyans*}
322     {

```

```

323         \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
324         \tl_set:Ne \l__enumext_check_start_line_env_tl
325         {
326             in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
327         }
328     }
329     {keyanspic}
330     {
331         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
332         \tl_set:Ne \l__enumext_check_start_line_env_tl
333         {
334             in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
335         }
336     }
337 }
338 }

```

(End of definition for `__enumext_keyans_name_and_start:`)

13.5.2 Utilities for log and terminal

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

```

\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
339 \cs_new_protected:Nn \__enumext_reset_global_vars:
340 {
341     \__enumext_reset_global_int:
342     \__enumext_reset_global_bool:
343     \__enumext_reset_global_tl:
344 }
345 \cs_new_protected:Nn \__enumext_reset_global_int:
346 {
347     \int_gzero:N \g__enumext_item_number_int
348     \int_gzero:N \g__enumext_item_anskey_int
349     \int_gzero:N \g__enumext_item_answer_diff_int
350 }
351 \cs_new_protected:Nn \__enumext_reset_global_bool:
352 {
353     \bool_gset_false:N \g__enumext_check_ans_key_bool
354     \bool_gset_false:N \g__enumext_standar_bool
355     \bool_gset_false:N \g__enumext_starred_bool
356 }
357 \cs_new_protected:Nn \__enumext_reset_global_tl:
358 {
359     \tl_gclear:N \g__enumext_store_name_tl
360     \tl_gclear:N \g__enumext_start_line_tl
361     \tl_gclear:N \g__enumext_envir_name_tl
362 }

```

(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.

```

363 \cs_new_protected:Nn \__enumext_log_global_vars:
364 {
365     \msg_log:nneeee { enumext } { prop-seq-int-hook }
366     { \g__enumext_store_name_tl }
367     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
368     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
369     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
370 }

```

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.

```

371 \cs_new_protected:Nn \__enumext_log_answer_vars:
372 {
373     \msg_log:nneeee { enumext } { item-answer-hook }
374     { \int_use:N \g__enumext_item_number_int }
375     { \int_use:N \g__enumext_item_anskey_int }
376     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
377 }

```

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

13.6 Copying list and minipage environments

The `list` environment provided by \LaTeX has the following plain form:

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

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

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

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

🔍 For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `ltxcmd` (see `latex-lab-block`[18]).

```
\__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.

```
378 \__enumext_at_begin_document:n
379 {
380   \cs_new_eq:NN \__enumext_start_list:nn \list
381   \cs_new_eq:NN \__enumext_stop_list: \endlist
382   \NewCommandCopy \__enumext_item_std:w \item
383 }
```

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

```
384 \__enumext_at_begin_document:n
385 {
386   \cs_new_eq:NN \__enumext_minipage:w \minipage
387   \cs_new_eq:NN \__enumext_endminipage: \endminipage
388 }
```

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

13.7 Compatibility with hyperref and footnotehyper

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

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

```
389 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
390 \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”.

```
391 \cs_new_protected:Nn \__enumext_after_hyperref:
392 {
393   \IfPackageLoadedTF { hyperref }
394   {
395     \msg_info:nnn { enumext } { package-load } { hyperref }
396     \bool_set_true:N \l__enumext_hyperref_bool
397     \IfHyperBoolean{hyperfootnotes}
398     {
399       \bool_set_true:N \l__enumext_footnotes_key_bool
400     }
401     { }
402   }
403 }
```

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`.

```
404 \bool_if:NT \l__enumext_footnotes_key_bool
405 {
406   \IfPackageLoadedTF { footnotehyper }
407   {
408     \msg_info:nnn { enumext } { package-load } { footnotehyper }
409   }
410 }
```

```

411         \bool_set_false:N \l__enumext_footnotes_key_bool
412     }
413 }

```

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.

```

414 \bool_if:NTF \l__enumext_hyperref_bool
415 {
416     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
417     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
418 }
419 {
420     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
421     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
422 }
423 }

```

(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.

```

424 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
425 {
426     \protected@write \@auxout { }
427     {
428         \token_to_str:N \newlabel {#1}
429         {
430             {#2}
431             \bool_if:NT \l__enumext_hyperref_bool
432             { { \thepage } {#2} {#1} }
433             { }
434         }
435     }
436     \__enumext_hypertarget:nn {#1} { }
437     \__enumext_phantomsection:
438 }

```

(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.

```

439 \cs_new_protected:Nn \__enumext_footnotetext:nn
440 {
441     \footnotetext[#1]{#2}
442 }
443 \cs_new_protected:Nn \__enumext_renew_footnote:
444 {
445     \RenewDocumentCommand \footnote { o +m }
446     {
447         \tl_if_novalue:nTF {##1}
448         {
449             \stepcounter{footnote}
450             \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
451         }
452         {
453             \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
454         }
455         \footnotemark [ \g__enumext_footnote_standar_int ]
456         \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
457         \seq_gput_right:NV

```

```

458         \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
459     }
460 }
461 \cs_new_protected:Nn \__enumext_print_footnote:
462 {
463     \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
464     {
465         \seq_map_pairwise_function:NNN
466         \g__enumext_footnote_standar_int_seq
467         \g__enumext_footnote_standar_arg_seq
468         \__enumext_footnotetext:nn
469     }
470     \seq_gclear:N \g__enumext_footnote_standar_arg_seq
471     \seq_gclear:N \g__enumext_footnote_standar_int_seq
472 }

```

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.

```

473 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
474 {
475     \RenewDocumentCommand \footnote { o +m }
476     {
477         \tl_if_novalue:nTF {##1}
478         {
479             \stepcounter{footnote}
480             \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
481         }
482         {
483             \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
484         }
485         \footnotemark [ \g__enumext_footnote_starred_int ]
486         \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
487         \seq_gput_right:NV
488         \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
489     }
490 }
491 \cs_new_protected:Nn \__enumext_print_footnote_mini:
492 {
493     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
494     {
495         \seq_map_pairwise_function:NNN
496         \g__enumext_footnote_starred_int_seq
497         \g__enumext_footnote_starred_arg_seq
498         \__enumext_footnotetext:nn
499     }
500     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
501     \seq_gclear:N \g__enumext_footnote_starred_int_seq
502 }

```

(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.

```

503 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
504 {
505     \bool_if:NT \g__enumext_standar_bool
506     {
507         \IfDocumentMetadataTF
508         {
509             \__enumext_renew_footnote:
510         }
511         {
512             \bool_if:NF \l__enumext_footnotes_key_bool
513             {
514                 \__enumext_renew_footnote:
515             }
516         }
517     }
518 }
519 \cs_new_protected:Nn \__enumext_print_footnote_standar:

```



```

520 {
521   \bool_if:NT \g__enumext_standar_bool
522   {
523     \IfDocumentMetadataTF
524     {
525       \__enumext_print_footnote:
526     }
527     {
528       \bool_if:NF \l__enumext_footnotes_key_bool
529       {
530         \__enumext_print_footnote:
531       }
532     }
533   }
534 }

```

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.

```

535 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
536 {
537   \IfDocumentMetadataTF
538   {
539     \__enumext_renew_footnote_mini:
540   }
541   {
542     \bool_if:NF \l__enumext_footnotes_key_bool
543     {
544       \__enumext_renew_footnote_mini:
545     }
546   }
547 }
548 \cs_new_protected:Nn \__enumext_print_footnote_starred:
549 {
550   \IfDocumentMetadataTF
551   {
552     \__enumext_print_footnote_mini:
553   }
554   {
555     \bool_if:NF \l__enumext_footnotes_key_bool
556     {
557       \__enumext_print_footnote_mini:
558     }
559   }
560 }

```

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

```

561 \__enumext_after_env:nn { enumext* }
562 {
563   \__enumext_print_footnote_starred:
564 }
565 \__enumext_after_env:nn { keyans* }
566 {
567   \__enumext_print_footnote_starred:
568 }

```

(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 (§13.39) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.44)

```

569 \cs_new_protected:Nn \__enumext_internal_mini_page:
570 {
571   \int_compare:nNt { \l__enumext_level_int } = { 0 }

```

```

572     {
573         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
574         {
575             \__enumext_renew_footnote_standar:
576             \__enumext_minipage:w [ t ] { ##1 }
577             \legacy_if_gset_false:n { @minipage }
578             \skip_vertical:N \c_zero_skip
579         }
580         {
581             \skip_vertical:N \c_zero_skip
582             \__enumext_endminipage:
583             \__enumext_print_footnote_standar:
584         }
585     }
586 }

```

(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.

```

587 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

```

\__enumext_define_counters:Nn
enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

```

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.

```

588 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
589 {
590     \cs_if_exist:cTF { c@ #2 }
591     { \msg_fatal:nnn { enumext } { counters }{ #2 } }
592     {
593         \tl_set:Nn #1 { #2 }
594         \newcounter { #2 }
595     }
596 }

```

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.

```

597 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
598 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
599 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
600 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
601 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
602 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
603 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
604 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

(End of definition for `__enumext_define_counters:Nn` 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.

```

\__enumext_register_counter_style:Nn

```

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

```

605 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
606 {
607     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
608     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
609 }
610 \__enumext_register_counter_style:Nn \arabic { 0 }
611 \__enumext_register_counter_style:Nn \Alph { M }
612 \__enumext_register_counter_style:Nn \alph { m }

```

```

613 \__enumext_register_counter_style:Nn \Roman { VIII }
614 \__enumext_register_counter_style:Nn \roman { viii }

```

(End of definition for __enumext_register_counter_style: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.

```

615 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
616 {
617   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
618   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
619 }
620 \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.

```

621 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
622 {
623   \tl_clear_new:N #1
624   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
625   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
626   \tl_map_inline:Nn \g__enumext_counter_styles_tl
627   {
628     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
629     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
630     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
631   }
632   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
633   { \tl_use:N \g__enumext_widest_label_tl }
634   \tl_set_eq:cN { the #2 } #1
635 }
636 \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 (§13.34). 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.

```

637 \cs_set_protected:Npn \__enumext_tmp:n #1
638 {
639   \keys_define:nn { enumext / #1 }
640   {
641     mode-box .bool_set:N = \l__enumext_mode_box_bool,
642     mode-box .initial:n = false,
643     mode-box .value_forbidden:n = true,
644   }
645 }
646 \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.

```

647 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
648 {
649   \keys_define:nn { enumext / #1 }
650   {
651     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
652     font .value_required:n = true,
653     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
654     labelsep .initial:n = {0.3333em},
655     labelsep .value_required:n = true,

```

```

656     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
657     labelwidth .value_required:n = true,
658     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
659     wrap-label .initial:n = {##1},
660     wrap-label .value_required:n = true,
661     wrap-label* .code:n = {
662         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
663         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
664     },
665     wrap-label* .value_required:n = true,
666 }
667 }
668 \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.

```

669 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
670 {
671     \keys_define:nn { enumext / #1 }
672     {
673         align .choice:,
674         align / left .code:n =
675             {
676                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
677                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
678                 \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
679             },
680         align / right .code:n =
681             {
682                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
683                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
684                 \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
685             },
686         align / center .code:n =
687             {
688                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
689                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
690                 \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
691             },
692         align / unknown .code:n =
693             \msg_error:nneee { enumext } { unknown-choice }
694             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
695         align .initial:n = left,
696         align .value_required:n = true,
697     }
698 }
699 \clist_map_inline:nn
700 {
701     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
702 }
703 { \__enumext_tmp:nn #1 }
704 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
705 {
706     \keys_define:nn { enumext / #1 }
707     {
708         align .choice:,
709         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
710         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
711         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
712         align / unknown .code:n =
713             \msg_error:nneee { enumext } { unknown-choice }
714             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
715         align .initial:n = left,
716         align .value_required:n = true,
717     }
718 }
719 \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 $\langle label \rangle$, the value of the variables $\backslash_enumext_label_X_tl$, the default values for $\backslash labelwidth$ and the “*label and ref*” system.

13.14.1 Define and set label and ref keys for enumext environment

Here we set the default $\langle labels \rangle$ of the *four levels* of `enumext` environment, along with the default value for `labelwidth` key and `ref` key.

```

label
ref
\__enumext_label_i_tl 720 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_ii_tl 721 {
\__enumext_label_iii_tl 722   \keys_define:nn { enumext / #1 }
\__enumext_label_iv_tl 723   {
724     label .code:n = {
725       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
726       { \__enumext_counter_#2_tl } {##1}
727       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
728       \__enumext_current_widest_dim
729     },
730     label .initial:n = #3,
731     label .value_required:n = true,
732     ref .code:n = \__enumext_standar_ref:n {##1},
733     ref .value_required:n = true,
734   }
735 }
736 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
737 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
738 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
739 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }
```

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

$\backslash_enumext_standar_ref:n$ The $\backslash_enumext_standar_ref:n$ first we will pass the key argument to $\backslash_enumext_ref_key_arg_tl$ and we will analyze its state, if it is not *empty* we will make a copy of the current counter in $\backslash_enumext_ref_the_count_tl$ and we will execute the function $\backslash_enumext_regex_counter_style:$ which will return the modified $\backslash_enumext_ref_key_arg_tl$ and we make the value of $\backslash_enumext_ref_the_count_tl$ the same as that $\backslash_enumext_the_counter_X_tl$ which contains $\backslash theenumX$ and finally we set $\backslash_enumext_renew_the_count_X_tl$ with the renewed command.

```

740 \cs_new_protected:Npn \__enumext_standar_ref:n #1
741 {
742   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
743   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
744   {
745     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
746   }
747   {
748     \tl_set_eq:Nc
749     \__enumext_ref_the_count_tl { \__enumext_counter_ \__enumext_level: _tl }
750     \__enumext_regex_counter_style:
751     \tl_set_eq:Nc
752     \__enumext_ref_the_count_tl { \__enumext_the_counter_ \__enumext_level: _tl }
753     \tl_put_right:ce { \__enumext_renew_the_count_ \__enumext_level: _tl }
754     {
755       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl } { \exp_not:V \l_
756     }
757   }
758 }
```

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

```

759 \cs_new_protected:Npn \__enumext_standar_ref:
760 {
761   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
762   {
763     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
764   }
765 }
```

(End of definition for $\backslash_enumext_standar_ref:n$ and $\backslash_enumext_standar_ref:$.)

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 766 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 767 {
768   \keys_define:nn { enumext / #1 }
769   {
770     label .code:n = {
771       \__enumext_label_style:cvn { l__enumext_label_#2_tl }
772       { l__enumext_counter_#2_tl } {##1}
773       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
774       \l__enumext_current_widest_dim
775     },
776     label .initial:n = #3,
777     label .value_required:n = true,
778     ref .code:n = \__enumext_starred_ref:n {##1},
779     ref .value_required:n = true,
780   }
781 }
782 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
783 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

(End of definition for label and others.)

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

```

\__enumext_starred_ref:n 784 \cs_new_protected:Npn \__enumext_starred_ref:n #1
\__enumext_starred_ref: 785 {
786   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
787   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
788   {
789     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
790     {
791       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
792     }
793     {
794       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
795       \__enumext_regex_counter_style:
796       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
797       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
798       {
799         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
800       }
801     }
802   }
803   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
804   {
805     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
806     {
807       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
808     }
809     {
810       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
811       \__enumext_regex_counter_style:
812       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
813       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
814       {
815         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
816       }
817     }
818   }
819 }

```

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

```

820 \cs_new_protected:Nn \__enumext_starred_ref:
821 {
822   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
823   {
824     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
825     {
826       \tl_use:N \l__enumext_renew_the_count_vii_tl

```



```

827     }
828   }
829   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
830   {
831     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
832     {
833       \tl_use:N \l__enumext_renew_the_count_viii_tl
834     }
835   }
836 }

```

(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.

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
837 \keys_define:nn { enumext / keyans }
838 {
839   label .code:n = {
840     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
841     { \l__enumext_counter_v_tl } {#1}
842     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
843     { \l__enumext_counter_vi_tl } {#1}
844     \dim_set_eq:NN
845     \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
846   },
847   label .initial:n = \Alph*,
848   label .value_required:n = true,
849   ref .code:n = \__enumext_keyans_ref:n {#1},
850   ref .value_required:n = true,
851 }

```

(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:
852 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
853 {
854   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
855   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
856   {
857     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
858   }
859   {
860     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
861     \__enumext_regex_counter_style:
862     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
863     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
864     {
865       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__enumext_ref_key_arg_tl }
866     }
867   }
868 }

```

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

```

869 \cs_new_protected:Nn \__enumext_keyans_ref:
870 {
871   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
872   {
873     \tl_use:N \l__enumext_renew_the_count_v_tl
874   }
875 }

```

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

13.15 Setting start, start* and widest keys

```
\__enumext_start_from:NNn
\__enumext_start_from:ccn
\__enumext_start_from:cce
```

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

```
#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3: ⟨integer or string⟩
```

The first argument of this function are the “counter style” set by `label` key, the second argument is returned by the function, the third argument can be an ⟨integer⟩ or ⟨string⟩ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```
876 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
877 {
878   \__enumext_if_is_int:nTF { #3 }
879   {
880     \int_set:Nn #2 {#3}
881   }
882   {
883     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
884     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
885     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
886     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
887   }
888 }
889 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

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

```
\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn
```

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

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

The second and third arguments of this function are the values set by `label` and `labelwidth` keys, the four argument can be an ⟨integer⟩ or ⟨string⟩ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. The value of the four argument is set temporarily for the identified counter in this point (level), then the value is expanded into a “box” and the “width” of the “box” is returned.

```
890 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
891 {
892   \__enumext_if_is_int:nTF {#4}
893   {
894     \setcounter{enumX#1} { #4 }
895   }
896   {
897     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
898     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
899     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
900     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
901   }
902   \__enumext_label_width_by_box:cv
903   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
904 }
905 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

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

```
start
start*
widest
```

```
906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907 {
908   \keys_define:nn { enumext / #1 }
909   {
910     start* .code:n = {
911       \__enumext_start_from:ccn
912       { \l__enumext_label_#2_tl }
913       { \l__enumext_start_#2_int } {##1}
914     },
915     start* .value_required:n = true,
916     start .code:n = {
917       \__enumext_start_from:cce
918       { \l__enumext_label_#2_tl }
919       { \l__enumext_start_#2_int } { \int_eval:n {##1} }
```

```

920         },
921         start .initial:n = 1,
922         start .value_required:n = true,
923         widest .code:n = {
924             \__enumext_widest_from:nccn {#2}
925             { l__enumext_label_#2_tl }
926             { l__enumext_labelwidth_#2_dim } {##1}
927         },
928         widest .value_required:n = true,
929     }
930 }
931 \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 vertical spaces

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

```

932 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
933 {
934     \keys_define:nn { enumext / #1 }
935     {
936         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
937         topsep .initial:n = {#3},
938         topsep .value_required:n = true,
939         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
940         partopsep .initial:n = {#4},
941         partopsep .value_required:n = true,
942         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
943         parsep .initial:n = {#5},
944         parsep .value_required:n = true,
945         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
946         itemsep .initial:n = {#6},
947         itemsep .value_required:n = true,
948         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
949         noitemsep .value_forbidden:n = true,
950         nosepp .meta:n = {
951             itemsep = 0pt, parsep = 0pt,
952             topsep = 0pt, partopsep = 0pt,
953         },
954         nosepp .value_forbidden:n = true,
955     }
956 }

```

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

```

957 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
958 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
959 { 4.0pt plus 2.0pt minus 1.0pt }
960 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
961 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
962 { 2.0pt plus 1.0pt minus 1.0pt }
963 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
964 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
965 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
966 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
967 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
968 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
969 { 2.0pt plus 1.0pt minus 1.0pt }
970 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
971 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
972 { 4.0pt plus 2.0pt minus 1.0pt }
973 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
974 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
975 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.17 Setting base-fix key

When nesting starting right after `\item` (without material between them) there is a problem with the alignment of the *baseline* between the two environments. One way to get around this problem is to place

`\mode_leave_vertical:` apply `\vspace{-\baselineskip}` and set `\topsep=0pt` for the “first level” of the nested `enumext` environment.

`base-fix`
`__enumext_nested_base_line_fix:`

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

```

976 \keys_define:nn { enumext / level-1 }
977 {
978   base-fix .bool_set:N = \__enumext_base_line_fix_bool,
979   base-fix .initial:n = false,
980   base-fix .value_forbidden:n = true,
981 }

```

The function `__enumext_nested_base_line_fix:` passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.39) will be responsible for applying the *baseline correction* and adjusting the `\keys` for the `enumext` environment and the `\printkeyans` with *starred argument* “*” (§13.47).

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.

```

982 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
983 {
984   \bool_lazy_all:nT
985   {
986     { \bool_if_p:N \__enumext_starred_first_bool }
987     { \bool_if_p:N \__enumext_base_line_fix_bool }
988     { \bool_not_p:n { \__enumext_print_keyans_star_bool } }
989   }
990   {
991     \mode_leave_vertical:
992     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
993     \keys_set:nn { enumext / level-1 }
994     {
995       topsep = 0pt, above = 0pt, above* = 0pt,
996     }
997   }

```

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`.

```

998   \bool_lazy_and:nnT
999   { \bool_if_p:N \__enumext_starred_first_bool }
1000   { \bool_if_p:N \__enumext_print_keyans_star_bool }
1001   {
1002     \mode_leave_vertical:
1003     \skip_vertical:n { -\baselineskip }
1004     \skip_vertical:N \c_zero_skip
1005     \keys_set:nn { enumext / level-1 }
1006     {
1007       topsep = 0pt, above = 0pt, above* = 0pt,
1008     }
1009   }
1010   \bool_set_false:N \__enumext_base_line_fix_bool
1011 }

```

(End of definition for `base-fix` and `__enumext_nested_base_line_fix:`)

13.18 Setting keys for horizontal spaces

`itemindent`
`rightmargin`
`listparindent`
`list-offset`
`list-indent`

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

```

1012 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1013 {
1014   \keys_define:nn { enumext / #1 }
1015   {
1016     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
1017     itemindent .value_required:n = true,
1018     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
1019     rightmargin .value_required:n = true,
1020     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },

```

```

1021         listparindent .value_required:n = true,
1022         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1023         list-offset   .value_required:n = true,
1024         list-indent   .code:n      =
1025                     \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1026                     \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1027         list-indent   .value_required:n = true,
1028     }
1029 }
1030 \clist_map_inline:nn
1031 {
1032     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1033 }
1034 { l__enumext_tmp:nn #1 }

```

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

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

```

1035 \cs_set_protected:Npn l__enumext_tmp:nn #1 #2
1036 {
1037     \keys_define:nn { enumext / #1 }
1038     {
1039         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1040         itemindent .value_required:n = true,
1041         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1042         rightmargin .value_required:n = true,
1043         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1044         listparindent .value_required:n = true,
1045         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1046         list-offset .value_required:n = true,
1047         list-indent .meta:n      = { list-offset = ##1 },
1048         list-indent .value_required:n = true,
1049     }
1050 }
1051 \clist_map_inline:nn
1052 {
1053     {enumext*}{vii}, {keyans*}{viii}
1054 }
1055 { l__enumext_tmp:nn #1 }

```

13.18.1 Functions for setting the fake `itemindent`

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

```

1056 \cs_set_protected:Nn l__enumext_fake_item_indent:
1057 {
1058     \dim_compare:nNnT
1059     { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
1060     >
1061     { \c_zero_dim }
1062     {
1063         \tl_set:ce { l__enumext_fake_item_indent_ l__enumext_level: _tl }
1064         {
1065             \exp_not:N \mode_leave_vertical:
1066             \exp_not:n { \skip_horizontal:n }
1067             { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
1068             \exp_not:N \ignorespaces
1069         }
1070     }
1071 }
1072 \cs_set_protected:Nn l__enumext_keyans_fake_item_indent:
1073 {
1074     \dim_compare:nNnT
1075     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1076     {
1077         \tl_set:Nc l__enumext_fake_item_indent_v_tl
1078         {
1079             \exp_not:N \mode_leave_vertical:
1080             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim

```

```

1081         \exp_not:N \ignorespaces
1082     }
1083 }
1084 }
1085 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1086 {
1087     \dim_compare:nNnT
1088     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1089     {
1090         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
1091         {
1092             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1093             \exp_not:N \ignorespaces
1094         }
1095     }
1096 }
1097 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1098 {
1099     \dim_compare:nNnT
1100     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1101     {
1102         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
1103         {
1104             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1105             \exp_not:N \ignorespaces
1106         }
1107     }
1108 }

```

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

13.19 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.

```

1109 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1110 {
1111     \keys_define:nn { enumext / #1 }
1112     {
1113         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
1114         show-length .initial:n = false,
1115     }
1116 }
1117 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20 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.

```

1118 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1119 {
1120     \keys_define:nn { enumext / #1 }
1121     {
1122         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
1123         before .value_required:n = true,
1124         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
1125         before* .value_required:n = true,
1126         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
1127         after .value_required:n = true,
1128         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
1129         first .value_required:n = true,
1130     }
1131 }
1132 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20.1 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⟩}`.

```
1133 \cs_new_protected:Nn \__enumext_before_args_exec:
1134 {
1135   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1136 }
```

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⟩}`

```
1137 \cs_new_protected:Nn \__enumext_before_keys_exec:
1138 {
1139   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1140 }
```

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

```
1141 \cs_new_protected:Nn \__enumext_after_stop_list:
1142 {
1143   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1144 }
```

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.`

```
1145 \cs_new_protected:Nn \__enumext_after_args_exec:
1146 {
1147   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1148 }
```

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

13.20.2 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:

1149 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1150 {
1151   \tl_use:N \l__enumext_before_starred_key_v_tl
1152 }
1153 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1154 {
1155   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1156 }
1157 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1158 {
1159   \tl_use:N \l__enumext_after_stop_list_v_tl
1160 }
1161 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1162 {
1163   \tl_use:N \l__enumext_after_list_args_v_tl
1164 }
```

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

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

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

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

1165 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1166 {
1167   \tl_use:N \l__enumext_before_starred_key_vii_tl
1168 }
1169 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1170 {
1171   \tl_use:N \l__enumext_before_starred_key_viii_tl
1172 }
1173 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1174 {
1175   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1176 }
1177 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1178 {
```

```

1179     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1180   }
1181   \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1182   {
1183     \tl_use:N \l__enumext_after_stop_list_vii_tl
1184   }
1185   \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1186   {
1187     \tl_use:N \l__enumext_after_stop_list_viii_tl
1188   }
1189   \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1190   {
1191     \tl_use:N \l__enumext_after_list_args_vii_tl
1192   }
1193   \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1194   {
1195     \tl_use:N \l__enumext_after_list_args_viii_tl
1196   }

```

(End of definition for __enumext_before_args_exec_vii: and others.)

13.21 Setting keys for multicols and minipage

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

```

1197 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1198 {
1199   \keys_define:nn { enumext / #1 }
1200   {
1201     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1202     mini-env .value_required:n = true,
1203     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1204     mini-sep .initial:n = 0.3333em,
1205     mini-sep .value_required:n = true,
1206     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1207     columns-sep .value_required:n = true,
1208     columns .int_set:c = { l__enumext_columns_#2_int },
1209     columns .initial:n = 1,
1210     columns .value_required:n = true,
1211   }
1212 }
1213 \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.

```

1214 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1215 {
1216   \keys_define:nn { enumext / #1 }
1217   {
1218     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1219     mini-right .value_required:n = true,
1220     mini-right* .code:n = {
1221       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1222       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1223     },
1224     mini-right* .value_required:n = true,
1225   }
1226 }
1227 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

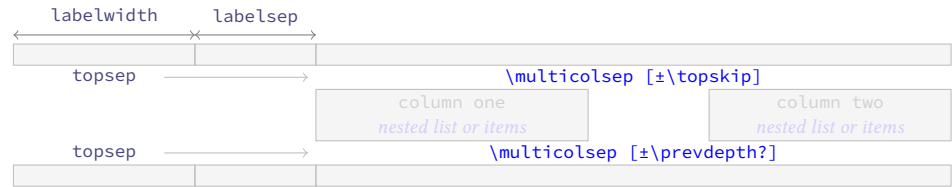
```

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

13.22 Adjustment of vertical spaces for multicols

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

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

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

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.

13.22.1 Adjustment of vertical spaces for multicols in enumext

`__enumext_multi_set_vskip:` The function `__enumext_multi_set_vskip:` will take care of determining the “*adjusted spaces*” that we will apply “*above*” and “*below*” the `multicols` environment in `enumext`.

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

```

1228 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1229 {
1230   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1231   {
1232     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1233   }
1234   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1235   {
1236     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1237   }
1238   \__enumext_add_pre_parsep:
1239 }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1240 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1241 {
1242   \int_case:nn { \l__enumext_level_int }
1243   {
1244     { 2 }{
1245       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1246       {
1247         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1248         {
1249           \l__enumext_parsep_i_skip
1250         }
1251       }
1252     }
1253     { 3 }{
1254       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1255       {
1256         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1257         {
1258           \l__enumext_parsep_ii_skip
1259         }
1260       }
1261     }
1262     { 4 }{
1263       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1264       {
1265         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1266         {
1267           \l__enumext_parsep_iii_skip
1268         }
1269       }
1270     }
1271   }

```

```

1269         }
1270     }
1271 }
1272 }

```

(End of definition for `__enumext_add_pre_parse:`)

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

```

1273 \cs_new_protected:Nn \__enumext_multi_addvspace:
1274 {
1275     \__enumext_multi_set_vskip:
1276     \mode_if_vertical:T
1277     {
1278         \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1279         {
1280             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1281         }
1282         \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1283         {
1284             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1285         }
1286     }
1287     \par\nopagebreak
1288     \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1289 }

```

(End of definition for `__enumext_multi_addvspace:`)

13.22.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1290 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1291 {
1292     \skip_set:Nn \l__enumext_multicols_above_v_skip
1293     {
1294         \l__enumext_topsep_v_skip
1295     }
1296     \skip_set:Nn \l__enumext_multicols_below_v_skip
1297     {
1298         \l__enumext_topsep_v_skip
1299     }
1300 }
1301 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1302 {
1303     \__enumext_keyans_multi_set_vskip:
1304     \mode_if_vertical:T
1305     {
1306         \skip_add:Nn \l__enumext_multicols_above_v_skip
1307         {
1308             \skip_use:N \l__enumext_partopsep_v_skip
1309         }
1310         \skip_add:Nn \l__enumext_multicols_below_v_skip
1311         {
1312             \skip_use:N \l__enumext_partopsep_v_skip
1313         }
1314     }
1315     \par\nopagebreak
1316     \addvspace{ \l__enumext_multicols_above_v_skip }
1317 }

```

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

13.23 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.

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.

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

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` is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[14] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

13.23.1 Adjustment of vertical spaces for minipage in enumext

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`.

```

1318 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1319 {
1320   \skip_set:Nn \l__enumext_minipage_right_skip
1321   {
1322     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1323   }
1324   \mode_if_vertical:T
1325   {
1326     \skip_add:Nn \l__enumext_minipage_right_skip
1327     {
1328       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1329     }
1330   }
1331   \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:`.

```

1332   \skip_set_eq:cN
1333   { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1334   \skip_set_eq:cN
1335   { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1336   \__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`.

```

1337   \int_compare:nNnT
1338   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1339   {
1340     \skip_zero:N \topskip
1341     \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1342   }
1343 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_mini_page` environment, taking into account whether \TeX is in $\langle horizontal\ mode \rangle$ or $\langle vertical\ mode \rangle$. Here we use the plain \TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1344 \cs_new_protected:Nn \__enumext_minipage_add_space:
1345 {
1346   \__enumext_minipage_set_skip:
1347   \__enumext_unskip_unkern:
1348   \mode_if_vertical:TF
1349   {
1350     \nopagebreak\nointerlineskip

```

```

1351     }
1352     {
1353         \par\nopagebreak\nointerlineskip
1354         \skip_zero:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1355     }
1356     \int_compare:nNnTF
1357     { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } > { 1 }
1358     {
1359         \addvspace{ 0.445\box_ht:N \strutbox }
1360     }
1361     {
1362         \addvspace{ 0.250\box_ht:N \strutbox }
1363     }
1364 }

```

(End of definition for \l__enumext_minipage_set_skip: and \l__enumext_minipage_add_space:.)

\l__enumext_pre_itemsep_skip: The function \l__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.

```

1365 \cs_new_protected:Nn \l__enumext_pre_itemsep_skip:
1366 {
1367     \int_case:nn { \l__enumext_level_int }
1368     {
1369         { 2 }{
1370             \skip_if_eq:nnTF
1371             { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1372             {
1373                 \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1374                 \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1375             }
1376             {
1377                 \dim_compare:nNnT
1378                 { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1379                 {
1380                     \skip_sub:Nn
1381                     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1382                     \skip_sub:Nn
1383                     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1384                     \skip_add:Nn
1385                     \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1386                     \skip_add:Nn
1387                     \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1388                 }
1389                 \dim_compare:nNnT
1390                 { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1391                 {
1392                     \skip_set:Nn \l__enumext_minipage_temp_skip
1393                     {
1394                         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1395                     }
1396                     \skip_sub:Nn
1397                     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1398                     \skip_sub:Nn
1399                     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1400                     \skip_add:Nn
1401                     \l__enumext_minipage_after_skip
1402                     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1403                     \skip_add:Nn
1404                     \l__enumext_multicols_below_ii_skip
1405                     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1406                 }
1407             }
1408         }
1409         { 3 }{
1410             \skip_if_eq:nnTF
1411             { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1412             {
1413                 \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1414                 \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1415             }

```



```

1416 {
1417     \dim_compare:nNnT
1418     { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1419     {
1420         \skip_sub:Nn
1421         \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1422         \skip_sub:Nn
1423         \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1424         \skip_add:Nn
1425         \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1426         \skip_add:Nn
1427         \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1428     }
1429     \dim_compare:nNnT
1430     { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1431     {
1432         \skip_set:Nn \l__enumext_minipage_temp_skip
1433         {
1434             \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1435         }
1436         \skip_sub:Nn
1437         \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1438         \skip_sub:Nn
1439         \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1440         \skip_add:Nn
1441         \l__enumext_minipage_after_skip
1442         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1443         \skip_add:Nn
1444         \l__enumext_multicols_below_iii_skip
1445         { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1446     }
1447 }
1448 }
1449 { 4 }{
1450     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1451     {
1452         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1453         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1454     }
1455     {
1456         \dim_compare:nNnT
1457         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1458         {
1459             \skip_sub:Nn
1460             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1461             \skip_sub:Nn
1462             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1463             \skip_add:Nn
1464             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1465             \skip_add:Nn
1466             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1467         }
1468         \dim_compare:nNnT
1469         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1470         {
1471             \skip_set:Nn \l__enumext_minipage_temp_skip
1472             {
1473                 \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1474             }
1475             \skip_sub:Nn
1476             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1477             \skip_sub:Nn
1478             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1479             \skip_add:Nn
1480             \l__enumext_minipage_after_skip
1481             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1482             \skip_add:Nn
1483             \l__enumext_multicols_below_iv_skip
1484             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1485         }
1486     }

```

```

1487         }
1488     }
1489 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

13.23.2 Adjustment of vertical spaces for minipage in keyans

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](#).

```

1490 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1491 {
1492     \skip_zero:N \l__enumext_minipage_after_skip
1493     \skip_zero:N \l__enumext_minipage_left_skip
1494     \skip_zero:N \l__enumext_minipage_right_skip
1495     \skip_set:Nn \l__enumext_minipage_right_skip
1496     {
1497         \l__enumext_topsep_v_skip
1498     }
1499     \mode_if_vertical:T
1500     {
1501         \skip_add:Nn \l__enumext_minipage_right_skip
1502         {
1503             \l__enumext_partopsep_v_skip
1504         }
1505     }
1506     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1507     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1508     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1509     \__enumext_keyans_pre_itemsep_skip:
1510     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1511     {
1512         \skip_zero:N \topskip
1513         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1514     }
1515 }
1516 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1517 {
1518     \__enumext_keyans_minipage_set_skip:
1519     \__enumext_unskip_unkern:
1520     \mode_if_vertical:TF
1521     {
1522         \nopagebreak\nointerlineskip
1523     }
1524     {
1525         \par\nopagebreak\nointerlineskip
1526         \skip_zero:N \l__enumext_partopsep_v_skip
1527     }
1528     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1529     {
1530         \addvspace{ 0.445\box_ht:N \strutbox }
1531     }
1532     {
1533         \addvspace{ 0.250\box_ht:N \strutbox }
1534     }
1535 }
1536 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1537 {
1538     \skip_if_eq:nnTF
1539     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1540     {
1541         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1542         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1543     }
1544     {
1545         \dim_compare:nNnT
1546         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1547         {
1548             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1549             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1550             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1551         \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1552     }
1553 \dim_compare:nNnT
1554 { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1555 {
1556     \skip_set:Nn \l__enumext_minipage_temp_skip
1557     {
1558         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1559     }
1560     \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1561     \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1562     \skip_add:Nn \l__enumext_minipage_after_skip
1563     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1564     \skip_add:Nn \l__enumext_multicols_below_v_skip
1565     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1566 }
1567 }
1568 }

```

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

13.23.3 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*`.

```

1569 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1570 {
1571     \skip_zero_new:N \l__enumext_minipage_left_skip
1572     \skip_gzero_new:N \g__enumext_minipage_right_skip
1573     \skip_gzero_new:N \g__enumext_minipage_after_skip
1574     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1575     {
1576         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1577         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1578     }
1579     {
1580         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1581         \skip_gset:Nn \g__enumext_minipage_right_skip
1582         {
1583             \l__enumext_topsep_vii_skip
1584         }
1585         \skip_gset:Nn \g__enumext_minipage_after_skip
1586         {
1587             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1588         }
1589     }
1590 }
1591 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1592 {
1593     \skip_zero_new:N \l__enumext_minipage_after_skip
1594     \skip_zero_new:N \l__enumext_minipage_left_skip
1595     \skip_zero_new:N \l__enumext_minipage_right_skip
1596     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1597     {
1598         \skip_set:Nn \l__enumext_minipage_left_skip
1599         {
1600             0.5\box_dp:N \strutbox
1601         }
1602         \skip_set:Nn \l__enumext_minipage_right_skip
1603         {
1604             \l__enumext_partopsep_viii_skip
1605         }
1606         \skip_set:Nn \l__enumext_minipage_after_skip
1607         {
1608             1.6\box_dp:N \strutbox
1609         }
1610     }
1611     {
1612         \skip_set:Nn \l__enumext_minipage_left_skip
1613         {

```

```

1614         0.5875\box_dp:N \strutbox
1615     }
1616     \skip_set:Nn \l__enumext_minipage_right_skip
1617     {
1618         \l__enumext_topsep_viii_skip
1619     }
1620     \skip_set:Nn \l__enumext_minipage_after_skip
1621     {
1622         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1623     }
1624 }
1625 }

```

(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 T_EX is in *(horizontal mode)* or *(vertical mode)*, since `\partopsep` is equal to `0pt` in both environments.

```

1626 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1627 {
1628     \__enumext_mini_set_vskip_vii:
1629     \par\nopagebreak
1630     \addvspace { \l__enumext_minipage_left_skip }
1631 }
1632 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1633 {
1634     \__enumext_mini_set_vskip_viii:
1635     \par\nopagebreak
1636     \addvspace { \l__enumext_minipage_left_skip }
1637 }

```

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

13.23.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_page` environment on the “left side”, open the `__enumext_mini_page` environment on the “right side” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “right side” environment. The *starred argument* ‘*’ inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `__enumext_mini_page` on the “right side”.

`\miniright`

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

```

1638 \NewDocumentCommand \miniright { s }
1639 {
1640     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1641     {
1642         \msg_error:nnn { enumext } { wrong-miniright-place }
1643     }
1644     % outside
1645     \bool_lazy_and:nnT
1646     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1647     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1648     {
1649         \msg_error:nnn { enumext } { wrong-miniright-place }
1650     }
1651     % starred env
1652     \bool_lazy_and:nnT
1653     { \bool_if_p:N \g__enumext_starred_bool }
1654     { \bool_not_p:n { \l__enumext_standar_bool } }
1655     {
1656         \msg_error:nnn { enumext } { wrong-miniright-starred }
1657     }
1658     % exec
1659     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1660     {
1661         \__enumext_keyans_mini_right_cmd:n {#1}
1662     }

```

```

1663 { \__enumext_mini_right_cmd:n {#1} }
1664 }

```

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

`__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.

```

1665 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1666 {
1667   \dim_compare:nNnTF
1668   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1669   {
1670     \__enumext_multicols_stop:
1671     \int_compare:nNnT
1672     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1673     {
1674       \par\addvspace{ \l__enumext_minipage_after_skip }
1675     }
1676     \end__enumext_mini_page
1677     \hfill
1678     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1679     \par\nointerlineskip
1680     \addvspace { \l__enumext_minipage_right_skip }
1681     \bool_if:nF {#1}
1682     {
1683       \centering
1684     }
1685     \int_gzero:N \g__enumext_minipage_stat_int
1686   }
1687   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1688   % paranoia
1689   \RenewDocumentCommand \miniright { s }
1690   {
1691     \msg_error:nn { enumext } { many-miniright-used }
1692   }
1693 }

```

(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.

```

1694 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1695 {
1696   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1697   {
1698     \__enumext_keyans_multicols_stop:
1699     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1700     {
1701       \par\addvspace{ \l__enumext_minipage_after_skip }
1702     }
1703     \end__enumext_mini_page
1704     \hfill
1705     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1706     \par\nointerlineskip
1707     \addvspace { \l__enumext_minipage_right_skip }
1708     \bool_if:nF {#1}
1709     {
1710       \centering
1711     }
1712     \int_gzero:N \g__enumext_minipage_stat_int
1713   }
1714   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1715   % paranoia
1716   \RenewDocumentCommand \miniright { s }
1717   {
1718     \msg_error:nn { enumext } { many-miniright-used }
1719   }

```

```

1719     }
1720 }

```

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

13.24 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.

```

above* 1721 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below   1722 {
below*  1723   \keys_define:nn { enumext / #1 }
        1724   {
        1725     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
        1726     above .value_required:n = true,
        1727     above* .code:n = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
        1728               \keys_set:nn { enumext / #1 } { above = {##1} },
        1729     above* .value_required:n = true,
        1730     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
        1731     below .value_required:n = true,
        1732     below* .code:n = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
        1733               \keys_set:nn { enumext / #1 } { below = {##1} },
        1734     below* .value_required:n = true,
        1735   }
        1736 }
1737 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.24.1 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.

```

1738 \cs_new_protected:Nn \__enumext_vspace_above:
1739 {
1740   \skip_if_eq:nnF
1741   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1742   {
1743     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1744     {
1745       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1746     }
1747     {
1748       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1749     }
1750   }
1751 }

```

(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.

```

1752 \cs_new_protected:Nn \__enumext_vspace_below:
1753 {
1754   \skip_if_eq:nnF
1755   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1756   {
1757     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1758     {
1759       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1760     }
1761     {
1762       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1763     }
1764   }
1765 }

```

(End of definition for `__enumext_vspace_below:`.)

13.24.2 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.

```

1766 \cs_new_protected:Nn \__enumext_vspace_above_v:
1767 {
1768   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1769   {
1770     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1771     {
1772       \vspace*{ \l__enumext_vspace_above_v_skip }
1773     }
1774     { \vspace { \l__enumext_vspace_above_v_skip } }
1775   }
1776 }

```

(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.

```

1777 \cs_new_protected:Nn \__enumext_vspace_below_v:
1778 {
1779   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1780   {
1781     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1782     {
1783       \vspace*{ \l__enumext_vspace_below_v_skip }
1784     }
1785     { \vspace { \l__enumext_vspace_below_v_skip } }
1786   }
1787 }

```

(End of definition for `__enumext_vspace_below_v:`.)

13.24.3 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:`

```

1788 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1789 {
1790   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1791   {
1792     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1793     {
1794       \vspace*{ \l__enumext_vspace_above_vii_skip }
1795     }
1796     { \vspace { \l__enumext_vspace_above_vii_skip } }
1797   }
1798 }
1799 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1800 {
1801   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1802   {
1803     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1804     {
1805       \vspace*{ \l__enumext_vspace_above_viii_skip }
1806     }
1807     { \vspace { \l__enumext_vspace_above_viii_skip } }
1808   }
1809 }

```

(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:`

```

1810 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1811 {
1812   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1813   {
1814     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1815     {
1816       \vspace*{ \l__enumext_vspace_below_vii_skip }

```



```

1817     }
1818     { \vspace { \l__enumext_vspace_below_vii_skip } }
1819   }
1820 }
1821 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1822 {
1823   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1824   {
1825     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1826     {
1827       \vspace*{ \l__enumext_vspace_below_viii_skip }
1828     }
1829     { \vspace { \l__enumext_vspace_below_viii_skip } }
1830   }
1831 }

```

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

13.25 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 $\langle keys \rangle$ passed to the *optional argument* of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific $\langle keys \rangle$. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in `chat-Tex-SX`

We define the keys `series`, `resume` and `resume*` only for the “first level” of `enumext` and `enumext*`.

```

series
resume
resume*
1832 \cs_set_protected:Npn \__enumext_tmp:n #1
1833 {
1834   \keys_define:nn { enumext / #1 }
1835   {
1836     series .str_set:N = \l__enumext_series_str,
1837     series .value_required:n = true,
1838     resume .code:n = \__enumext_resume_series:n {##1},
1839     resume* .code:n = \__enumext_resume_starred:,
1840     resume* .value_forbidden:n = true,
1841   }
1842 }
1843 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.25.1 Internal functions for series key

The function `__enumext_filter_series:n` will be in charge of filtering the $\langle keys \rangle$ we want to store where `{#1}` represents the *optional argument* passed to the environment.

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1844 \cs_new:Npn \__enumext_filter_series:n #1
1845 {
1846   \use:e
1847   {
1848     \keyval_parse:NNn
1849     \__enumext_filter_series_key:n
1850     \__enumext_filter_series_pair:nn {#1}
1851   }
1852 }

```

The function `__enumext_filter_series_key:n` will be responsible for filtering the $\langle keys \rangle$ that are passed “without value” by excluding the `resume`, `resume*` and `base-fix` keys.

```

1853 \cs_new:Npn \__enumext_filter_series_key:n #1
1854 {
1855   \str_case:nnF {#1}
1856   {
1857     { resume } {} { resume* } {} { base-fix } {}
1858   }
1859   { , { \exp_not:n {#1} } }
1860 }

```

The function `__enumext_filter_series_pair:nn` will be responsible for filtering the $\langle keys \rangle$ that are passed “with value” by excluding the `series`, `resume`, `start`, `start*`, `save-ans` and `save-key` keys.

```

1861 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1862 {
1863   \str_case:nnF {#1}
1864   {
1865     { series } {} { resume } {} { start } {}

```

```

1866         { start* } {} { save-ans } {} { save-key } {}
1867     }
1868     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1869 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *keys*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.39) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.44).

```

1870 \cs_new_protected:Npn \__enumext_parse_series:n #1
1871 {
1872     \str_if_empty:NTF \l__enumext_series_str
1873     {
1874         \bool_if:NF \l__enumext_resume_active_bool
1875         {
1876             \__enumext_resume_last:n {#1}
1877         }
1878     }
1879     {
1880         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1881         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1882             { \__enumext_filter_series:n {#1} }
1883         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1884         {
1885             \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1886         }
1887     }
1888 }

```

The function `__enumext_resume_last:n` will be in charge of saving the filtering *keys* when the `series` key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the `enumext` environment and in the variable `\g__enumext_starred_series_tl` for the `enumext*` environment.

```

1889 \cs_new_protected:Npn \__enumext_resume_last:n #1
1890 {
1891     \bool_if:NT \l__enumext_standar_first_bool
1892     {
1893         \tl_gclear:N \g__enumext_standar_series_tl
1894         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1895     }
1896     \bool_if:NT \l__enumext_starred_first_bool
1897     {
1898         \tl_gclear:N \g__enumext_starred_series_tl
1899         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1900     }
1901 }

```

(End of definition for `__enumext_parse_series:n` and `__enumext_resume_last:n`.)

13.25.2 Internal function to save counter value

```
\__enumext_resume_save_counter:
```

The `__enumext_resume_save_counter:` function will save the last counter value to `\g__enumext_series_⟨series name⟩_int` if the `series={⟨series name⟩}` key has been passed, to `\g__enumext_resume_int` if it has passed the key `resume without value` and the key `series` is not active, in `\g__enumext_series_⟨series name⟩_int` if the key `resume={⟨series name⟩}` has been passed and in `\g__enumext_series_⟨store name⟩_int` if the key has been passed `save-ans={⟨store name⟩}`.

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§13.39) and the `enumext*` environment definition (§13.44).

```

1902 \cs_new_protected:Nn \__enumext_resume_save_counter:
1903 {

```

```

1904 \bool_if:NT \g__enumext_standar_bool
1905 {
1906   \tl_if_empty:NF \l__enumext_series_str
1907   {
1908     \int_gset_eq:cN
1909     { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1910   }
1911   \tl_if_empty:NTF \l__enumext_resume_name_tl
1912   {
1913     \str_if_empty:NT \l__enumext_series_str
1914     {
1915       \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1916     }
1917   }
1918   {
1919     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1920     {
1921       \int_gset_eq:cN
1922       { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1923     }
1924   }
1925   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1926   {
1927     \int_gset_eq:cN
1928     { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1929   }
1930 }
1931 \bool_if:NT \g__enumext_starred_bool
1932 {
1933   \tl_if_empty:NF \l__enumext_series_str
1934   {
1935     \int_gset_eq:cN
1936     { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1937   }
1938   \tl_if_empty:NTF \l__enumext_resume_name_tl
1939   {
1940     \str_if_empty:NT \l__enumext_series_str
1941     {
1942       \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1943     }
1944   }
1945   {
1946     \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1947     {
1948       \int_gset_eq:cN
1949       { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1950     }
1951   }
1952   \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1953   {
1954     \int_gset_eq:cN
1955     { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1956   }
1957 }
1958 }

```

(End of definition for `__enumext_resume_save_counter:`.)

13.25.3 Internal functions for resume key

`__enumext_resume_series:n`

The function `__enumext_resume_series:n` will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function `__enumext_resume_counter:` is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```

1959 \cs_new_protected:Npn \__enumext_resume_series:n #1
1960 {
1961   \tl_if_empty:NTF {#1}
1962   {
1963     \__enumext_resume_counter:n { }

```

```

1964     }
1965     {
1966         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1967         {
1968             \__enumext_resume_counter:n {#1}
1969             \bool_if:NT \g__enumext_standar_bool
1970             {
1971                 \keys_set:nv { enumext / level-1 }
1972                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1973             }
1974             \bool_if:NT \g__enumext_starred_bool
1975             {
1976                 \keys_set:nv { enumext / enumext* }
1977                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1978             }
1979         }
1980     }
1981     \bool_if:NT \g__enumext_standar_bool
1982     {
1983         \msg_error:nnn { enumext } { unknown-series } {#1}
1984     }
1985     \bool_if:NT \g__enumext_starred_bool
1986     {
1987         \msg_error:nnn { enumext } { unknown-series } {#1}
1988     }
1989 }
1990 }
1991 }

```

(End of definition for __enumext_resume_series:n.)

```

\__enumext_resume_counter:n
\__enumext_resume_counter:
  \__enumext_resume_counter_series:
  \__enumext_resume_counter_save_ans:

```

The function `__enumext_resume_counter:n` will set the variable `\l__enumext_resume_active_bool` to true and pass the value of the key `resume` to the variable `\l__enumext_series_name_tl` which will contain the `{\series name}`. If the variable `\l__enumext_series_name_tl` is empty, that is, we are passing the key `resume` *without value*, we will execute the function `__enumext_resume_counter:`; otherwise, when we pass `resume={\series name}` we will execute the function `__enumext_resume_counter_series:`, finally we will execute the function `__enumext_resume_counter_save_ans:` which is associated with the key `save-ans`.

```

1992 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1993 {
1994     \bool_set_true:N \l__enumext_resume_active_bool
1995     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1996     \tl_if_empty:NTF \l__enumext_resume_name_tl
1997     {
1998         \__enumext_resume_counter:
1999     }
2000     {
2001         \__enumext_resume_counter_series:
2002     }
2003     \__enumext_resume_counter_save_ans:
2004 }

```

The `__enumext_resume_counter:` function is executed when the `resume` key is used *without value*, only the counters for the “*first level*” of the environments will be set.

```

2005 \cs_new_protected:Nn \__enumext_resume_counter:
2006 {
2007     \bool_if:NT \g__enumext_standar_bool
2008     {
2009         \int_gincr:N \g__enumext_resume_int
2010         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2011     }
2012     \bool_if:NT \g__enumext_starred_bool
2013     {
2014         \int_gincr:N \g__enumext_resume_vii_int
2015         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2016     }
2017 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume={\series name}` key is active, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the `series` key.

```

2018 \cs_new_protected:Nn \__enumext_resume_counter_series:
2019 {
2020   \bool_if:NT \g__enumext_standar_bool
2021   {
2022     \int_set:Nn \l__enumext_start_i_int
2023     {
2024       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2025     }
2026   }
2027   \bool_if:NT \g__enumext_starred_bool
2028   {
2029     \int_set:Nn \l__enumext_start_vii_int
2030     {
2031       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2032     }
2033   }
2034 }

```

The function `__enumext_resume_counter_save_ans:` will be executed when the `save-ans` key is active along with the `resume` key, setting the counters for the “first level” of the environments according to the value of the integer variables created by the `save-ans` key.

```

2035 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2036 {
2037   \bool_lazy_and:nnT
2038   { \bool_if_p:N \l__enumext_standar_first_bool }
2039   { \bool_if_p:N \l__enumext_store_active_bool }
2040   {
2041     \int_set:Nn \l__enumext_start_i_int
2042     {
2043       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2044     }
2045   }
2046   \bool_lazy_and:nnT
2047   { \bool_if_p:N \l__enumext_starred_first_bool }
2048   { \bool_if_p:N \l__enumext_store_active_bool }
2049   {
2050     \int_set:Nn \l__enumext_start_vii_int
2051     {
2052       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2053     }
2054   }
2055 }

```

(End of definition for `__enumext_resume_counter:n` and others.)

13.25.4 Internal function for `resume*` key

`__enumext_resume_starred:`

The function `__enumext_resume_starred:` 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 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.

```

2056 \cs_new_protected:Nn \__enumext_resume_starred:
2057 {
2058   \bool_if:NT \g__enumext_standar_bool
2059   {
2060     \tl_if_empty:NF \g__enumext_standar_series_tl
2061     {
2062       \__enumext_resume_counter:n { }
2063       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2064     }
2065   }
2066   \bool_if:NT \g__enumext_starred_bool
2067   {
2068     \tl_if_empty:NF \g__enumext_starred_series_tl
2069     {
2070       \__enumext_resume_counter:n { }
2071       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2072     }
2073   }
2074 }

```

(End of definition for `__enumext_resume_starred:`.)

13.26 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.

13.26.1 Setting save-ans key

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

```

2075 \cs_set_protected:Npn \__enumext_tmp:n #1
2076 {
2077   \keys_define:nn { enumext / #1 }
2078   {
2079     save-ans .code:n = \__enumext_storing_set:n {##1},
2080     save-ans .value_required:n = true,
2081   }
2082 }
2083 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.26.2 Internal functions for save-ans key

`__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:`.

```

2084 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2085 {
2086   \msg_term:nnVV { enumext } { save-ans-log }
2087   \g__enumext_envir_name_tl \l__enumext_store_name_tl
2088 }
2089 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2090 {
2091   \msg_term:nnVV { enumext } { save-ans-log-hook }
2092   \g__enumext_envir_name_tl \g__enumext_store_name_tl
2093 }
```

(End of definition for `__enumext_start_save_ans_msg:` and `__enumext_stop_save_ans_msg:`.)

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

```

2094 \cs_new_protected:Npn \__enumext_storing_set:n #1
2095 {
2096   \tl_set:Nx \l__enumext_store_name_tl {#1}
2097   \tl_if_empty:NTF \l__enumext_store_name_tl
2098   {
2099     \bool_lazy_or:nnT
2100     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2101     {
2102       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2103     }
2104   }
2105   {
2106     \bool_lazy_or:nnT
2107     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2108     {
2109       \__enumext_start_save_ans_msg:
2110       \__enumext_storing_exec:
2111     }
2112   }
2113 }
```

The function `__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to “true” the variable `\l__enumext_check_answers_bool` used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, copy `{⟨store name⟩}` into the variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§13.31).

```

2114 \cs_new_protected:Nn \__enumext_storing_exec:
```

```

2115 {
2116   \bool_set_true:N \l__enumext_store_active_bool
2117   \bool_set_true:N \l__enumext_check_answers_bool
2118   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2119   \__enumext_anskey_env_make:V \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*`.

```

2120   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2121   {
2122     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2123     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2124   }
2125   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2126   {
2127     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2128     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2129   }
2130   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2131   {
2132     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2133     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2134   }
2135 }

```

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

13.26.3 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.

13.26.4 Setting check-ans and no-store keys

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

```

check-ans no-store
2136 \cs_set_protected:Npn \__enumext_tmp:n #1
2137 {
2138   \keys_define:nn { enumext / #1 }
2139   {
2140     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2141     check-ans .initial:n = false,
2142     check-ans .value_required:n = true,
2143     no-store .code:n = {
2144       \bool_set_false:N \l__enumext_check_answers_bool

```



```

2145         \bool_set_false:N \l__enumext_check_ans_key_bool
2146     },
2147     no-store .value_forbidden:n = true,
2148 }
2149 }
2150 \clist_map_inline:nn
2151 {
2152     level-1, level-2, level-3, level-4, enumext*
2153 }
2154 { \l__enumext_tmp:n {#1} }

```

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

13.26.5 Set-up check answer mechanism

`\l__enumext_check_ans_active:` The function `\l__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 `\l__enumext_check_ans_level:` only if “*true*”, i.e. the key *no-store* is not active.

```

2155 \cs_new_protected:Nn \l__enumext_check_ans_active:
2156 {
2157     \tl_if_empty:NF \l__enumext_store_name_tl
2158     {
2159         \bool_if:NT \l__enumext_check_answers_bool
2160         {
2161             \l__enumext_check_ans_level:
2162         }
2163     }
2164 }

```

The function `\l__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*”.

```

2165 \cs_new_protected:Nn \l__enumext_check_ans_level:
2166 {
2167     \int_case:nn { \l__enumext_level_int }
2168     {
2169         { 1 }{
2170             \bool_lazy_all:nT
2171             {
2172                 { \bool_if_p:N \g__enumext_starred_bool }
2173                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2174             }
2175             {
2176                 \int_gdecr:N \g__enumext_item_number_int
2177                 \bool_set_false:N \l__enumext_item_number_bool
2178             }
2179         }
2180         { 2 }{
2181             \int_gdecr:N \g__enumext_item_number_int
2182             \bool_set_false:N \l__enumext_item_number_bool
2183         }
2184         { 3 }{
2185             \int_gdecr:N \g__enumext_item_number_int
2186             \bool_set_false:N \l__enumext_item_number_bool
2187         }
2188         { 4 }{
2189             \int_gdecr:N \g__enumext_item_number_int
2190             \bool_set_false:N \l__enumext_item_number_bool
2191         }
2192     }

```

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.

```

2193     \int_case:nn { \l__enumext_level_h_int }
2194     {
2195         { 1 }{
2196             \bool_lazy_all:nT
2197             {
2198                 { \bool_if_p:N \g__enumext_standar_bool }
2199                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }

```

```

2200         }
2201         {
2202             \int_gdecr:N \g__enumext_item_number_int
2203             \bool_set_false:N \l__enumext_item_number_bool
2204         }
2205     }
2206 }
2207 }

```

(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.

```

2208 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2209 {
2210     \bool_lazy_and:nnT
2211     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2212     { \bool_if_p:N \g__enumext_standar_bool }
2213     {
2214         \bool_gset_true:N \g__enumext_check_ans_key_bool
2215     }
2216     \bool_lazy_and:nnT
2217     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2218     { \bool_if_p:N \g__enumext_starred_bool }
2219     {
2220         \bool_gset_true:N \g__enumext_check_ans_key_bool
2221     }
2222 }

```

(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:.

```

2223 \cs_new_protected:Nn \__enumext_item_answer_diff:
2224 {
2225     \int_gset:Nn \g__enumext_item_answer_diff_int
2226     {
2227         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2228     }
2229 }

```

(End of definition for __enumext_item_answer_diff:.)

__enumext_check_ans_show:

__enumext_check_ans_msg_less:

__enumext_check_ans_msg_same_ok:

__enumext_check_ans_msg_greater:

The function __enumext_check_ans_show: will be executed within 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:.

```

2230 \cs_new_protected:Nn \__enumext_check_ans_show:
2231 {
2232     \int_case:nn { \g__enumext_item_answer_diff_int }
2233     {
2234         { -1 } { \__enumext_check_ans_msg_less: }
2235         { 0 } { \__enumext_check_ans_msg_same_ok: }
2236         { 1 } { \__enumext_check_ans_msg_greater: }
2237     }
2238 }
2239 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2240 {
2241     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2242     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2243 }
2244 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2245 {
2246     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2247     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2248 }
2249 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:

```

```

2250 {
2251     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2252     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2253 }

```

(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:`.

```

2254 \cs_new_protected:Nn \__enumext_check_ans_log:
2255 {
2256     \int_case:nn { \g__enumext_item_answer_diff_int }
2257     {
2258         { -1 } { \__enumext_check_ans_log_msg_less: }
2259         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2260         { 1 } { \__enumext_check_ans_log_msg_greater: }
2261     }
2262 }
2263 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2264 {
2265     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2266     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2267 }
2268 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2269 {
2270     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2271     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2272 }
2273 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2274 {
2275     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2276     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2277 }

```

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

13.26.6 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.

```

2278 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2279 {
2280     \int_compare:nNt
2281     { \g__enumext_check_starred_cmd_int } = { 0 }
2282     {
2283         \msg_warning:nnnV
2284         { enumext } { missing-starred } { #1 } { \l__enumext_check_start_line_env_tl }
2285     }
2286     \int_compare:nNt
2287     { \g__enumext_check_starred_cmd_int } > { 1 }
2288     {
2289         \msg_warning:nnnV
2290         { enumext } { many-starred } { #1 } { \l__enumext_check_start_line_env_tl }
2291     }
2292     \int_gzero:N \g__enumext_check_starred_cmd_int
2293     \tl_clear:N \l__enumext_check_start_line_env_tl
2294 }

```

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

13.27 Keys and functions associated with storage

13.27.1 Keys for marks, wrapp 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* For the `keyans` and `keyans*` environments we will only add the keys `mark-ans*`, `mark-pos*`, `mark-sep*`,
 mark-pos* `wrap-ans*`, `wrap-opt`, `save-sep`, `show-ans` and `show-pos`.
 mark-sep* 2295 `\cs_set_protected:Npn __enumext_tmp:nn #1 #2`
 wrap-ans* 2296 `{`
 wrap-opt 2297 `\keys_define:nn { enumext / #1 }`
 save-sep 2298 `{`
 show-ans 2299 `mark-ans* .tl_set:c = { __enumext_mark_answer_sym_#2_tl },`
 show-pos 2300 `mark-ans* .initial:n = \textasteriskcentered,`
 2301 `mark-ans* .value_required:n = true,`
 2302 `mark-pos* .choice:,`
 2303 `mark-pos* / left .code:n = \str_set:cn { __enumext_mark_position_#2_str } { l },`
 2304 `mark-pos* / right .code:n = \str_set:cn { __enumext_mark_position_#2_str } { r },`
 2305 `mark-pos* / center .code:n = \str_set:cn { __enumext_mark_position_#2_str } { c },`
 2306 `mark-pos* / unknown .code:n =`
 2307 `\msg_error:nnee { enumext } { unknown-choice }`
 2308 `{ mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },`
 2309 `mark-pos* .initial:n = right,`
 2310 `mark-pos* .value_required:n = true,`
 2311 `mark-sep* .dim_set:c = { __enumext_mark_sym_sep_#2_dim },`
 2312 `mark-sep* .value_required:n = true,`
 2313 `wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,`
 2314 `wrap-ans* .value_required:n = true,`
 2315 `wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,`
 2316 `wrap-opt .initial:n = [{##1}],`
 2317 `wrap-opt .value_required:n = true,`
 2318 `save-sep .tl_set:c = { __enumext_store_keyans_item_opt_sep_#2_tl },`
 2319 `save-sep .initial:n = {, ~ },`
 2320 `save-sep .value_required:n = true,`
 2321 `show-ans .bool_set:N = __enumext_show_answer_bool,`
 2322 `show-ans .initial:n = false,`
 2323 `show-ans .value_required:n = true,`
 2324 `show-pos .bool_set:N = __enumext_show_position_bool,`
 2325 `show-pos .initial:n = false,`
 2326 `show-pos .value_required:n = true,`
 2327 `}`
 2328 `}`
 2329 `\clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { __enumext_tmp:nn #1 }`

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

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

mark-ans 2330 `\cs_set_protected:Npn __enumext_tmp:n #1`
 mark-pos 2331 `{`
 mark-sep 2332 `\keys_define:nn { enumext / #1 }`
 wrap-ans 2333 `{`
 mark-ans* 2334 `mark-ref .tl_set:N = __enumext_mark_ref_sym_tl,`
 mark-pos* 2335 `mark-ref .initial:n = \textreferencemark,`
 mark-sep* 2336 `mark-ref .value_required:n = true,`
 wrap-ans* 2337 `save-ref .bool_set:N = __enumext_store_ref_key_bool,`
 wrap-opt 2338 `save-ref .initial:n = false,`
 save-sep 2339 `save-ref .value_required:n = true,`
 2340 `show-ans .bool_set:N = __enumext_show_answer_bool,`
 2341 `show-ans .initial:n = false,`
 2342 `show-ans .value_required:n = true,`
 2343 `show-pos .bool_set:N = __enumext_show_position_bool,`
 2344 `show-pos .initial:n = false,`
 2345 `show-pos .value_required:n = true,`
 2346 `mark-ans .tl_set:N = __enumext_mark_answer_sym_tl,`
 2347 `mark-ans .initial:n = \textasteriskcentered,`
 2348 `mark-ans .value_required:n = true,`
 2349 `mark-sep .dim_set:N = __enumext_mark_sym_sep_dim,`
 2350 `mark-sep .value_required:n = true,`
 2351 `mark-pos .choice:,`
 2352 `mark-pos / left .code:n = \str_set:Nn __enumext_mark_position_str { l },`
 2353 `mark-pos / right .code:n = \str_set:Nn __enumext_mark_position_str { r },`
 2354 `mark-pos / center .code:n = \str_set:Nn __enumext_mark_position_str { c },`
 2355 `mark-pos / unknown .code:n =`
 2356 `\msg_error:nnee { enumext } { unknown-choice }`

```

2357             { mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },
2358 mark-pos .initial:n = right,
2359 mark-pos .value_required:n = true,
2360
2361 wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2362 wrap-ans .initial:n =
2363     {
2364         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2365     },
2366 wrap-ans .value_required:n = true,
2367 mark-ans* .code:n = {
2368     \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2369     \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2370 },
2371 mark-ans* .value_required:n = true,
2372 mark-pos* .code:n = {
2373     \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2374     \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2375 },
2376 mark-pos* .value_required:n = true,
2377 mark-sep* .code:n = {
2378     \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2379     \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2380 },
2381 mark-sep* .value_required:n = true,
2382 wrap-ans* .code:n = {
2383     \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2384     \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2385 },
2386 wrap-ans* .value_required:n = true,
2387 wrap-opt .code:n = {
2388     \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2389     \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2390 },
2391 wrap-opt .value_required:n = true,
2392 save-sep .code:n = {
2393     \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2394     \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2395 },
2396 save-sep .value_required:n = true,
2397 }
2398 }
2399 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.27.2 Storing structure of the environments

The idea behind “*storing structure*” in the *sequence* is to have a copy of the *structure of the environment* in which the key `save-ans` is being executed so we must capture the *optional argument* passed to the levels of the environment in which it is executed and “*storing*” this in the *sequence*.

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n` will be responsible for the “*storing keys*” filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `__enumext_store_save_key_X_tl` with the filtered *keys*.

```

2400 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2401 {
2402     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2403     {
2404         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2405         \tl_set:ce
2406             { l__enumext_store_save_key_ \__enumext_level: _tl }
2407             { \__enumext_filter_save_key:n {#1} }
2408     }
2409 }
2410 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2411 {
2412     \bool_if:NF \l__enumext_store_save_key_vii_bool
2413     {

```

```

2414         \tl_clear:N \l__enumext_store_save_key_vii_tl
2415         \tl_set:Nx \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2416     }
2417 }

```

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

13.27.3 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.

```

2418 \cs_set_protected:Npn \__enumext_tmp:n #1
2419 {
2420     \keys_define:nn { enumext / enumext* }
2421     {
2422         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2423         save-key .value_required:n = true,
2424     }
2425     \keys_define:nn { enumext / #1 }
2426     {
2427         save-key .code:n = \__enumext_parse_save_key:n {##1},
2428         save-key .value_required:n = true,
2429     }
2430 }
2431 \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*`.

```

2432 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2433 {
2434     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2435     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2436     \tl_set:ce
2437     { l__enumext_store_save_key_ \__enumext_level: _tl }
2438     { \__enumext_filter_save_key:n {#1} }
2439 }
2440 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2441 {
2442     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2443     \tl_clear:N \l__enumext_store_save_key_vii_tl
2444     \tl_set:Nx \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2445 }

```

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

13.27.4 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.

```

2446 \cs_new:Npn \__enumext_filter_save_key:n #1
2447 {
2448     \use:e
2449     {
2450         \keyval_parse:NNn
2451         \__enumext_filter_save_key_key:n
2452         \__enumext_filter_save_key_pair:nn {#1}
2453     }
2454 }

```

The function `__enumext_filter_save_key_key:n` will be responsible for “*filtering keys*” that are passed “*without value*” by excluding the `resume`, `resume*`, `no-store` and `base-fix` keys.

```

2455 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2456 {
2457     \str_case:nnF {#1}
2458     {

```

```

2459         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2460     }
2461     { , { \exp_not:n {#1} } }
2462 }

```

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`, `wrap-ans`, `mark-ans`, `wrap-opt`, `wrap-ans*`, `save-sep`, `mark-sep`, `mark-ref`, `save-sep`, `mini-env`, `mini-sep`, `mini-right` and `mini-right*` keys.

```

2463 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2464 {
2465     \str_case:nnF {#1}
2466     {
2467         { series } {} { resume } {} { save-ans } {} { save-ref } {}
2468         { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2469         { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { wrap-ans* } {}
2470         { save-sep } {} { mark-sep } {} { mark-ref } {} { mini-env } {}
2471         { mini-sep } {} { mini-right } {} { mini-right* } {}
2472     }
2473     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2474 }

```

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

13.27.5 Function for storing content in prop list

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

`__enumext_store_addto_prop:V`

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

```

2475 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2476 {
2477     \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2478     {
2479         \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2480     }
2481     { #1 }
2482 }
2483 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

13.27.6 Function for storing content in sequence

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

`__enumext_store_addto_seq:v`

`__enumext_store_addto_seq:V`

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.

```

2484 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2485 {
2486     \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2487 }
2488 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

13.27.7 Functions for storing structure in the sequence

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

`__enumext_store_level_close:`

```

2489 \cs_new_protected:Nn \__enumext_store_level_open:
2490 {
2491     \bool_if:NT \__enumext_check_answers_bool
2492     {
2493         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2494         {
2495             \__enumext_store_addto_seq:n
2496             {
2497                 \item \begin{enumext}
2498             }

```



```

2499     }
2500     {
2501         \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2502         {
2503             \item \begin{enumext} [
2504             ]
2505             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2506             {
2507             ]
2508             }
2509             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2510         }
2511     }
2512 }
2513 \cs_new_protected:Nn \__enumext_store_level_close:
2514 {
2515     \bool_if:NT \l__enumext_check_answers_bool
2516     {
2517         \__enumext_store_addto_seq:n { \end{enumext} }
2518     }
2519 }

```

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

__enumext_store_level_open_vii:
 __enumext_store_level_close_vii:

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

```

2520 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2521 {
2522     \bool_if:NT \l__enumext_check_answers_bool
2523     {
2524         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2525         {
2526             \__enumext_store_addto_seq:n
2527             {
2528                 \item \begin{enumext*}
2529             }
2530         }
2531         {
2532             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2533             {
2534                 \item \begin{enumext*}[
2535                 ]
2536             }
2537             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2538             {
2539             ]
2540             }
2541             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2542         }
2543     }
2544 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2545 {
2546     \bool_if:NT \l__enumext_check_answers_bool
2547     {
2548         \__enumext_store_addto_seq:n { \end{enumext*} }
2549     }
2550 }

```

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

13.27.8 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

```

2551 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2552 {
2553     \mode_leave_vertical:
2554     \skip_horizontal:n { -\dim_use:N #2 }
2555     \hbox_overlap_left:n

```

```

2556     {
2557         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2558         {
2559             \tl_use:N \l__enumext_mark_answer_sym_tl
2560         }
2561     }
2562     \skip_horizontal:n { \dim_use:N #2 }
2563 }
2564 \cs_generate_variant:Nn \l__enumext_print_keyans_box:NN { cc }

```

(End of definition for `\l__enumext_print_keyans_box:NN`.)

13.28 The internal label and ref

The function `\l__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`.

`\l__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`.

```

2565 \cs_new_protected:Nn \l__enumext_store_internal_ref:
2566 {
2567     \cs_set_protected:Npn \l__enumext_tmp:n ##1
2568     {
2569         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2570         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2571         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2572         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2573     }
2574     \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {##1} }
2575     \cs_set:Npn \l__enumext_tmp:n ##1
2576     { . \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.

```

2577     \bool_lazy_all:nT
2578     {
2579         { \bool_if_p:N \g__enumext_starred_bool }
2580         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2581     }
2582     {
2583         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2584         { \tl_use:N \l__enumext_label_copy_vii_tl }
2585     }
2586     \bool_lazy_all:nT
2587     {
2588         { \bool_not_p:n { \g__enumext_standar_bool } }
2589         { \bool_if_p:N \l__enumext_standar_bool }
2590         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2591     }
2592     {
2593         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2594         {
2595             \tl_use:N \l__enumext_label_copy_vii_tl
2596             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2597         }
2598     }

```

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.

```

2599     \bool_lazy_all:nT
2600     {
2601         { \bool_if_p:N \g__enumext_standar_bool }
2602         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2603         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2604     }
2605     {
2606         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2607         {
2608             \tl_use:N \l__enumext_label_copy_i_tl
2609             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2610         }

```

```

2611     }
2612     \cs_set:Npn \__enumext_tmp:n ##1
2613     { \tl_use:c { \__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2614     \bool_lazy_all:nT
2615     {
2616         { \bool_if_p:N \g__enumext_standar_bool }
2617         { \bool_if_p:N \l__enumext_starred_bool }
2618         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2619     }
2620     {
2621         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2622         {
2623             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2624             \tl_use:N \l__enumext_label_copy_vii_tl
2625         }
2626     }

```

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

```

2627     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2628     {
2629         \l__enumext_store_name_tl \c_colon_str
2630         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2631     }

```

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

```

2632     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2633     {
2634         \__enumext_newlabel:nn
2635         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2636         { \l__enumext_newlabel_arg_two_tl }
2637     }
2638     \l__enumext_write_aux_file_tl
2639 }

```

(End of definition for `__enumext_store_internal_ref:`)

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

`__enumext_store_anskey_code:n`

The internal function `__enumext_store_anskey_code: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*” $\langle \textit{argument} \rangle$.

```

2640 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2641 {
2642     \int_gincr:N \g__enumext_item_anskey_int
2643     \__enumext_store_addto_prop:n {#1}
2644     \bool_if:NT \l__enumext_store_ref_key_bool
2645     {
2646         \__enumext_store_internal_ref:
2647     }
2648     \__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`.

```

2649     \tl_clear:N \l__enumext_store_anskey_arg_tl
2650     \bool_lazy_and:nnT
2651     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2652     { \bool_not_p:n { \l__enumext_starred_bool } }
2653     {
2654         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2655     }
2656     \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`.

```

2657     \bool_lazy_and:nnT
2658     { \bool_not_p:n { \l__enumext_starred_bool } }
2659     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2660     {
2661         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl

```

```

2662     {
2663     ( \exp_not:V \l__enumext_store_item_join_int )
2664     }
2665 }

```

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

```

2666 \bool_if:NTF \l__enumext_store_item_star_bool
2667 {
2668   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2669   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2670   {
2671     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2672     {
2673       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2674     }
2675   }
2676   \dim_compare:nT
2677   {
2678     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2679   }
2680   {
2681     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2682     {
2683       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2684     }
2685   }
2686   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2687 }
2688 {
2689   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2690 }

```

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*.

```

2691 \bool_lazy_and:nnT
2692 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2693 { \bool_if_p:N \l__enumext_hyperref_bool }
2694 {
2695   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2696   {
2697     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2698     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2699   }
2700 }
2701 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2702 }

```

(End of definition for `__enumext_store_anskey_code: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.

```

2703 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2704 {
2705   \par
2706   \bool_if:NTF \l__enumext_starred_bool
2707   {
2708     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2709     {
2710       \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2711     }
2712     \__enumext_print_keyans_box:NN
2713     \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2714   }
2715   {
2716     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2717     {
2718       \dim_set:Nn \l__enumext_mark_sym_sep_dim
2719       {
2720         \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2721       }
2722     }

```

```

2723     \__enumext_print_keyans_box:cc
2724     { \__enumext_labelwidth_ \__enumext_level: _dim } { \__enumext_mark_sym_sep_dim }
2725   }
2726   \__enumext_anskey_wrapper:n { #1 }
2727 }

```

(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.

```

2728 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2729 {
2730   \bool_if:NT \__enumext_show_answer_bool
2731   {
2732     \__enumext_anskey_show_wrap_arg:n { #1 }
2733   }
2734   \bool_if:NT \__enumext_show_position_bool
2735   {
2736     \tl_set:Nx \__enumext_mark_answer_sym_tl
2737     {
2738       \group_begin:
2739       \exp_not:N \normalfont
2740       \exp_not:N \footnotesize [ \int_eval:n
2741       {
2742         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
2743       }
2744       ]
2745       \group_end:
2746     }
2747     \__enumext_anskey_show_wrap_arg:n { #1 }
2748   }
2749 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

13.30 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[\langle key = val \rangle]{\langle content \rangle}`.

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

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

```

2750 \keys_define:nn { enumext / anskey }
2751 {
2752   break-col .bool_set:N = \__enumext_store_columns_break_bool,
2753   break-col .default:n = true,
2754   break-col .value_forbidden:n = true,
2755   item-join .int_set:N = \__enumext_store_item_join_int,
2756   item-join .value_required:n = true,
2757   item-star .bool_set:N = \__enumext_store_item_star_bool,
2758   item-star .default:n = true,
2759   item-star .value_forbidden:n = true,
2760   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
2761   item-sym* .value_required:n = true,
2762   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
2763   item-pos* .value_required:n = true,
2764   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2765 }

```

The $\langle keys \rangle$ are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `__enumext_anskey_unknown:n`.

```

2766 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2767 {
2768   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2769 }
2770 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2771 {
2772   \tl_if_blank:nTF {#2}
2773   {

```

```

2774         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2775     }
2776     {
2777         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2778     }
2779 }

```

(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_code:n`.

```

2780 \NewDocumentCommand \anskey { o +m }
2781 {
2782     \__enumext_anskey_safe_outer:
2783     \group_begin:
2784         \bool_if:NT \l__enumext_check_answers_bool
2785         {
2786             \tl_if_novalue:nF {#1}
2787             {
2788                 \keys_set:nn { enumext / anskey } {#1}
2789             }
2790             \tl_if_blank:nTF {#2}
2791             {
2792                 \msg_error:nn { enumext } { anskey-empty-arg }
2793             }
2794             {
2795                 \__enumext_anskey_safe_inner:
2796                 \__enumext_store_anskey_code:n {#2}
2797             }
2798         }
2799     \group_end:
2800 }

```

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

13.30.1 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:`

```

2801 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2802 {
2803     \bool_if:NF \l__enumext_store_active_bool
2804     {
2805         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2806     }
2807     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2808     {
2809         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2810     }
2811     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2812     {
2813         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2814     }
2815     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2816     {
2817         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2818     }
2819 }

```

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.

```

2820 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2821 {
2822     \int_incr:N \l__enumext_anskey_level_int
2823     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2824     {

```

```

2825     \msg_error:nn { enumext } { anskey-nested }
2826   }
2827   \bool_if:NF \l__enumext_item_number_bool
2828   {
2829     \msg_error:nn { enumext } { anskey-unnumber-item }
2830   }
2831   \mode_if_math:T
2832   {
2833     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2834   }
2835 }

```

(End of definition for `__enumext_anskey_safe_outer:` and `__enumext_anskey_safe_inner:`.)

13.31 The environment `anskey*`

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the `scontents` package, so to be able to have support at this point it is best to play a little with the internal code of `scontents` and `hooks`. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to `store-env`, if you are using the key `print-env` or the `write-out` key, sure, I can detect and modify it within the `enumext` and `enumext*` environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the `save-ans` key is executed. and have it available only when that key is executed, here I would have absolute control of the `<keys>` and I make sure that `write-out` is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

`__enumext_undefine_anskey_env:`

The function `__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `__enumext_execute_after_env:` (§13.32) which is executed after the environment in which the key `save-ans` is active.

```

2836 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2837 {
2838   \cs_undefine:c { anskey* }
2839   \cs_undefine:c { endanskey* }
2840   \cs_undefine:c { __scontents_anskey*_env_begin: }
2841   \cs_undefine:c { __scontents_anskey*_env_end: }
2842 }

```

Detection of the `anskey*` environment outside the `enumext` and `enumext*` environments.

```

2843 \__enumext_before_env:nn { enumext }
2844 {
2845   \bool_lazy_and:nnT
2846   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2847   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2848   {
2849     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2850     {
2851       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2852     }
2853   }
2854 }
2855 \__enumext_before_env:nn { enumext* }
2856 {
2857   \bool_lazy_and:nnT
2858   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2859   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2860   {
2861     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2862     {
2863       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2864     }
2865   }
2866 }

```

Detection of the `anskey*` environment inside the `keyans`, `keyans*` and `keyanspic` environments, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2867 \__enumext_before_env:nn { anskey* }
2868 {

```



```

2869 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2870 {
2871   \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2872 }
2873 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2874 {
2875   \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2876 }
2877 \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2878 {
2879   \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2880 }
2881 \bool_if:Nf \l__enumext_item_number_bool
2882 {
2883   \msg_error:nn { enumext } { anskey-unnumber-item }
2884 }
2885 \mode_if_math:T
2886 {
2887   \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2888 }
2889 }

```

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

anskey*

The function `\l__enumext_anskey_env_make:n` creates the environment **anskey*** (custom version of `scontents` environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`.

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `\l__enumext_storing_exec:` (§13.26.1) and we will execute it only if the variable `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package **scontents**.

```

2890 \cs_new_protected:Npn \l__enumext_anskey_env_make:n #1
2891 {
2892   \bool_if:NT \l__enumext_anskey_env_bool
2893   {
2894     \newenvsc{anskey*}[store-env=#1,print-env=false]
2895     \l__enumext_anskey_env_exec:
2896   }
2897 }
2898 \cs_generate_variant:Nn \l__enumext_anskey_env_make:n { V }

```

The function `\l__enumext_anskey_env_define_keys:` will add the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` and will leave the keys `print-env`, `store-env` and `write-out` undefined. We will apply this function using the *hook* function `\l__enumext_before_env:nn`.

```

2899 \cs_new_protected:Nn \l__enumext_anskey_env_define_keys:
2900 {
2901   \keys_define:nn { scontents / scontents }
2902   {
2903     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2904     break-col .default:n   = true,
2905     break-col .value_forbidden:n = true,
2906     item-join .int_gset:N   = \g__enumext_store_item_join_int,
2907     item-join .value_required:n = true,
2908     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2909     item-star .default:n   = true,
2910     item-star .value_forbidden:n = true,
2911     item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2912     item-sym* .value_required:n = true,
2913     item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2914     item-pos* .value_required:n = true,
2915     print-env .undefine:,
2916     store-env .undefine:,
2917     write-out .undefine:,
2918     unknown   .code:n      = { \l__enumext_anskey_env_unknown:n {##1} },
2919   }
2920 }

```

The *⟨keys⟩* are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `\l__enumext_anskey_env_unknown:n`.

```

2921 \cs_new_protected:Npn \l__enumext_anskey_env_unknown:n #1
2922 {

```

```

2923 \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2924 }
2925 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2926 {
2927   \tl_if_blank:nTF {#2}
2928   {
2929     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2930   }
2931   {
2932     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2933   }
2934 }

```

The function `__enumext_anskey_env_reset_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the *hook* function `__enumext_after_env:nn`.

```

2935 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2936 {
2937   \keys_define:nn { scontents / scontents }
2938   {
2939     break-col .undefine:,
2940     item-join .undefine:,
2941     item-star .undefine:,
2942     item-sym* .undefine:,
2943     item-pos* .undefine:,
2944     write-out .code:n = {
2945       \bool_set_false:N \l__scontents_storing_bool
2946       \bool_set_true:N \l__scontents_writing_bool
2947       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2948     },
2949     write-out .value_required:n = true,
2950     print-env .meta:nn = { scontents } { print-env = ##1 },
2951     print-env .default:n = true,
2952     store-env .meta:nn = { scontents } { store-env = ##1 },
2953     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2954   }
2955 }

```

The function `__enumext_rescan_anskey_env:n` will be responsible for bringing the *body* of the environment saved in the sequence `\g__scontents_name_⟨store name⟩_seq` to pass it to our *sequence* and *prop list*.

```

2956 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2957 {
2958   \group_begin:
2959   \int_set:Nn \tex_newlinechar:D { ``^^J }
2960   \__scontents_rescan_tokens:x
2961   {
2962     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2963     #1
2964   }
2965 }

```

(End of definition for *anskey** and others. This function is documented on page 14.)

`__enumext_anskey_env_exec:`

The function `__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our *keys*.

```

2966 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2967 {
2968   \__enumext_before_env:nn { anskey* }
2969   {
2970     \__enumext_anskey_env_define_keys:
2971   }

```

Now we will execute our actions after the *anskey** environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_⟨store name⟩_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```

2972   \hook_if_empty:nF {env/anskey*/after}
2973   {
2974     \hook_gremove_code:nn {env/anskey*/after} { * }
2975   }
2976   \__enumext_after_env:nn { anskey* }
2977   {

```

```

2978     \__enumext_anskey_env_save_keys:
2979     \tl_clear:N \__enumext_store_anskey_env_tl
2980     \tl_clear:N \__enumext_store_anskey_opt_tl
2981     \bool_if:NT \__enumext_check_answers_bool
2982     {
2983         \tl_gset:Ne \__enumext_store_anskey_env_tl
2984         {
2985             \seq_item:ce { g__scontents_name_ \__enumext_store_name_tl _seq } { -1 }
2986         }
2987         \regex_match:nVTF
2988         { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2989         \__enumext_store_anskey_env_tl
2990         {
2991             \msg_error:nn { enumext } { anskey-empty-arg }
2992         }
2993         {
2994             \__enumext_anskey_env_store:
2995         }
2996     }
2997     \__enumext_anskey_env_clean_vars:
2998     \__enumext_anskey_env_reset_keys:
2999 }
3000 }

```

• The use of `\hook_gremove_code:nn` is necessary here, otherwise the `{\code}` passed to `__enumext_after-env:nn{anskey*}` will be accumulated for each execution. The last function `__enumext_anskey_env_reset_keys:` is necessary so as not to hinder any `scontents` environment running within `enumext` or `enumext*`.

(End of definition for `__enumext_anskey_env_exec:.`)

```

\__enumext_anskey_env_save_keys:
\__enumext_anskey_env_store:
\__enumext_anskey_env_clean_vars:

```

The function `__enumext_anskey_env_save_keys:` processing the `[key = val]` passed to the environment and save this in the variable `__enumext_store_anskey_opt_tl`. If the `break-col` key is present and the environment is running under `enumext` (not in `enumext*`) we will add the key `break-col`.

```

3001 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
3002 {
3003     \bool_lazy_and:nnT
3004     { \bool_if_p:N \g__enumext_store_columns_break_bool }
3005     { \bool_not_p:n { \__enumext_starred_bool } }
3006     {
3007         \tl_put_left:Ne \__enumext_store_anskey_opt_tl { ,break-col, }
3008     }

```

If the `item-join` key is present and the command is running under `enumext*` we will add to `__enumext_store_anskey_opt_tl`.

```

3009     \bool_lazy_and:nnT
3010     { \bool_not_p:n { \__enumext_starred_bool } }
3011     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
3012     {
3013         \tl_put_left::Ne \__enumext_store_anskey_opt_tl
3014         {
3015             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
3016         }
3017     }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `__enumext_store_anskey_opt_tl`.

```

3018     \bool_if:NT \g__enumext_store_item_star_bool
3019     {
3020         \tl_put_left:Ne \__enumext_store_anskey_opt_tl
3021         {
3022             ,item-star,
3023         }
3024         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
3025         {
3026             \tl_put_left:Ne \__enumext_store_anskey_opt_tl
3027             {
3028                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
3029             }
3030         }
3031         \dim_compare:nT
3032         {
3033             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim

```

```

3034     }
3035     {
3036         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
3037         {
3038             ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
3039         }
3040     }
3041 }
3042 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `__enumext_store_anskey_code:n` and `__enumext_rescan_anskey_env:n`.

```

3043 \cs_new_protected:Nn \__enumext_anskey_env_store:
3044 {
3045     \group_begin:
3046     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
3047     {
3048         \exp_args:Ne
3049         \__enumext_store_anskey_code:n
3050         {
3051             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3052         }
3053     }
3054     {
3055         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
3056         \exp_args:Ne
3057         \__enumext_store_anskey_code:n
3058         {
3059             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3060         }
3061     }
3062     \group_end:
3063 }

```

The function `__enumext_anskey_env_clean_vars:` will return the global variables used by the *(keys)* to their initial state.

```

3064 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3065 {
3066     \bool_gset_false:N \g__enumext_store_columns_break_bool
3067     \int_gzero:N \g__enumext_store_item_join_int
3068     \bool_gset_false:N \g__enumext_store_item_star_bool
3069     \tl_gclear:N \g__enumext_store_item_symbol_tl
3070     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
3071 }

```

(End of definition for `__enumext_anskey_env_save_keys:`, `__enumext_anskey_env_store:`, and `__enumext_anskey_env_clean_vars:`.)

13.32 Executing anskey*, check-ans 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, undefine the environment `anskey*` (§13.31) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

3072 \cs_new_protected:Nn \__enumext_execute_after_env:
3073 {
3074     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3075     {
3076         \tl_if_empty:NF \g__enumext_store_name_tl
3077         {
3078             \__enumext_stop_save_ans_msg:
3079             \__enumext_item_answer_diff:
3080             \__enumext_log_global_vars:
3081             \__enumext_log_answer_vars:
3082             \bool_if:NTF \g__enumext_check_ans_key_bool
3083             {
3084                 \__enumext_check_ans_show:
3085             }

```

```

3086         { \__enumext_check_ans_log: }
3087         \__enumext_undefine_anskey_env:
3088     }
3089     \__enumext_reset_global_vars:
3090 }
3091 }

```

(End of definition for __enumext_execute_after_env:.)

- This function is passed to the function __enumext_after_env:nn for the environments `enumext` (§13.39) and `enumext*` (§13.44) and it is executed only when the environments are not nested or at some level of these..

13.33 Common functions for keyans, keyans* and keyanspic

13.33.1 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 $\backslash item^*$ in `keyans` environment and the current $\langle label \rangle$ for $\backslash anspic^*$ 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.

```

3092 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3093 {
3094     \tl_clear:N \__enumext_store_current_label_tl
3095     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
3096     {
3097         \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
3098     }
3099     {
3100         \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
3101     }

```

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

```

3102     \tl_if_novalue:nF { #1 }
3103     {
3104         \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_v_tl
3105         {
3106             \tl_put_right:Ne \__enumext_store_current_label_tl
3107             {
3108                 \__enumext_store_keyans_item_opt_sep_v_tl
3109             }
3110         }
3111         \tl_put_right:Ne \__enumext_store_current_label_tl { #1 }
3112     }
3113     \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
3114 }

```

(End of definition for __enumext_keyans_addto_prop:n.)

13.33.2 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 $\backslash item^*$ and $\backslash anspic^*$ with the $\langle contents \rangle$ of the *optional argument*. The mechanism defined here will allow to execute $\backslash ref\{ \langle store name \rangle : \langle position \rangle \}$ 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 $\backslash item^*$ and $\backslash anspic^*$ 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.

```

3115 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3116 {
3117     \bool_if:NT \__enumext_store_ref_key_bool
3118     {
3119         \cs_set_protected:Npn \__enumext_tmp:n ##1
3120         {
3121             \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3122             \tl_reverse:c { \__enumext_label_copy_##1_tl }
3123             \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3124             \tl_reverse:c { \__enumext_label_copy_##1_tl }
3125         }
3126         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3127         \__enumext_keyans_store_ref_aux_i:
3128     }
3129 }

```

The auxiliary function `__enumext_keyans_store_ref_aux_i`: set the variable `\l__enumext_newlabel_arg_one_tl` which will contain $\langle store\ name : position \rangle$ analyzing whether the environment in which they are executed is `enumext*` or `enumext`.

```

3130 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3131 {
3132   \bool_if:NT \g__enumext_starred_bool
3133   {
3134     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3135   }
3136   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3137   {
3138     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3139     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3140   }
3141   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3142   {
3143     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3144     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3145   }
3146   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3147   {
3148     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3149     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3150   }
3151   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3152   {
3153     \l__enumext_store_name_tl \c_colon_str
3154     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3155   }
3156   \__enumext_keyans_store_ref_aux_ii:
3157 }

```

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.

```

3158 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3159 {
3160   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3161   {
3162     \__enumext_newlabel:nn
3163     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3164     { \l__enumext_newlabel_arg_two_tl }
3165   }
3166   \l__enumext_write_aux_file_tl
3167 }

```

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

13.33.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

The function `__enumext_keyans_addto_seq:n` will pass the contents of the current $\langle label \rangle$ `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3168 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3169 {
3170   \tl_clear:N \l__enumext_store_current_label_tl
3171   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3172   {
3173     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3174   }
3175   {
3176     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3177   }
3178   \tl_if_novalue:nF { #1 }
3179   {
3180     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3181     {
3182       \tl_put_right:Ne \l__enumext_store_current_label_tl
3183       {
3184         \l__enumext_store_keyans_item_opt_sep_v_tl

```

```

3185         }
3186     }
3187     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3188 }
3189 \__enumext_keyans_addto_seq_link:
3190 }

```

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.

```

3191 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3192 {
3193     \bool_lazy_and:nnT
3194     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3195     { \bool_if_p:N \l__enumext_hyperref_bool }
3196     {
3197         \tl_put_right:Ne \l__enumext_store_current_label_tl
3198         {
3199             \hfill \exp_not:N \hyperlink
3200             {
3201                 \exp_not:V \l__enumext_newlabel_arg_one_tl
3202             }
3203             { \exp_not:V \l__enumext_mark_ref_sym_tl }
3204         }
3205     }
3206     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3207     \bool_if:NT \l__enumext_check_answers_bool
3208     {
3209         \int_gincr:N \g__enumext_item_anskey_int
3210     }
3211 }

```

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

13.33.4 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:

```

```

3212 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3213 {
3214     \tl_if_novalue:nF { #1 }
3215     {
3216         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3217     }
3218 }

```

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.

```

3219 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3220 {
3221     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3222     {
3223         \bool_lazy_or:nnT
3224         { \bool_if_p:N \l__enumext_show_answer_bool }
3225         { \bool_if_p:N \l__enumext_show_position_bool }
3226         {
3227             \__enumext_keyans_wrapper_opt_v:n
3228             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3229         }
3230     }
3231 }

```

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.

```

3232 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3233 {
3234     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3235     {
3236         \bool_lazy_or:nnT

```



```

3237         { \bool_if_p:N \l__enumext_show_answer_bool }
3238         { \bool_if_p:N \l__enumext_show_position_bool }
3239         {
3240             \__enumext_keyans_wrapper_opt_viii:n
3241             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3242         }
3243     }
3244 }

```

(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 $\langle label \rangle$.

```

3245 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3246 {
3247     \__enumext_label_width_by_box:Nn
3248     \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3249     \str_case:Vn \l__enumext_align_label_pos_v_str
3250     {
3251         { l }
3252         {
3253             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3254         }
3255         { r }
3256         {
3257             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3258             { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3259         }
3260         { c }
3261         {
3262             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3263             { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3264         }
3265     }

```

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

```

3266     \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3267     {
3268         \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3269     }
3270     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3271     \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3272     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3273 }

```

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

```

3274 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3275 {
3276     \bool_lazy_all:nT
3277     {
3278         { \bool_if_p:N \l__enumext_show_answer_bool }
3279         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3280     }
3281     {
3282         \__enumext_keyans_pos_mark_set:
3283         \__enumext_print_keyans_box:NN
3284         \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3285     }
3286 }

```

The function `__enumext_keyans_show_pos:` will print the $\langle position \rangle$ of the stored content in *prop list*. Need add `1` to `\g__enumext_<store name>_prop` for `keyans` environment.

```

3287 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3288 {
3289     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3290     {
3291         \int_incr:N \l__enumext_show_pos_tmp_int
3292     }
3293     {
3294         \int_zero:N \l__enumext_show_pos_tmp_int

```

```

3295     }
3296     \bool_lazy_all:nT
3297     {
3298         { \bool_if_p:N \l__enumext_show_position_bool }
3299         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3300     }
3301     {
3302         \tl_set:Nx \l__enumext_mark_answer_sym_v_tl
3303         {
3304             \group_begin:
3305             \exp_not:N \normalfont
3306             \exp_not:N \footnotesize [ \int_eval:n
3307                 {
3308                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3309                     + \l__enumext_show_pos_tmp_int
3310                 }
3311             ]
3312             \group_end:
3313         }
3314         \__enumext_keyans_pos_mark_set:
3315         \__enumext_print_keyans_box:NN
3316         \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3317     }
3318 }

```

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

13.34 Redefining `\item` and `\makeLabel` in `enumext`

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makeLabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

When *labeling* PDF is active `\makeLabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is to redefine `\makeLabel` using `\makebox`. The best way to implement this is to use the conditional command `\IfDocumentMetadataTF` to force this redefinition and the dedicated `mode-box` key to manually activate it by the user.

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext` and we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

```
\__enumext_default_item:n
```

First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```

3319 \cs_new_protected:Npn \__enumext_default_item:n #1
3320 {
3321     \tl_if_novalue:nTF {#1}
3322     {
3323         \bool_if:NT \l__enumext_check_answers_bool
3324         {
3325             \int_gincr:N \g__enumext_item_number_int
3326             \bool_set_true:N \l__enumext_item_number_bool
3327         }
3328         \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3329         \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3330     }
3331     {
3332         \bool_set_eq:cc
3333         { l__enumext_wrap_label_ \__enumext_level: _bool }
3334         { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3335         \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3336     }
3337 }

```

(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`.

```
3338 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3339 {
3340   \tl_if_novalue:nTF {#1}
3341   {
3342     \tl_gset_eq:Nc
3343     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3344   }
3345   {
3346     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3347   }
3348   \tl_if_novalue:nTF {#2}
3349   {
3350     \dim_set_eq:cc
3351     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3352     { \l__enumext_labelsep_ \__enumext_level: _dim }
3353   }
3354   {
3355     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3356   }
3357   \bool_if:NT \l__enumext_check_answers_bool
3358   {
3359     \int_gincr:N \g__enumext_item_number_int
3360     \bool_set_true:N \l__enumext_item_number_bool
3361   }
3362   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3363   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3364 }
```

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

```
3365 \cs_new_protected:Nn \__enumext_item_starred_exec:
3366 {
3367   \tl_if_empty:cF { \l__enumext_item_symbol_ \__enumext_level: _tl }
3368   {
3369     \mode_leave_vertical:
3370     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3371     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3372     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3373   }
3374 }
```

(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 (§13.39).

```
3375 \cs_new_protected:Nn \__enumext_redefine_item:
3376 {
3377   \RenewDocumentCommand \item { s o o }
3378   {
3379     \bool_if:nTF {##1}
3380     {
3381       \__enumext_item_starred_exec:nn {##2} {##3}
3382     }
3383     { \__enumext_default_item:n {##2} }
3384   }
3385 }
```

(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 (§13.39).

```

3386 \cs_new_protected:Nn \__enumext_make_label:
3387 {
3388   \IfDocumentMetadataTF
3389   {
3390     \__enumext_make_label_box:
3391   }
3392   {
3393     \bool_if:NTF \l__enumext_mode_box_bool
3394     {
3395       \__enumext_make_label_box:
3396     }
3397     {
3398       \__enumext_make_label_std:
3399     }
3400   }
3401 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3402 \cs_new_protected:Nn \__enumext_make_label_std:
3403 {
3404   \RenewDocumentCommand \makeLabel { m }
3405   {
3406     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3407     \__enumext_item_starred_exec:
3408     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3409     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3410     {
3411       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3412     }
3413     { ##1 }
3414     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3415     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3416   }
3417 }

```

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`.

```

3418 \cs_new_protected:Nn \__enumext_make_label_box:
3419 {
3420   \RenewDocumentCommand \makeLabel { m }
3421   {
3422     \strut\smash
3423     {
3424       \makebox
3425       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3426       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3427       {
3428         \__enumext_item_starred_exec:
3429         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3430         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3431         {
3432           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3433         }
3434         { ##1 }
3435         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3436       }
3437     } % close smash
3438   }
3439 }

```

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

13.35 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*
3440 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3441 {
3442   \keys_define:nn { enumext / #1 }
3443   {
3444     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3445     item-sym* .value_required:n = true,
3446     item-sym* .initial:n = {\textborn},
3447     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3448     item-pos* .value_required:n = true,
3449   }
3450 }
3451 \clist_map_inline:nn
3452 {
3453   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3454 }
3455 { \__enumext_tmp:nn #1 }

```

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

13.36 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.

13.36.1 Handling unknown keys for keyans, keyans* and keyanspic

unknown 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.

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3456 \cs_set_protected:Npn \__enumext_tmp:n #1
3457 {
3458   \keys_define:nn { enumext / #1 }
3459   {
3460     unknown .code:n = {
3461       \tl_set:Nn \l__enumext_envir_name_tl {#1}
3462       \__enumext_keyans_unknown_keys:n {##1}
3463     },
3464   }
3465 }
3466 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3467 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3468 {
3469   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3470 }
3471 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3472 {
3473   \tl_if_blank:nTF {#2}
3474   {
3475     \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3476   }
3477   {
3478     \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3479   }
3480 }

```

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

13.36.2 Handling unknown keys for enumext*

```

unknown Define and set unknown key for enumext* environment.
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3481 \keys_define:nn { enumext / enumext* }
3482 {
3483   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3484 }

```

Internal functions for handling unknown key.

```

3485 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3486 {
3487   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}

```

```

3488     }
3489     \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3490     {
3491         \tl_if_blank:nTF {#2}
3492         {
3493             \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3494         }
3495         {
3496             \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3497         }
3498     }

```

(End of definition for `unknown`, `__enumext_starred_unknown_keys:n`, and `__enumext_starred_unknown_keys:nn`.)

13.36.3 Handling unknown keys for enumext

`unknown` Defines and set the key `unknown` for `enumext` environment.

```

\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
3499 \cs_set_protected:Npn \__enumext_tmp:n #1
3500 {
3501     \keys_define:nn { enumext / #1 }
3502     {
3503         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3504     }
3505 }
3506 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3507 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3508 {
3509     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3510 }
3511 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3512 {
3513     \tl_if_blank:nTF {#2}
3514     {
3515         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3516     }
3517     {
3518         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3519     }
3520 }

```

(End of definition for `unknown`, `__enumext_standar_unknown_keys:n`, and `__enumext_standar_unknown_keys:nn`.)

13.37 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`.

```

3521 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3522 {
3523     \tl_if_novalue:nTF { #1 }
3524     {
3525         \bool_set_true:N \__enumext_wrap_label_v_bool
3526         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3527     }
3528     {
3529         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3530         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3531     }
3532 }

```

(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.

```

3533 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3534 {

```

```

3535     \__enumext_keyans_save_item_opt:n { #1 }
3536     \bool_set_true:N \l__enumext_wrap_label_v_bool
3537     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3538     \__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.

```

3539     \__enumext_keyans_addto_prop:n { #1 }
3540     \__enumext_keyans_store_ref:
3541     \__enumext_keyans_addto_seq:n { #1 }
3542     \int_gincr:N \g__enumext_check_starred_cmd_int
3543 }

```

(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 (§13.38).

```

3544 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3545 {
3546   \RenewDocumentCommand \item { s o }
3547   {
3548     \bool_if:nTF {##1}
3549     {
3550       \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3551       \peek_remove_spaces:n
3552       {
3553         \__enumext_keyans_starred_item:n {##2}
3554       }
3555     }
3556     {
3557       \bool_set_false:N \l__enumext_item_wrap_key_bool
3558       \__enumext_keyans_default_item:n {##2}
3559     }
3560   }
3561 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 16.)

`__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 (§13.38).

```

3562 \cs_new_protected:Nn \__enumext_keyans_make_label:
3563 {
3564   \IfDocumentMetadataTF
3565   {
3566     \__enumext_keyans_make_label_box:
3567   }
3568   {
3569     \bool_if:NTF \l__enumext_mode_box_bool
3570     {
3571       \__enumext_keyans_make_label_box:
3572     }
3573     {
3574       \__enumext_keyans_make_label_std:
3575     }
3576   }
3577 }

```

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

```

3578 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3579 {
3580   \bool_lazy_all:nT
3581   {
3582     { \bool_if_p:N \l__enumext_wrap_label_v_bool }
3583     { \bool_if_p:N \l__enumext_show_answer_bool }

```



```

3584     { \bool_if_p:N \l__enumext_item_wrap_key_bool          }
3585     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3586   }
3587   {
3588     \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3589   }
3590   \bool_if:NTF \l__enumext_wrap_label_v_bool
3591   {
3592     \__enumext_wrapper_label_v:n { #1 }
3593   }
3594   { #1 }
3595 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3596 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3597 {
3598   \RenewDocumentCommand \makeLabel { m }
3599   {
3600     \tl_use:N \l__enumext_label_fill_left_v_tl
3601     \__enumext_keyans_show_ans:
3602     \__enumext_keyans_show_pos:
3603     \tl_use:N \l__enumext_label_font_style_v_tl
3604     \__enumext_keyans_wrapper_label:n { ##1 }
3605     \tl_use:N \l__enumext_label_fill_right_v_tl
3606   }
3607 }

```

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

```

3608 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3609 {
3610   \RenewDocumentCommand \makeLabel { m }
3611   {
3612     \strut\smash
3613     {
3614       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3615       {
3616         \__enumext_keyans_show_ans:
3617         \__enumext_keyans_show_pos:
3618         \tl_use:N \l__enumext_label_font_style_v_tl
3619         \__enumext_keyans_wrapper_label:n { ##1 }
3620       }
3621     }
3622   }
3623 }

```

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

13.38 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

13.38.1 Calculation of `\leftmargin` and `\itemindent`

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

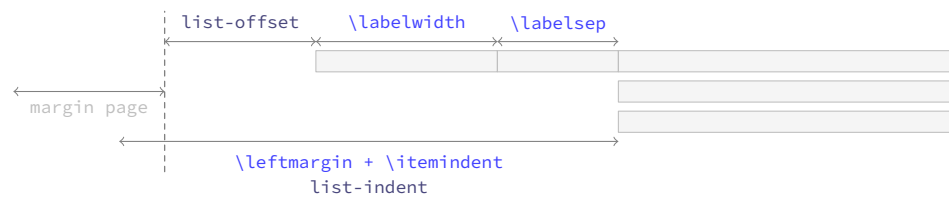


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

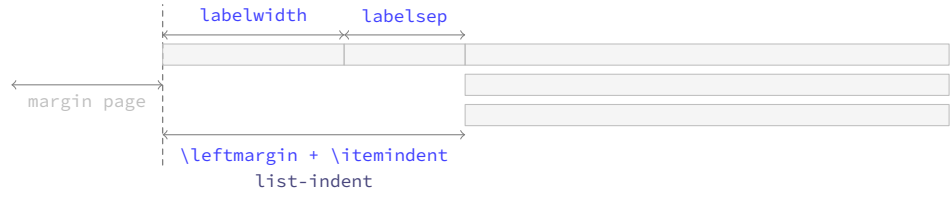
Where the default values will look like in the figure 11.

```

\__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`.

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§13.38).

```

3624 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3625 {
3626   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3627   {
3628     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3629     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3630   }
3631   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3632   {
3633     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3634     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3635   }

```

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

```

3636   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3637   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3638   {
3639     \dim_set:Nn #6 { #1 + #2 - #4 }
3640     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3641   }
3642   {
3643     \dim_compare:nNnT { #4 } = { #1 + #2 }
3644     { \dim_set:Nn #6 { \c_zero_dim } }
3645     \dim_compare:nNnT { #4 } < { #1 + #2 }
3646     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3647     \dim_compare:nNnT { #4 } > { #1 + #2 }
3648     {
3649       \dim_set:Nn #6 { -#1 - #2 + #4 }
3650       \dim_set:Nn #6 { #6*-1 }
3651     }
3652     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3653   }
3654 }
3655 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }

```

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

13.38.2 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.

```

3656 \cs_set_protected:Npn \__enumext_tmp:n #1

```

©2024 by Pablo González L

```

3657 {
3658   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3659   {
3660     \__enumext_calc_hspace:ccccc
3661     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3662     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3663     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3664     { \__enumext_leftmargin_tmp_#1_bool }
3665     \clist_map_inline:nn
3666       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3667       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3668     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3669       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3670     \usecounter { enumX#1 }
3671     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3672     \str_if_eq:nnTF {#1} { v }
3673     {
3674       \__enumext_keyans_redefine_item:
3675       \__enumext_keyans_make_label:
3676       \__enumext_keyans_ref:
3677       \__enumext_keyans_fake_item_indent:
3678       \bool_if:cT { \__enumext_show_length_#1_bool }
3679       {
3680         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3681       }
3682     }
3683     {
3684       \__enumext_redefine_item:
3685       \__enumext_make_label:
3686       \__enumext_standar_ref:
3687       \__enumext_fake_item_indent:
3688       \bool_if:cT { \__enumext_show_length_#1_bool }
3689       {
3690         \msg_term:nnne { enumext } { list-lengths } {#1}
3691         { \int_use:N \__enumext_level_int }
3692       }
3693     }
3694   }
3695 }
3696 \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 `\lisparindent` and `parsep` to set the value of `\parskip` locally.

```

3697 \cs_set_protected:Npn \__enumext_tmp:n #1
3698 {
3699   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3700   {
3701     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3702     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3703     \__enumext_calc_hspace:ccccc
3704     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3705     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3706     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3707     { \__enumext_leftmargin_tmp_#1_bool }
3708     \clist_map_inline:nn
3709       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3710       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3711     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3712       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3713     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3714     \skip_zero:N \partopsep
3715     \usecounter { enumX#1 }
3716     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3717     \__enumext_starred_ref:
3718     \str_if_eq:nnTF {#1} { vii }
3719     {
3720       \__enumext_fake_item_indent_vii:

```

```

3721         \bool_if:cT { \__enumext_show_length_vii_bool }
3722         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3723     }
3724     {
3725         \__enumext_fake_item_indent_viii:
3726         \bool_if:cT { \__enumext_show_length_#1_bool }
3727         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3728     }
3729 }
3730 }
3731 \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:.)

13.39 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”.

```

3732 \cs_new_protected:Nn \__enumext_safe_exec:
3733 {
3734     \__enumext_is_not_nested:
3735     \__enumext_internal_mini_page:
3736     \int_incr:N \l__enumext_level_int
3737     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3738     { \msg_fatal:nn { enumext } { list-too-deep } }
3739     \bool_set_true:N \l__enumext_standar_bool
3740     \bool_set_false:N \l__enumext_starred_bool
3741     \__enumext_is_on_first_level:
3742 }

```

(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_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.

```

3743 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3744 {
3745     \tl_if_novalue:nF {#1}
3746     {
3747         \str_clear:N \l__enumext_series_str
3748         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3749         {
3750             \keys_set:nn { enumext / level-1 } {#1}
3751             \__enumext_parse_series:n {#1}
3752             \__enumext_nested_base_line_fix:
3753         }
3754         {
3755             \exp_args:Ne \keys_set:nn
3756             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3757         }
3758         \__enumext_store_active_keys:n {#1}
3759     }
3760 }

```

(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*`.

```

3761 \cs_new_protected:Nn \__enumext_start_store_level:
3762 {
3763     \bool_lazy_all:nT
3764     {
3765         { \bool_if_p:N \l__enumext_store_active_bool }
3766         { \bool_not_p:n { \l__enumext_keyans_env_bool } }

```

```

3767     { \bool_if_p:N \g__enumext_standar_bool }
3768   }
3769   {
3770     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3771     {
3772       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3773       \__enumext_store_level_open:
3774     }
3775   }

```

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

```

3776   \bool_lazy_all:nT
3777   {
3778     { \bool_if_p:N \l__enumext_store_active_bool }
3779     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3780     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3781   }
3782   {
3783     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3784     {
3785       \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3786       \__enumext_store_level_open:
3787     }
3788   }
3789 }

```

(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*`.

```

3790 \cs_new_protected:Nn \__enumext_stop_store_level:
3791 {
3792   \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3793   {
3794     \__enumext_store_level_close:
3795   }
3796 }

```

(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.

```

3797 \cs_new_protected:Nn \__enumext_multicols_start:
3798 {
3799   \int_compare:nNnT
3800   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3801   {
3802     \dim_compare:nNnT
3803     { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3804     {
3805       \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
3806       {
3807         ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
3808           + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
3809         ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
3810         - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3811       }
3812     }
3813     \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
3814     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3815     {
3816       \dim_zero:N \columnseprule
3817     }
3818   }

```

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.

```

3818   \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3819   {
3820     \skip_zero:N \multicolsep
3821     \__enumext_multi_addvspace:
3822   }

```

```

3823         \raggedcolumns
3824         \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
3825     }
3826 }

```

(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:.

```

3827 \cs_new_protected:Nn \__enumext_multicols_stop:
3828 {
3829     \int_compare:nNtF
3830     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3831     {
3832         \__enumext_stop_list:
3833         \__enumext_stop_store_level:
3834         \end{multicols}
3835         \__enumext_unskip_unkern:
3836         \__enumext_unskip_unkern:
3837         \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3838     }
3839     {
3840         \__enumext_stop_list:
3841         \__enumext_stop_store_level:
3842     }
3843 }

```

(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.

```

3844 \cs_new_protected:Nn \__enumext_before_list:
3845 {
3846     \__enumext_vspace_above:
3847     \__enumext_before_args_exec:
3848     \__enumext_check_ans_active:

```

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

```

3849     \dim_compare:nNt
3850     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3851     {
3852         \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
3853         {
3854             \linewidth
3855             - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
3856             - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
3857         }

```

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

```

3858         \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
3859         \int_gincr:N \g__enumext_minipage_stat_int
3860         \__enumext_minipage_add_space:
3861         \noindent
3862         \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
3863     }
3864     \__enumext_multicols_start:
3865 }

```

(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.

```

3866 \cs_new_protected:Nn \__enumext_second_part:
3867 {
3868   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3869   {
3870     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3871     {
3872       \msg_warning:nn { enumext } { missing-miniright }
3873       \miniright
3874     }
3875     \int_gzero:N \g__enumext_minipage_stat_int
3876     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3877     \end__enumext_mini_page
3878   }
3879   {
3880     \__enumext_multicols_stop:
3881   }

```

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.

```

3882   \__enumext_after_stop_list:
3883   \__enumext_check_ans_key_hook:
3884   \__enumext_vspace_below:
3885   \bool_set_false:N \l__enumext_standar_bool
3886   \__enumext_resume_save_counter:
3887 }

```

(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.

```

3888 \cs_new_protected:Nn \__enumext_set_item_width:
3889 {
3890   \dim_set:Nn \itemwidth { \linewidth }
3891   \dim_compare:nT
3892   {
3893     \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3894   }
3895   {
3896     \dim_sub:Nn \itemwidth
3897     {
3898       \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3899     }
3900   }
3901 }

```

(End of definition for `__enumext_set_item_width:`.)

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

```

3902 \NewDocumentEnvironment{enumext}{0}{}
3903 {
3904   \__enumext_safe_exec:
3905   \__enumext_parse_keys:n {#1}
3906   \__enumext_before_list:
3907   \__enumext_start_store_level:
3908   \__enumext_start_list:nn
3909   { \tl_use:c { \l__enumext_label_ \__enumext_level: _tl } }
3910   {
3911     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3912     \__enumext_before_keys_exec:
3913   }
3914   \__enumext_set_item_width:
3915   \__enumext_after_args_exec:
3916 }
3917 {

```



```

3918     \__enumext_second_part:
3919 }

```

(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*.

```

3920 \__enumext_after_env:nn {enumext}
3921 {
3922     \__enumext_execute_after_env:
3923 }

```

13.40 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”.

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.

```

3924 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3925 {
3926     \bool_if:NF \l__enumext_store_active_bool
3927     {
3928         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3929     }
3930     \int_incr:N \l__enumext_keyans_level_int
3931     \bool_set_true:N \l__enumext_keyans_env_bool
3932     \__enumext_keyans_name_and_start:
3933     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3934     \bool_set_false:N \l__enumext_store_active_bool
3935     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3936     {
3937         \msg_error:nn { enumext } { keyans-nested }
3938     }
3939     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3940     {
3941         \msg_error:nn { enumext } { keyans-wrong-level }
3942     }
3943 }

```

(End of definition for *__enumext_keyans_safe_exec:.*)

__enumext_keyans_parse_keys:n Parse [*key = val*] for *keyans* environment.

```

3944 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3945 {
3946     \keys_set:nn { enumext / keyans } {#1}
3947 }

```

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

__enumext_before_list_v: Same implementation as the one used in the *enumext* environment.

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_second_part_v:
3948 \cs_new_protected:Nn \__enumext_before_list_v:
3949 {
3950     \__enumext_vspace_above_v:
3951     \__enumext_before_args_exec_v:
3952     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3953     {
3954         \dim_set:Nn \l__enumext_minipage_left_v_dim
3955         {
3956             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3957         }
3958         \bool_set_true:N \l__enumext_minipage_active_v_bool
3959         \int_gincr:N \g__enumext_minipage_stat_int
3960         \__enumext_keyans_minipage_add_space:
3961         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3962     }
3963     \__enumext_keyans_multicols_start:
3964 }
3965 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3966 {

```

```

3967 \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3968 {
3969   \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3970   {
3971     \dim_set:Nn \l__enumext_columns_sep_v_dim
3972     {
3973       (
3974         \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3975       ) / \l__enumext_columns_v_int
3976       - \l__enumext_listoffset_v_dim
3977     }
3978   }
3979   \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3980   \dim_zero:N \columnseprule % no rule here
3981   \bool_if:NF \l__enumext_minipage_active_v_bool
3982   {
3983     \skip_zero:N \multicolsep
3984     \__enumext_keyans_multi_addvspace:
3985   }
3986   \raggedcolumns
3987   \begin{multicols}{ \l__enumext_columns_v_int }
3988 }
3989 }
3990 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3991 {
3992   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3993   {
3994     \__enumext_stop_list:
3995     \end{multicols}
3996     \__enumext_unskip_unkern:
3997     \__enumext_unskip_unkern:
3998     \par\addvspace{ \l__enumext_multicols_below_v_skip }
3999   }
4000   {
4001     \__enumext_stop_list:
4002   }
4003 }
4004 \cs_new_protected:Nn \__enumext_second_part_v:
4005 {
4006   \bool_if:NTF \l__enumext_minipage_active_v_bool
4007   {
4008     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
4009     {
4010       \msg_warning:nn { enumext } { missing-miniright }
4011       \miniright
4012     }
4013     \int_gzero:N \g__enumext_minipage_stat_int
4014     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4015     \end__enumext_mini_page
4016     \par\addvspace{ \l__enumext_minipage_after_skip }
4017   }
4018   {
4019     \__enumext_keyans_multicols_stop:
4020   }
4021   \bool_set_false:N \l__enumext_keyans_env_bool
4022   \__enumext_after_stop_list_v:
4023   \__enumext_vspace_below_v:
4024 }

```

(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.

```

4025 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4026 {
4027   \dim_set:Nn \itemwidth { \linewidth }
4028   \dim_compare:nT
4029   {
4030     \l__enumext_listoffset_v_dim != \c_zero_dim
4031   }
4032   {

```

```

4033         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4034     }
4035 }

```

(End of definition for `__enumext_keyans_set_item_width:`)

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

```

4036 \NewDocumentEnvironment{keyans}{ 0{} }
4037 {
4038     \__enumext_keyans_safe_exec:
4039     \__enumext_keyans_parse_keys:n {#1}
4040     \__enumext_before_list_v:
4041     \__enumext_start_list:nn
4042     { \tl_use:N \l__enumext_label_v_tl }
4043     {
4044         \__enumext_list_arg_two_v:
4045         \__enumext_before_keys_exec_v:
4046     }
4047     \__enumext_keyans_set_item_width:
4048     \__enumext_after_args_exec_v:
4049 }
4050 {
4051     \__enumext_check_starred_cmd:n { item }
4052     \__enumext_second_part_v:
4053 }

```

(End of definition for `keyans`. This function is documented on page 15.)

13.41 Tagging PDF support for non-standart list environments

The \LaTeX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the 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`[17] and `ltsockets`[19]. 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](#).

13.41.1 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
4054 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4055 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {start-list-tags}
4056 {
4057     \tag_resume:n {#1}
4058     \tag_mc_end_push:
4059     \tag_struct_begin:n {tag=LI}
4060     \tag_struct_begin:n {tag=Lbl}
4061     \tag_mc_begin:n {tag=Lbl}
4062 }
4063 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4064 {
4065     \tag_mc_end:
4066     \tag_struct_end:n {tag=Lbl}
4067     \tag_struct_begin:n {tag=LBody}
4068     \tag_struct_begin:n {tag=text-unit}
4069     \tag_struct_begin:n {tag=text}
4070 }
4071 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4072 {
4073     \tag_struct_end:n {tag=text}
4074     \tag_struct_end:n {tag=text-unit}
4075     \tag_struct_end:n {tag=LBody}
4076     \tag_struct_end:n {tag=LI}
4077     \tag_mc_begin_pop:n {}
4078     \tag_suspend:n {#1}
4079 }

```

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

```

4080 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4081 {

```

```

4082     \IfDocumentMetadataTF
4083     {
4084         \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4085         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4086     } {}
4087 }
4088 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
4089 {
4090     \IfDocumentMetadataTF
4091     {
4092         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4093         \socket_use:nn {tagsupport/__enumext/starred} { }
4094     } {}
4095 }
4096 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4097 {
4098     \IfDocumentMetadataTF
4099     {
4100         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4101         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4102     } {}
4103 }

```

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

13.41.2 Socket for tagging support in keyanspic

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

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:
4104 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4105 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4106 {
4107     \tag_resume:n {keyanspic}
4108     \tag_mc_end_push:
4109         \tag_struct_begin:n {tag=LI}
4110         \tag_struct_begin:n {tag=Lbl}
4111         \tag_mc_begin:n {tag=Lbl}
4112 }
4113 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4114 {
4115     \tag_mc_end:
4116     \tag_struct_end:n {tag=Lbl}
4117     \tag_struct_begin:n {tag=LBody}
4118     \tag_struct_begin:n {tag=text-unit}
4119     \tag_struct_begin:n {tag=text}
4120     \tag_mc_begin:n {tag=text}
4121 }
4122 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4123 {
4124     \tag_mc_end:
4125     \tag_struct_end:n {tag=text}
4126     \tag_struct_end:n {tag=text-unit}
4127     \tag_struct_end:n {tag=LBody}
4128     \tag_struct_end:n {tag=LI}
4129     \tag_mc_begin_pop:n {}
4130     \tag_suspend:n {keyanspic}
4131 }

```

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

```

4132 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4133 {
4134     \IfDocumentMetadataTF
4135     {
4136         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4137         \socket_use:n {tagsupport/__enumext/keyanspic}
4138     } {}
4139 }
4140 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4141 {
4142     \IfDocumentMetadataTF
4143     {
4144         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4145         \socket_use:n {tagsupport/__enumext/keyanspic}
4146     } {}

```

```

4147     }
4148     \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4149     {
4150         \IfDocumentMetadataTF
4151         {
4152             \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4153             \socket_use:n {tagsupport/__enumext/keyanspic}
4154         } {}
4155     }

```

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

13.42 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a `list` based environment that uses the same configuration for “*spacing*” and `<label>` as the `keyans` environment, but it does not use `\item`. The `<contents>` are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the `<label>` centered “*above*” or “*below*”, adjusting *widths* and *position* according to the options passed to the environment.

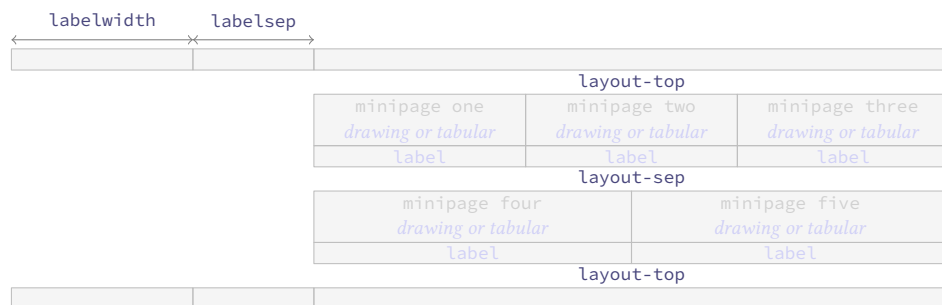


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

13.42.1 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 231 will be label-pos, the vertical separation of these from drawing or tabular will be handled with the key
layout-sty 231 label-sep. The “layout style” will be handled with the key layout-sty will take two values separated
layout-sep 231 by comma {(n° upper, n° lower)} and will determine the number of minipage environments in which all
layout-top 231 arguments of \anspic will be printed at the “upper” and “lower” within the environments separated by the
mark-ans 231 value of the key layout-sep. The vertical space “top” and “bottom” of the environment will be handled with
mark-pos 231 the key layout-top.
mark-sep 231
save-sep 231
wrap-opt 231
wrap-ans* 231
show-ans 231
show-pos 231

4156 \keys_define:nn { enumext / keyanspic }
4157 {
4158     label-pos .choice:,
4159     label-pos / above .code:n =
4160         \bool_set_true:N \l__enumext_anspic_label_above_bool
4161         \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4162     label-pos / below .code:n =
4163         \bool_set_false:N \l__enumext_anspic_label_above_bool
4164         \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4165     label-pos / unknown .code:n =
4166         \msg_error:nneee { enumext } { unknown-choice }
4167         { label-pos } { above,~ below } { \exp_not:n {#1} },
4168     label-pos .initial:n = below,
4169     label-pos .value_required:n = true,
4170     label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4171     label-sep .value_required:n = true,
4172     layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4173     layout-sty .value_required:n = true,
4174     layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4175     layout-sep .value_required:n = true,
4176     layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4177     layout-top .value_required:n = true,
4178     mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4179     mark-ans .value_required:n = true,
4180     mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4181     mark-pos .value_required:n = true,

```

```

4182 mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4183 mark-sep .value_required:n = true,
4184 save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4185 save-sep .value_required:n = true,
4186 wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4187 wrap-opt .value_required:n = true,
4188 wrap-ans* .code:n = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4189 wrap-ans* .value_required:n = true,
4190 show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4191 show-ans .value_required:n = true,
4192 show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4193 show-pos .value_required:n = true,
4194 unknown .code:n = {
4195     \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4196     \__enumext_keyans_unknown_keys:n {#1}
4197 },
4198 }

```

(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.

```

4199 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4200 {
4201     \int_incr:N \l__enumext_keyans_pic_level_int
4202     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4203     {
4204         \msg_error:nn { enumext } { keyanspic-nested }
4205     }
4206     \__enumext_keyans_name_and_start:
4207 }

```

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

```

4208 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4209 {
4210     \tl_if_novalue:nF {#1}
4211     {
4212         \keys_set:nn { enumext / keyanspic } {#1}
4213     }
4214 }

```

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

```

4215 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4216 {
4217     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4218     {
4219         \skip_set:Nn #1 { -#1 }
4220     }
4221 }

```

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.

```

4222 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4223 {
4224     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4225     \__enumext_list_arg_two_v:
4226     \__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*.

```

4227     \bool_if:NF \l__enumext_anspic_label_above_bool
4228     {
4229         \stepcounter { enumXv }
4230         \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4231         \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4232         {

```

```

4233         \box_ht_plus_dp:N \l__enumext_anspic_label_box
4234     }
4235     \skip_add:Nn \parsep
4236     {
4237         \l__enumext_anspic_label_htdp_dim
4238         + \box_dp:N \strutbox
4239         + \l__enumext_anspic_label_sep_skip
4240     }
4241 }

```

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.

```

4242     \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4243     \ignorespaces
4244     \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4245     \dim_zero:N \listparindent
4246     \skip_zero:N \partopsep
4247     \skip_zero:N \itemsep
4248 }

```

(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.

```

4249 \NewDocumentEnvironment{keyanspic}{ o }
4250 {
4251     \__enumext_keyans_pic_safe_exec:
4252     \__enumext_keyans_pic_parse_keys:n {#1}
4253     \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4254     \IfDocumentMetadataTF
4255     {
4256         \tag_suspend:n {list}
4257     }{}
4258     \item[] \scan_stop:
4259     \RenewDocumentCommand \item {}
4260     {
4261         \msg_error:nn { enumext } { keyanspic-item-cmd }
4262     }
4263     \IfDocumentMetadataTF
4264     {
4265         \tag_resume:n {keyanspic}
4266         \tag_tool:n {para/tagging=false}
4267         \tag_suspend:n {keyanspic}
4268     } { }
4269 }

```

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.

```

4270 {
4271     \IfDocumentMetadataTF
4272     {
4273         \tag_resume:n {keyanspic}
4274         \tag_mc_end_push:
4275         \tag_struct_begin:n {tag=L,attribute=enumerate}
4276     } { }
4277     \__enumext_anspic_exec:
4278     \IfDocumentMetadataTF
4279     {
4280         \tag_suspend:n {keyanspic}
4281     } { }
4282     \end{list}
4283     \IfDocumentMetadataTF
4284     {
4285         \tag_struct_end:n {tag=L}
4286         \tag_mc_begin_pop:n {}
4287         \tag_struct_end:n {tag=L}
4288         \tag_mc_begin_pop:n {}
4289     } { }

```


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

```

4290 \__enumext_check_starred_cmd:n { anspic }
4291 \setcounter { enumXvi } { 0 }
4292 \bool_if:NTF \l__enumext_anspic_label_above_bool
4293 {
4294   \par\addvspace{ 0.5\box_dp:N \strutbox }
4295 }
4296 {
4297   \par
4298   \addvspace
4299   {
4300     \dim_eval:n
4301     {
4302       \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4303       + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4304     }
4305   }
4306 }
4307 }

```

(End of definition for `keyanspic`. This function is documented on page 16.)

13.42.2 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.

```

4308 \NewDocumentCommand \anspic { s o +m }
4309 {
4310   \bool_if:NF \l__enumext_store_active_bool
4311   {
4312     \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4313   }
4314   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4315   {
4316     \msg_error:nn { enumext } { keyanspic-wrong-level }
4317   }
4318   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4319   {
4320     \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4321   }
4322   \seq_put_right:Nn \l__enumext_anspic_args_seq
4323   {
4324     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4325   }
4326 }

```

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”.

```

4327 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4328 {
4329   \bool_if:NF \l__enumext_anspic_label_above_bool
4330   {
4331     \IfDocumentMetadataTF
4332     {
4333       \tag_suspend:n {keyanspic}
4334     } { }
4335     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4336     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4337     {
4338       \box_ht_plus_dp:N \l__enumext_anspic_body_box

```

```

4339     }
4340     \IfDocumentMetadataTF
4341     {
4342         \tag_resume:n {keyanspic}
4343     } { }
4344 }
4345 }

```

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.

```

4346 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4347 {
4348     \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4349     {
4350         \bool_if:nTF { #1 }
4351         {
4352             \bool_set_true:N \l__enumext_item_wrap_key_bool
4353             \bool_set_true:N \l__enumext_wrap_label_v_bool
4354             \__enumext_keyans_save_item_opt:n { #2 }
4355             \__enumext_keyans_addto_prop:n { #2 }
4356             \__enumext_keyans_store_ref:
4357             \__enumext_keyans_addto_seq:n { #2 }
4358             \int_gincr:N \g__enumext_check_starred_cmd_int
4359             \__enumext_keyans_show_ans:
4360             \__enumext_keyans_show_pos:
4361             \makebox[ \l__enumext_labelwidth_v_dim ][ c ]
4362             {
4363                 \tl_use:N \l__enumext_label_font_style_v_tl
4364                 \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4365             }
4366             \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4367             \__enumext_keyans_show_item_opt:
4368         }
4369         {
4370             \bool_set_false:N \l__enumext_item_wrap_key_bool
4371             \tl_use:N \l__enumext_label_font_style_v_tl
4372             \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4373         }
4374     }
4375 }

```

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.

```

4376 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4377 {
4378     \stepcounter { enumXvi }
4379     \__enumext_anspic_body_dim:n { #3 }
4380     \bool_if:NTF \l__enumext_anspic_label_above_bool
4381     {
4382         \__enumext_anspic_label:nn { #1 } { #2 }
4383     }
4384     {
4385         \raisebox
4386         {
4387             -\dim_eval:n
4388             {
4389                 \l__enumext_anspic_label_htdp_dim
4390                 + \l__enumext_anspic_body_htdp_dim
4391                 + \box_dp:N \strutbox
4392                 + \l__enumext_anspic_label_sep_skip
4393             }
4394         }
4395         [ opt ] [ opt ]
4396         {
4397             \__enumext_anspic_label:nn { #1 } { #2 }
4398         }
4399     }
4400 }
4401 %

```

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.

```

4402 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4403 {
4404   \__enumext_anspic_start_list_tag:
4405   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4406   \__enumext_anspic_stop_start_list_tag:
4407   \bool_if:NTF \l__enumext_anspic_label_above_bool
4408   {
4409     \[\l__enumext_anspic_label_sep_skip] #3
4410   }
4411   {
4412     \[ #3
4413   }
4414   \__enumext_anspic_stop_list_tag:
4415 }

```

The value $\langle n^{\circ} upper, n^{\circ} lower \rangle$ passed to the `layout-sty` key is split by comma and is handled directly by the function `__enumext_anspic_print:n` and passed to the function `__enumext_anspic_row:n`.

```

4416 \cs_new_protected:Nn \__enumext_anspic_print:n
4417 {
4418   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4419 }
4420 \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.

```

4421 \cs_new_protected:Nn \__enumext_anspic_row:n
4422 {
4423   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4424   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4425   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4426   \int_step_inline:nnn
4427   { \l__enumext_anspic_above_int + 1 }
4428   { \l__enumext_anspic_below_int }
4429   {
4430     \IfDocumentMetadataTF
4431     {
4432       \tag_suspend:n {minipage}
4433     } { }
4434     \begin{minipage}[ \l__enumext_anspic_mini_pos_str ][ \l__enumext_anspic_mini_width_dim ]
4435       \centering
4436       \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4437     \end{minipage}
4438     \IfDocumentMetadataTF
4439     {
4440       \tag_resume:n {minipage}
4441     } { }
4442   }
4443   \par
4444 }

```

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*.

```

4445 \cs_new_protected:Nn \__enumext_anspic_exec:
4446 {
4447   \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4448   {
4449     \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4450   }
4451   {
4452     \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4453   }
4454 }

```

(End of definition for `\anspic` and others. This function is documented on page 17.)

13.43 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard \LaTeX list environments. The fundamental part of the code is adapted from the `shortlist` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makeLabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the $\langle content \rangle$ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`, we will also add the *optional argument* ($\langle number \rangle$) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* ($\langle number \rangle$).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by \LaTeX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.43.1 Functions for item box width

We set the default value for the *width of the box* containing the $\langle content \rangle$ of the items for `enumext*` environment.

```
\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
4455 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4456 {
4457   \dim_compare:nNt { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4458   {
4459     \dim_set:Nn \__enumext_columns_sep_vii_dim
4460     {
4461       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4462       / \__enumext_columns_vii_int
4463     }
4464   }
4465   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4466   \dim_set:Nn \__enumext_item_width_vii_dim
4467   {
4468     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4469     / \__enumext_columns_vii_int
4470     - \__enumext_labelwidth_vii_dim
4471     - \__enumext_labelsep_vii_dim
4472   }
```

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

```
4473   \dim_compare:nNt { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4474   {
4475     \dim_sub:Nn \__enumext_item_width_vii_dim
4476     {
4477       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4478       / \__enumext_columns_vii_int
4479     }
4480     \dim_add:Nn \__enumext_columns_sep_vii_dim
4481     {
4482       \__enumext_rightmargin_vii_dim
4483     }
4484   }
4485 }
```

Same implementation for the `keyans*` environment.

```
4486 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4487 {
4488   \dim_compare:nNt { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4489   {
4490     \dim_set:Nn \__enumext_columns_sep_viii_dim
4491     {
4492       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
4493       / \__enumext_columns_viii_int
4494     }
4495   }
```

```

4496 \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4497 \dim_set:Nn \l__enumext_item_width_viii_dim
4498 {
4499   ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4500   / \l__enumext_columns_viii_int
4501   - \l__enumext_labelwidth_viii_dim
4502   - \l__enumext_labelsep_viii_dim
4503 }
4504 \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4505 {
4506   \dim_sub:Nn \l__enumext_item_width_viii_dim
4507   {
4508     ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4509     / \l__enumext_columns_viii_int
4510   }
4511   \dim_add:Nn \l__enumext_columns_sep_viii_dim
4512   {
4513     \l__enumext_rightmargin_viii_dim
4514   }
4515 }
4516 }

```

(End of definition for `\l__enumext_starred_columns_set_vii:` and `\l__enumext_starred_columns_set_viii:`)

13.43.2 Functions for join item columns

`\l__enumext_starred_joined_item_vii:n`
`\l__enumext_starred_joined_item_viii:n`

The functions `\l__enumext_starred_joined_item_vii:n` and `\l__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.

```

4517 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
4518 {
4519   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4520   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4521   {
4522     \msg_warning:nnee { enumext } { item-joined }
4523     { \int_use:N \l__enumext_joined_item_vii_int }
4524     { \int_use:N \l__enumext_columns_vii_int }
4525     \int_set:Nn \l__enumext_joined_item_vii_int
4526     {
4527       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4528     }
4529   }
4530   \int_compare:nNnT
4531   { \l__enumext_joined_item_vii_int }
4532   >
4533   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4534   {
4535     \msg_warning:nnee { enumext } { item-joined-columns }
4536     { \int_use:N \l__enumext_joined_item_vii_int }
4537     {
4538       \int_eval:n
4539       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4540     }
4541     \int_set:Nn \l__enumext_joined_item_vii_int
4542     {
4543       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4544     }
4545   }
4546   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4547   {
4548     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4549     \int_decr:N \l__enumext_joined_item_aux_vii_int
4550     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4551     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4552     \dim_set:Nn \l__enumext_joined_width_vii_dim
4553     {
4554       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4555       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4556         + \l__enumext_columns_sep_vii_dim
4557         ) * \l__enumext_joined_item_aux_vii_int
4558     }
4559     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim

```

```

4560     }
4561     {
4562         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4563         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4564     }
4565 }

```

Same implementation for the **keyans*** environment.

```

4566 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4567 {
4568     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4569     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4570     {
4571         \msg_warning:nnee { enumext } { item-joined }
4572         { \int_use:N \l__enumext_joined_item_viii_int }
4573         { \int_use:N \l__enumext_columns_viii_int }
4574         \int_set:Nn \l__enumext_joined_item_viii_int
4575         {
4576             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4577         }
4578     }
4579     \int_compare:nNnT
4580     { \l__enumext_joined_item_viii_int }
4581     >
4582     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4583     {
4584         \msg_warning:nnee { enumext } { item-joined-columns }
4585         { \int_use:N \l__enumext_joined_item_viii_int }
4586         {
4587             \int_eval:n
4588             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4589         }
4590         \int_set:Nn \l__enumext_joined_item_viii_int
4591         {
4592             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4593         }
4594     }
4595     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4596     {
4597         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4598         \int_decr:N \l__enumext_joined_item_aux_viii_int
4599         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4600         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4601         \dim_set:Nn \l__enumext_joined_width_viii_dim
4602         {
4603             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4604             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4605               + \l__enumext_columns_sep_viii_dim
4606             ) * \l__enumext_joined_item_aux_viii_int
4607         }
4608         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4609     }
4610     {
4611         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4612         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4613     }
4614 }

```

(End of definition for `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n`)

13.43.3 Functions for mini-env, mini-right and mini-right* keys

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

```

4615 \cs_new_protected:Nn \__enumext_start_mini_vii:
4616 {
4617     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4618     {
4619         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4620         {

```

```

4621         \linewidth
4622         - \l__enumext_minipage_right_vii_dim
4623         - \l__enumext_minipage_hsep_vii_dim
4624     }
4625     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4626     \dim_gset_eq:NN
4627         \g__enumext_minipage_right_vii_dim
4628         \l__enumext_minipage_right_vii_dim
4629     \__enumext_mini_addvspace_vii:
4630     \nointerlineskip\noindent
4631     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4632 }
4633 }

```

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:nn` 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 (§13.44).

```

4634 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4635 {
4636     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4637     {
4638         \__enumext_stop_list:
4639         \__enumext_stop_store_level_vii:
4640         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4641         \end__enumext_mini_page
4642         \hfill
4643         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4644     }
4645     {
4646         \__enumext_stop_list:
4647         \__enumext_stop_store_level_vii:
4648     }
4649 }

```

(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.

```

4650 \__enumext_after_env:nn {enumext*}
4651 {
4652     \bool_if:NT \g__enumext_minipage_active_vii_bool
4653     {
4654         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4655         \legacy_if_gset_false:n { @minipage }
4656         \skip_vertical:N \c_zero_skip
4657         \par\addvspace { \g__enumext_minipage_right_skip }
4658         \bool_if:NF \g__enumext_minipage_center_vii_bool
4659         {
4660             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4661             {
4662                 \centering
4663             }
4664         }
4665         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4666         {
4667             \tl_use:N \g__enumext_miniright_code_vii_tl
4668         }
4669         \box_use_drop:N \l__enumext_miniright_code_vii_box
4670         \skip_vertical:N \c_zero_skip
4671         \__enumext_endminipage:
4672         \par\addvspace{ \g__enumext_minipage_after_skip }
4673     }
4674     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4675     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4676     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4677     \dim_gzero:N \g__enumext_minipage_right_vii_dim

```



```

4678   \bool_gset_false:N \g__enumext_starred_bool
4679 }

```

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

`__enumext_stop_mini_viii:`

```

4680 \cs_new_protected:Nn \__enumext_start_mini_viii:
4681 {
4682   \dim_compare:nNtT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4683   {
4684     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4685     {
4686       \linewidth
4687       - \l__enumext_minipage_right_viii_dim
4688       - \l__enumext_minipage_hsep_viii_dim
4689     }
4690     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4691     \dim_gset_eq:NN
4692     \g__enumext_minipage_right_viii_dim
4693     \l__enumext_minipage_right_viii_dim
4694     \__enumext_mini_addvspace_viii:
4695     \nointerlineskip\noindent
4696     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4697   }
4698 }
4699 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4700 {
4701   \bool_if:NTF \l__enumext_minipage_active_viii_bool
4702   {
4703     \__enumext_stop_list:
4704     \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4705     \end__enumext_mini_page
4706     \hfill
4707     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4708   }
4709   {
4710     \__enumext_stop_list:
4711   }
4712 }
4713 \__enumext_after_env:nn {keyans*}
4714 {
4715   \bool_if:NT \g__enumext_minipage_active_viii_bool
4716   {
4717     \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4718     \par\addvspace { \g__enumext_minipage_right_skip }
4719     \bool_if:NF \g__enumext_minipage_center_viii_bool
4720     {
4721       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4722       {
4723         \centering
4724       }
4725     }
4726     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4727     {
4728       \tl_use:N \g__enumext_miniright_code_viii_tl
4729     }
4730     \box_use_drop:N \l__enumext_miniright_code_viii_box
4731     \end__enumext_mini_page
4732     \par\addvspace{ \g__enumext_minipage_after_skip }
4733   }
4734   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4735   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4736   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4737   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4738 }

```

(End of definition for `__enumext_start_mini_viii:` and `__enumext_stop_mini_viii:`)

13.44 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 pdf \TeX and Lua \TeX .

```

4739 \NewDocumentEnvironment{enumext*}{ o }
4740 {
4741   \__enumext_safe_exec_vii:
4742   \__enumext_parse_keys_vii:n {#1}
4743   \__enumext_before_list_vii:
4744   \__enumext_start_store_level_vii:
4745   \__enumext_start_list:nn { }
4746   {
4747     \__enumext_list_arg_two_vii:
4748     \__enumext_before_keys_exec_vii:
4749   }
4750   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4751   \__enumext_starred_columns_set_vii:
4752   \item[] \scan_stop:
4753   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4754   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4755   \ignorespaces
4756 }
4757 {
4758   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4759   \__enumext_stop_item_tmp_vii:
4760   \__enumext_remove_extra_parsep_vii:
4761   \__enumext_after_list_vii:
4762 }

```

(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.

```

4763 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4764 {
4765   \__enumext_is_not_nested:
4766   \__enumext_internal_mini_page:
4767   \int_incr:N \l__enumext_level_h_int
4768   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4769   {
4770     \msg_error:nn { enumext } { nested }
4771   }
4772   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4773   {
4774     \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4775   }
4776   \bool_set_true:N \l__enumext_starred_bool
4777   \bool_set_false:N \l__enumext_standar_bool
4778   \__enumext_is_on_first_level:
4779 }

```

(End of definition for `__enumext_safe_exec_vii:`.)

`__enumext_parse_keys_vii:n`

First we will clear the variable `\l__enumext_series_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.

```

4780 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4781 {
4782   \tl_if_novalue:nF {#1}
4783   {
4784     \str_clear:N \l__enumext_series_str
4785     \keys_set:nn { enumext / enumext* } {#1}
4786     \__enumext_parse_series:n {#1}
4787     \__enumext_store_active_keys_vii:n {#1}
4788   }
4789 }

```

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

`__enumext_before_list_vii:` The function `__enumext_before_list_vii:` first calls the function `__enumext_vspace_above_vii:` used by the keys `above` and `above*`, then calls the function `__enumext_check_ans_active:` for the check answer mechanism and finally calls the functions `__enumext_before_args_exec_vii:` and `__enumext_start_mini_vii:` used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```
4790 \cs_new_protected:Nn \__enumext_before_list_vii:
4791 {
4792   \__enumext_vspace_above_vii:
4793   \__enumext_check_ans_active:
4794   \__enumext_before_args_exec_vii:
4795   \__enumext_start_mini_vii:
4796 }
```

(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:` (§13.43.3) 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.

```
4797 \cs_new_protected:Nn \__enumext_after_list_vii:
4798 {
4799   \__enumext_stop_mini_vii:
4800   \__enumext_after_stop_list_vii:
4801   \__enumext_check_ans_key_hook:
4802   \__enumext_vspace_below_vii:
4803   \bool_set_false:N \__enumext_starred_bool
4804   \__enumext_resume_save_counter:
4805 }
```

(End of definition for `__enumext_after_list_vii:`.)

`__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`.

```
4806 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4807 {
4808   \bool_if:NT \__enumext_store_active_bool
4809   {
4810     \int_compare:nNt { \__enumext_level_int } > { 0 }
4811     {
4812       \__enumext_store_level_open_vii:
4813     }
4814   }
4815 }
4816 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4817 {
4818   \bool_if:NT \__enumext_store_active_bool
4819   {
4820     \int_compare:nNt { \__enumext_level_int } > { 0 }
4821     {
4822       \__enumext_store_level_close_vii:
4823     }
4824   }
4825 }
```

(End of definition for `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:`.)

13.44.1 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.

```
4826 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4827 {
4828   \skip_horizontal:n
4829   {
4830     -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim
```

```

4831     }
4832     \ignorespaces
4833 }

```

(End of definition for `__enumext_first_item_tmp_vii:`)

```

\__enumext_start_item_tmp_vii:
\__enumext_item_peek_args_vii:
\__enumext_joined_item_vii:w
\__enumext_standar_item_vii:w
\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w

```

First we will call the function `__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```

4834 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4835 {
4836     \__enumext_stop_item_tmp_vii:
4837     \int_incr:N \l__enumext_item_column_pos_vii_int
4838     \int_gincr:N \g__enumext_item_count_all_vii_int
4839     \__enumext_item_peek_args_vii:
4840 }

```

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).

```

4841 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4842 {
4843     \peek_meaning:NTF (
4844     { \__enumext_joined_item_vii:w }
4845     { \__enumext_joined_item_vii:w (1) }
4846 }

```

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`.

```

4847 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4848 {
4849     \__enumext_starred_joined_item_vii:n {#1}
4850     \peek_meaning_remove:NTF *
4851     { \__enumext_starred_item_vii:w }
4852     { \__enumext_standar_item_vii:w }
4853 }

```

The function `__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_vii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_vii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[<custom>]` by means of the function `__enumext_start_item_vii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_vii:w [\l__enumext_label_vii_tl]`.

```

4854 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4855 {
4856     \bool_set_false:N \l__enumext_item_starred_vii_bool
4857     \peek_meaning:NTF [
4858     {
4859         \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4860         \__enumext_start_item_vii:w
4861     }
4862     {
4863         \bool_set_true:N \l__enumext_wrap_label_vii_bool
4864         \legacy_if_set_true:n { @noitemarg }
4865         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4866     }
4867 }

```

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>]`.

```

4868 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4869 {
4870     \bool_set_true:N \l__enumext_item_starred_vii_bool
4871     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4872     \peek_meaning:NTF [
4873     { \__enumext_starred_item_vii_aux_i:w }

```

```

4874     { \__enumext_starred_item_vii_aux_ii:w }
4875   }
4876 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4877 {
4878   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4879   \__enumext_starred_item_vii_aux_ii:w
4880 }
4881 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4882 {
4883   \peek_meaning:NTF [
4884     { \__enumext_starred_item_vii_aux_iii:w }
4885     {
4886       \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4887       \legacy_if_set_true:n { @noitemarg }
4888       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4889     }
4890   ]
4891 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4892 {
4893   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4894   \legacy_if_set_true:n { @noitemarg }
4895   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4896 }

```

(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 `lua1latex`

```

4897 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4898 {
4899   \legacy_if:nT { @noitemarg }
4900   {
4901     \legacy_if_set_false:n { @noitemarg }
4902     \legacy_if:nT { @nmbrrlist }
4903     {
4904       \IfDocumentMetadataTF
4905       {
4906         \bool_if:NT \l__enumext_hyperref_bool
4907         {
4908           \legacy_if_set_true:n { @hyper@item }
4909         }
4910       } { }
4911       \refstepcounter{enumXvii}
4912       \bool_if:NT \l__enumext_check_answers_bool
4913       {
4914         \int_gincr:N \g__enumext_item_number_int
4915         \bool_set_true:N \l__enumext_item_number_bool
4916       }
4917     }
4918   }
4919   \bool_if:NT \l__enumext_item_starred_vii_bool
4920   {
4921     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4922     {
4923       \tl_gset_eq:NN
4924       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4925     }
4926     \mode_leave_vertical:
4927     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4928     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4929     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim

```

```

4930     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4931   }
4932   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4933   {
4934     \tl_use:N \l__enumext_label_font_style_vii_tl
4935     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4936       {
4937         \__enumext_wrapper_label_vii:n {#1}
4938       }
4939       { #1 }
4940   }
4941   \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4942 }

```

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

13.44.2 Real definition of `\item` in `enumext*`

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`__enumext_start_item_vii:w` The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to `__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and “*item content*” in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```

4943 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4944 {
4945   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4946   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4947   {
4948     \l__enumext_joined_width_vii_dim
4949     + \l__enumext_labelwidth_vii_dim
4950     + \l__enumext_labelsep_vii_dim
4951   }

```

Redefine the `\footnote` command.

```

4952   \__enumext_renew_footnote_starred:

```

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

```

4953   \__enumext_start_list_tag:n {enumext*}
4954   \__enumext_fake_make_label_vii:n {#1}
4955   \__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.

```

4956   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4957   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4958   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4959   \__enumext_unskip_unkern:
4960   \__enumext_unskip_unkern:
4961   \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4962   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4963   \tl_use:N \l__enumext_after_list_args_vii_tl
4964 }

```

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*.

```

4965 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4966 {
4967   \__enumext_endminipage:
4968   \__enumext_stop_list_tag:n {enumext*}
4969   \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*.

```

4970   \int_set:Nn \hbadness { 10000 }
4971   \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`.

```

4972 \int_compare:nNnTF
4973 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4974 {
4975     \par\noindent
4976     \int_zero:N \l__enumext_item_column_pos_vii_int
4977 }
4978 {
4979     \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4980 }
4981 }

```

(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)*.

```

4982 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4983 {
4984     \int_compare:nNnTF
4985     {
4986         \int_mod:nn
4987         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4988     }
4989     =
4990     { 0 }
4991     {
4992         \para_end:
4993         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4994         \skip_vertical:N \c_zero_skip
4995         \int_gzero:N \g__enumext_item_count_all_vii_int
4996     }
4997 }

```

(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`.

```

4998 \__enumext_after_env:nn {enumext*}
4999 {
5000     \__enumext_execute_after_env:
5001 }

```

13.45 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.

```

5002 \NewDocumentEnvironment{keyans*}{ o }
5003 {
5004     \__enumext_safe_exec_viii:
5005     \__enumext_parse_keys_viii:n {#1}
5006     \__enumext_before_list_viii:
5007     \__enumext_start_list:nn { }
5008     {
5009         \__enumext_list_arg_two_viii:
5010         \__enumext_before_keys_exec_viii:
5011     }
5012     \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { } { }
5013     \__enumext_starred_columns_set_viii:
5014     \item[] \scan_stop:
5015     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5016     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
5017     \ignorespaces
5018 }
5019 {
5020     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { } { }
5021     \__enumext_stop_item_tmp_viii:
5022     \__enumext_remove_extra_parsep_viii:
5023     \__enumext_check_starred_cmd:n { item }

```



```

5024   \__enumext_after_list_viii:
5025   }

```

(End of definition for `keyans*`. This function is documented on page 15.)

`__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.

```

5026 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5027 {
5028   \bool_if:NF \l__enumext_store_active_bool
5029   {
5030     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
5031   }
5032   \int_incr:N \l__enumext_keyans_level_h_int
5033   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5034   {
5035     \msg_error:nn { enumext } { nested }
5036   }
5037   \__enumext_keyans_name_and_start:
5038   \bool_if:NT \l__enumext_starred_bool
5039   {
5040     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5041   }
5042   \bool_set_true:N \l__enumext_starred_bool
5043   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5044   \bool_set_false:N \l__enumext_store_active_bool
5045   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5046   {
5047     \msg_error:nn { enumext } { keyans-wrong-level }
5048   }
5049 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

`__enumext_parse_keys_viii:n`

Parse [`<key = val>`] for `keyans*`.

```

5050 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5051 {
5052   \tl_if_novalue:nF {#1}
5053   {
5054     \keys_set:nn { enumext / keyans* } {#1}
5055   }
5056 }

```

(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`.

```

5057 \cs_new_protected:Nn \__enumext_before_list_viii:
5058 {
5059   \__enumext_vspace_above_viii:
5060   \__enumext_before_args_exec_viii:
5061   \__enumext_start_mini_viii:
5062 }

```

(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.

```

5063 \cs_new_protected:Nn \__enumext_after_list_viii:
5064 {
5065   \__enumext_stop_mini_viii:
5066   \__enumext_after_stop_list_viii:
5067   \__enumext_vspace_below_viii:
5068 }

```

(End of definition for `__enumext_after_list_viii:`.)

13.45.1 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.

```
5069 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5070 {
5071   \skip_horizontal:n
5072   {
5073     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
5074   }
5075   \ignorespaces
5076 }
```

(End of definition for `__enumext_first_item_tmp_viii:`.)

`__enumext_start_item_tmp_viii:`

`__enumext_item_peek_args_viii:`

`__enumext_joined_item_viii:w`

`__enumext_standar_item_viii:w`

First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
5077 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5078 {
5079   \__enumext_stop_item_tmp_viii:
5080   \int_incr:N \l__enumext_item_column_pos_viii_int
5081   \int_gincr:N \g__enumext_item_count_all_viii_int
5082   \__enumext_item_peek_args_viii:
5083 }
```

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)`.

```
5084 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5085 {
5086   \peek_meaning:NTF (
5087     { \__enumext_joined_item_viii:w }
5088     { \__enumext_joined_item_viii:w (1) }
5089 }
```

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`.

```
5090 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5091 {
5092   \__enumext_starred_joined_item_viii:n {#1}
5093   \peek_meaning_remove:NTF *
5094     { \__enumext_starred_item_viii:w }
5095     { \__enumext_standar_item_viii:w }
5096 }
```

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

```
5097 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5098 {
5099   \bool_set_false:N \l__enumext_item_starred_viii_bool
5100   \bool_set_false:N \l__enumext_item_wrap_key_bool
5101   \peek_meaning:NTF [
5102     {
5103       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
```

```

5104     \__enumext_start_item_viii:w
5105   }
5106   {
5107     \bool_set_true:N \__enumext_wrap_label_viii_bool
5108     \legacy_if_set_true:n { @noitemarg }
5109     \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5110   }
5111 }

```

(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]`.

```

5112 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5113 {
5114   \bool_set_true:N \__enumext_item_starred_viii_bool
5115   \bool_set_true:N \__enumext_item_wrap_key_bool
5116   \bool_set_true:N \__enumext_wrap_label_viii_bool
5117   \peek_meaning:NTF [
5118     { \__enumext_starred_item_viii_aux_i:w }
5119     { \__enumext_starred_item_viii_aux_ii:w }
5120   }

```

The function __enumext_starred_item_viii_aux_i:w will save the *optional argument* to `\item*` in `__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `__enumext_store_current_label_tl` if present, then call the function `__enumext_starred_item_viii_aux_ii:w`.

```

5121 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5122 {
5123   \tl_clear:N \__enumext_store_current_label_tl
5124   \tl_if_no_value:nF { #1 }
5125   {
5126     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_viii_tl
5127     {
5128       \tl_put_right:Ne \__enumext_store_current_label_tl
5129       {
5130         \__enumext_store_keyans_item_opt_sep_viii_tl
5131       }
5132       \tl_put_right:Ne \__enumext_store_current_label_tl { #1 }
5133     }
5134     \tl_set:Ne \__enumext_store_current_opt_arg_tl { #1 }
5135   }
5136   \__enumext_starred_item_viii_aux_ii:w
5137 }
5138 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5139 {
5140   \legacy_if_set_true:n { @noitemarg }
5141   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5142 }

```

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.

```

5143 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5144 {
5145   \tl_put_left:Ne \__enumext_store_current_label_tl { \__enumext_label_viii_tl }
5146   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
5147   \__enumext_keyans_store_ref:
5148   \tl_put_left:Ne \__enumext_store_current_label_tl { \item }
5149   \__enumext_keyans_addto_seq_link:
5150   \int_gincr:N \g__enumext_check_starred_cmd_int
5151   \dim_compare:nNt { \__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5152   {
5153     \dim_set:Nn \__enumext_mark_sym_sep_viii_dim { \__enumext_labelsep_viii_dim }
5154   }
5155   \bool_if:NT \__enumext_show_answer_bool
5156   {
5157     \tl_set_eq:NN \__enumext_mark_answer_sym_tl \__enumext_mark_answer_sym_viii_tl
5158     \str_set_eq:NN \__enumext_mark_position_str \__enumext_mark_position_viii_str
5159     \__enumext_print_keyans_box:NN

```

```

5160         \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5161     }
5162     \bool_if:NT \l__enumext_show_position_bool
5163     {
5164         \tl_set:Nc \l__enumext_mark_answer_sym_tl
5165         {
5166             \group_begin:
5167             \exp_not:N \normalfont
5168             \exp_not:N \footnotesize [ \int_eval:n
5169             {
5170                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5171             }
5172             ]
5173             \group_end:
5174         }
5175         \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5176         \__enumext_print_keyans_box:NN
5177         \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5178     }
5179 }

```

(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.

```

5180 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5181 {
5182     \bool_lazy_all:nT
5183     {
5184         { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5185         { \bool_if_p:N \l__enumext_show_answer_bool }
5186         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5187         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5188     }
5189     {
5190         \cs_set_eq:NN
5191         \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5192     }
5193     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5194     {
5195         \__enumext_wrapper_label_viii:n {#1}
5196     }
5197     { #1 }
5198 }
5199 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5200 {
5201     \legacy_if:nT { @noitemarg }
5202     {
5203         \legacy_if_set_false:n { @noitemarg }
5204         \legacy_if:nT { @nmbrrlist }
5205         {
5206             \refstepcounter{enumXviii}
5207         }
5208     }
5209     \bool_if:NT \l__enumext_item_starred_viii_bool
5210     {
5211         \__enumext_keyans_starred_item_star:
5212     }
5213     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5214     {
5215         \tl_use:N \l__enumext_label_font_style_viii_tl
5216         \__enumext_keyans_wrapper_label_viii:n {#1}
5217     }
5218     \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5219 }

```

(End of definition for `__enumext_keyans_wrapper_label_viii:n` and `__enumext_fake_make_label_viii:n`.)

13.45.2 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.

```

5220 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5221 {

```

```

5222 \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5223 \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5224 {
5225   \l__enumext_joined_width_viii_dim
5226   + \l__enumext_labelwidth_viii_dim
5227   + \l__enumext_labelsep_viii_dim
5228 }
5229 \__enumext_renew_footnote_starred:
5230 \__enumext_start_list_tag:n {keyans*}
5231 \__enumext_fake_make_label_viii:n {#1}
5232 \__enumext_stop_start_list_tag:
5233 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5234 \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5235 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5236 \__enumext_unskip_unkern:
5237 \__enumext_unskip_unkern:
5238 \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5239 \tl_use:N \l__enumext_fake_item_indent_viii_tl
5240 \bool_if:NT \l__enumext_item_starred_viii_bool
5241 {
5242   \__enumext_keyans_show_item_opt_viii:
5243 }
5244 \tl_use:N \l__enumext_after_list_args_viii_tl
5245 }
5246 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5247 {
5248   \__enumext_endminipage:
5249   \__enumext_stop_list_tag:n {keyans*}
5250   \hbox_set_end:
5251   \int_set:Nn \hbadness { 10000 }
5252   \box_use_drop:N \l__enumext_item_text_viii_box
5253   \int_compare:nNnTF
5254   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5255   {
5256     \par\noindent
5257     \int_zero:N \l__enumext_item_column_pos_viii_int
5258   }
5259   {
5260     \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5261   }
5262 }

```

(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.

```

5263 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5264 {
5265   \int_compare:nNnT
5266   {
5267     \int_mod:nn
5268     { \g__enumext_item_count_all_viii_int }
5269     { \l__enumext_columns_viii_int }
5270   }
5271   =
5272   { 0 }
5273   {
5274     \para_end:
5275     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5276     \skip_vertical:N \c_zero_skip
5277     \int_gzero:N \g__enumext_item_count_all_viii_int
5278   }
5279 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

13.46 The command \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-anskey`.

```

5280 \NewDocumentCommand \getkeyans { m }
5281 {

```

```

5282 \exp_args:Ne \__enumext_getkeyans_aux:n
5283 { \tl_to_str:e { \text_expand:n {#1} } }
5284 }

```

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

```

5285 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5286 {
5287   \str_if_in:nnTF {#1} { : }
5288   {
5289     \use:e
5290     {
5291       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5292       { {##1} {##2} }
5293     }
5294     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5295   }
5296   { \msg_error:nnn { enumext } { missing-colon } {#1} }
5297 }

```

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*.

```

5298 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5299 {
5300   \prop_if_exist:cTF { g__enumext_#1_prop }
5301   {
5302     \prop_item:cn { g__enumext_#1_prop }{#2}
5303   }
5304   {
5305     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5306   }
5307 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 18.)

13.47 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⟩]`.

```

5308 \keys_define:nn { enumext / print }
5309 {
5310   print* .code:n = \keys_precompile:neN { enumext / enumext* }
5311               { \__enumext_filter_save_key:n {#1} }
5312               \l__enumext_print_keyans_starred_tl, % starred cmd
5313   print* .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5314                       rightmargin=opt, listparindent=opt, nosepl, label=\arabic*,
5315                       columns=2, first=\small, font=\small },
5316   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5317                   { \__enumext_filter_save_key:n {#1} }
5318                   \l__enumext_print_keyans_i_tl,
5319   print-1 .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5320                       rightmargin=opt, listparindent=opt, nosepl, label=\arabic*,
5321                       columns=2, first=\small, font=\small },
5322   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5323                   { \__enumext_filter_save_key:n {#1} }
5324                   \l__enumext_print_keyans_ii_tl,
5325   print-2 .initial:n = { labelwidth=opt, labelsep=0.3333em, itemindent=opt, list-offset=opt,
5326                       rightmargin=opt, listparindent=opt, nosepl, label=(\alph*),
5327                       first=\small, font=\small },
5328   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5329                   { \__enumext_filter_save_key:n {#1} }
5330                   \l__enumext_print_keyans_iii_tl,

```

```

5331   print-3 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5332                        rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5333                        first=\small, font=\small },
5334   print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5335                        { \__enumext_filter_save_key:n {#1} }
5336                        \l__enumext_print_keyans_iv_tl,
5337   print-4 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5338                        rightmargin=0pt, listparindent=0pt, nosep, label=\Alpha*.,
5339                        first=\small, font=\small },
5340   print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5341                        { \__enumext_filter_save_key:n {#1} }
5342                        \l__enumext_print_keyans_vii_tl, % starred nested
5343   print-* .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5344                        rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5345                        first=\small, font=\small },
5346   }

```

• The reason for storing *(keys)* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

`\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`.

```

5347 \NewDocumentCommand \printkeyans { s O{} m }
5348 {
5349   \group_begin:
5350     \tl_use:N \l__enumext_print_keyans_i_tl
5351     \tl_use:N \l__enumext_print_keyans_ii_tl
5352     \tl_use:N \l__enumext_print_keyans_iii_tl
5353     \tl_use:N \l__enumext_print_keyans_iv_tl
5354     \tl_use:N \l__enumext_print_keyans_vii_tl
5355     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5356   \group_end:
5357 }

```

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.

```

5358 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5359 {
5360   \seq_if_exist:cTF { g__enumext_#3_seq }
5361   {
5362     \seq_if_empty:cF { g__enumext_#3_seq }
5363     {

```

If the *starred argument* ‘`*`’ is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default *(keys)* for the environment `enumext*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```

5364     \bool_if:nTF {#1}
5365     {
5366       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5367       {
5368         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5369       }
5370       {
5371         \tl_use:N \l__enumext_print_keyans_starred_tl
5372         \bool_set_true:N \l__enumext_base_line_fix_bool
5373         \bool_set_true:N \l__enumext_print_keyans_star_bool
5374         \begin{enumext*}[#2]
5375           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5376           \end{enumext*}
5377         \bool_set_false:N \l__enumext_base_line_fix_bool
5378         \bool_set_false:N \l__enumext_print_keyans_star_bool
5379       }
5380     }

```

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

```

5381   {

```



```

5382         \begin{enumext}[#2]
5383         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5384         \end{enumext}
5385     }
5386 }
5387 }
5388 {
5389     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5390 }
5391 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 19.)

13.48 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*` environment and “first level” of the `enumext` environment so as not to capture *⟨keys⟩* that complicate us.

The function `__enumext_filter_first_level:n` will be in charge of filtering the *⟨keys⟩* passed to the environment `enumext*` and “first level” of the environment `enumext`.

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
5392 \cs_new:Npn \__enumext_filter_first_level:n #1
5393 {
5394     \use:e
5395     {
5396         \keyval_parse:NNn
5397         \__enumext_filter_first_level_key:n
5398         \__enumext_filter_first_level_pair:nn {#1}
5399     }
5400 }

```

The function `__enumext_filter_first_level_key:n` will be responsible for filtering the *⟨keys⟩* that are passed “without value” by excluding the keys `resume` and `resume*`.

```

5401 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5402 {
5403     \str_case:nnF {#1}
5404     {
5405         { resume } {}
5406         { resume* } {}
5407     }
5408     { , { \exp_not:n {#1} } }
5409 }

```

The function `__enumext_filter_first_level_pair:nn` will be responsible for filtering the *⟨keys⟩* that are passed “with value” by excluding the `series`, `resume` and `save-ans` keys.

```

5410 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5411 {
5412     \str_case:nnF {#1}
5413     {
5414         { series } {}
5415         { resume } {}
5416         { save-ans } {}
5417     }
5418     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5419 }

```

(End of definition for `__enumext_filter_first_level:n`, `__enumext_filter_first_level_key:n`, and `__enumext_filter_first_level_pair:nn`.)

Now define a “meta families” of *⟨keys⟩* to access from `\setenumext`.

```

5420 \keys_define:nn { enumext / meta-families }
5421 {
5422     enumext-1 .code:n =
5423     {
5424         \keys_set:ne { enumext / level-1 }
5425         {
5426             \__enumext_filter_first_level:n {#1}
5427         }
5428     } ,
5429     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5430     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5431     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5432     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,

```

```

5433     enumext* .code:n =
5434         {
5435             \keys_set:ne { enumext / enumext* }
5436             {
5437                 \__enumext_filter_first_level:n {#1}
5438             }
5439         } ,
5440     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5441     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5442     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5443     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5444     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5445     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5446     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5447     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5448 }

```

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

```

5449 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5450 {
5451     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5452     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5453 }

```

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

```

\__enumext_set_parse:n 5454 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
\__enumext_set_error:nn 5455 {
5456     \seq_clear:N \l__enumext_setkey_tmpa_seq
5457     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5458     \int_set:Nn \l__enumext_setkey_tmpa_int
5459     {
5460         \seq_count:N \l__enumext_setkey_tmpb_seq
5461     }
5462     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5463     {
5464         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5465         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5466         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5467         {
5468             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5469         }
5470     }
5471     {
5472         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5473     }
5474     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5475     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5476     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5477     {
5478         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5479     }
5480 }

```

Internal functions used by the `\setenumext` command.

```

5481 \cs_new_protected:Npn \__enumext_set_parse:n #1
5482 {
5483     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5484     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5485     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5486     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5487     {
5488         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5489         { \tl_trim_spaces:n {#1} }
5490     }
5491     { \__enumext_set_error:nn {#1} { } }
5492 }
5493 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5494 { \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.)

13.49 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

5495 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5496 {
5497   {enumext,1} = level-1,
5498   {enumext,2} = level-2,
5499   {enumext,3} = level-3,
5500   {enumext,4} = level-4,
5501   {enumext*} = enumext*
5502 }
```

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

```
5503 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5504 {
5505   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5506   { \msg_error:nn { enumext } { prohibited-unknown } }
5507   {
5508     \bool_if:nTF {#1}
5509     {
5510       \int_step_inline:nn { 4 }
5511       { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5512       __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5513     }
5514     { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
5515   }
5516 }
```

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

```
5517 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
5518 {
5519   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5520   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5521   \prop_get:NVNTF
5522   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5523   { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5524   {
5525     \msg_error:nnn { enumext } { unknown-set } {#1}
5526     \use_none:nn
5527   }
5528 }
5529 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
5530 {
5531   \bool_lazy_or:nnTF
5532   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5533   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5534   { \msg_error:nnn { enumext } { already-defined } {#2} }
5535   {
5536     \keys_define:nn { enumext / #1 }
5537     {
5538       #2 .meta:n = {#3},
5539       #2 .value_forbidden:n = true
5540     }
5541   }
5542 }
5543 \cs_generate_variant:Nn __enumext_def_meta_key:nnn { V }
```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

13.50 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
5544 \keys_define:nn { enumext / foreach }
```

```

5545 {
5546   before .tl_set:N = \l__enumext_foreach_before_tl,
5547   before .value_required:n = true,
5548   after  .tl_set:N = \l__enumext_foreach_after_tl,
5549   after  .value_required:n = true,
5550   start  .int_set:N = \l__enumext_foreach_start_int,
5551   start  .value_required:n = true,
5552   stop   .int_set:N = \l__enumext_foreach_stop_int,
5553   stop   .value_required:n = true,
5554   step   .int_set:N = \l__enumext_foreach_step_int,
5555   step   .value_required:n = true,
5556   wrapper .cs_set_protected:Np = \l__enumext_foreach_wrapper:n #1,
5557   wrapper .value_required:n = true,
5558   sep     .tl_set:N = \l__enumext_foreach_sep_tl,
5559   sep     .value_required:n = true,
5560   unknown .code:n = { \l__enumext_parse_foreach_keys:n {#1} }
5561 }
5562 \keys_precompile:nnN { enumext / foreach }
5563 {
5564   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5565 }
5566 \g__enumext_foreach_default_keys_tl

```

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

```

5567 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:nn #1#2
5568 {
5569   \tl_if_blank:nTF {#2}
5570   {
5571     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5572   }
5573   {
5574     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5575   }
5576 }
5577 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:n #1
5578 {
5579   \exp_args:NV \l__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5580 }

```

We create the command.

```

5581 \NewDocumentCommand \foreachkeyans { +0{ } m }
5582 {
5583   \l__enumext_foreach_keyans:nn {#1} {#2}
5584 }

```

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.

```

5585 \cs_new_protected:Npn \l__enumext_foreach_keyans:nn #1 #2
5586 {
5587   \tl_use:N \g__enumext_foreach_default_keys_tl
5588   \keys_set:nn { enumext / foreach } {#1}
5589   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5590   \prop_if_exist:cF { g__enumext_#2_prop }
5591   {
5592     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5593   }
5594   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5595   {
5596     \int_set:Nn \l__enumext_foreach_stop_int
5597     { \prop_count:c { g__enumext_#2_prop } }
5598   }
5599   \seq_clear:N \l__enumext_foreach_print_seq
5600   \int_step_function:nnnN
5601   { \l__enumext_foreach_start_int }
5602   { \l__enumext_foreach_step_int }
5603   { \l__enumext_foreach_stop_int }
5604   \l__enumext_foreach_add_body:n
5605   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5606 }
5607 \cs_new_protected:Npn \l__enumext_foreach_add_body:n #1
5608 {
5609   \seq_put_right:Ne \l__enumext_foreach_print_seq

```

```

5610     {
5611         \exp_not:V \l__enumext_foreach_before_tl
5612         \__enumext_foreach_wrapper:n
5613         {
5614             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5615         }
5616         \exp_not:V \l__enumext_foreach_after_tl
5617     }
5618 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 18.)

13.51 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5619 \msg_new:nnn { enumext } { package-load }
5620 {
5621     The ~ '#1' ~ package ~ is ~ already ~ loaded.
5622 }
5623 \msg_new:nnn { enumext } { package-not-load }
5624 {
5625     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5626 }
5627 \msg_new:nnn { enumext } { package-load-foot }
5628 {
5629     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5630 }

```

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

```

5631 \msg_new:nnn { enumext } { counters }
5632 {
5633     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5634     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5635 }

```

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

```

5636 \msg_new:nnn { enumext } { unknown-choice }
5637 {
5638     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5639 }

```

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

```

5640 \msg_new:nnnn { enumext } { anskey-env-error }
5641 {
5642     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5643     'enumext' ~ package, ~ It~ is~ already~ defined.
5644 }
5645 {
5646     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5647     for ~ the ~ 'save-ans' ~ key.\\
5648 }

```

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

```

5649 \msg_new:nnn { enumext } { store-prop }
5650 {
5651     * ~ Package ~ enumext: ~ Creating ~
5652     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5653 }
5654 \msg_new:nnn { enumext } { store-seq }
5655 {
5656     * ~ Package ~ enumext: ~ Creating ~
5657     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5658 }
5659 \msg_new:nnn { enumext } { store-int }
5660 {
5661     * ~ Package ~ enumext: ~ Creating ~
5662     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5663 }
5664 \msg_new:nnn { enumext } { prop-seq-int-hook }
5665 {
5666     * ~ Package ~ enumext: ~ Elements ~ in ~
5667     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5668     * ~ Package ~ enumext: ~ Elements ~ in ~

```

```

5669 \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5670 * ~ Package ~ enumext: ~ Value ~ off ~
5671 \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5672 }
5673 \msg_new:nnn { enumext } { item-answer-hook }
5674 {
5675   * ~ Package ~ enumext: ~ Value ~ off ~
5676   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5677   * ~ Package ~ enumext: ~ Value ~ off ~
5678   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5679   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5680 }

```

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

```

5681 \msg_new:nnn { enumext } { invalid-key }
5682 {
5683   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5684 }
5685 \msg_new:nnn { enumext } { unknown-key-family }
5686 {
5687   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5688 }

```

Messages used in length calculation.

```

5689 \msg_new:nnn { enumext } { width-negative }
5690 {
5691   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5692   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5693 }
5694 \msg_new:nnn { enumext } { width-zero }
5695 {
5696   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5697   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5698 }

```

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

```

5699 \msg_new:nnn { enumext } { list-lengths }
5700 {
5701   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5702   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5703   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5704   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5705   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5706   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5707   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5708   \__enumext_show_length:nnn { skip } { topsep } {#1}
5709   \__enumext_show_length:nnn { skip } { parsep } {#1}
5710   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5711   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5712   ****
5713 }

```

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

```

5714 \msg_new:nnn { enumext } { list-lengths-not-nested }
5715 {
5716   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5717   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5718   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5719   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5720   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5721   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5722   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5723   \__enumext_show_length:nnn { skip } { topsep } {#1}
5724   \__enumext_show_length:nnn { skip } { parsep } {#1}
5725   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5726   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5727   ****
5728 }

```

Messages used by `ref` key.

```

5729 \msg_new:nnn { enumext } { key-ref-empty }
5730 {
5731   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5732 }

```

Messages used by `save-ans` key.

```

5733 \msg_new:nnn { enumext } { save-ans-empty }
5734 {
5735   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ \msg_line_context:.
5736 }
5737 \msg_new:nnn { enumext } { save-ans-log }
5738 {
5739   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5740 }
5741 \msg_new:nnn { enumext } { save-ans-log-hook }
5742 {
5743   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5744 }
5745 \msg_new:nnn { enumext } { save-ans-hook }
5746 {
5747   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5748 }

```

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

```

5749 \msg_new:nnn { enumext } { need-save-ans }
5750 {
5751   Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5752 }
5753 \msg_new:nnn { enumext } { items-same-answer }
5754 {
5755   *****\\
5756   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5757   for ~ \c_left_brace_str #2 \c_right_brace_str\\
5758   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5759   'OK', ~ all ~ items ~ with ~ answer.\\
5760   *****
5761 }
5762 \msg_new:nnn { enumext } { item-greater-answer }
5763 {
5764   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5765   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5766   Items ~ > ~ Answers.
5767 }
5768 \msg_new:nnn { enumext } { item-less-answer }
5769 {
5770   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5771   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5772   Items ~ < ~ Answers.
5773 }

```

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

```

5774 \msg_new:nnn { enumext } { missing-starred }
5775 {
5776   Missing ~ '\c_backslash_str #1*' ~ #2.
5777 }
5778 \msg_new:nnn { enumext } { many-starred }
5779 {
5780   Many ~ '\c_backslash_str #1*' ~ #2.
5781 }

```

Messages used by `\printkeyans*` command.

```

5782 \msg_new:nnn { enumext } { print-starred }
5783 {
5784   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5785   #2 ~ environment ~ \msg_line_context:.
5786 }

```

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

```

5787 \msg_new:nnn { enumext } { list-too-deep }
5788 {
5789   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5790   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5791 }

```

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

```

5792 \msg_new:nnn { enumext } { anskey-unnumber-item }
5793 {
5794   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.

```



```

5795     }
5796 \msg_new:nnn { enumext } { anskey-already-stored }
5797 {
5798     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5799 }
5800 \msg_new:nnn { enumext } { anskey-empty-arg }
5801 {
5802     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5803 }
5804 \msg_new:nnn { enumext } { anskey-wrong-place }
5805 {
5806     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5807     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5808 }
5809 \msg_new:nnn { enumext } { anskey-nested }
5810 {
5811     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5812 }
5813 \msg_new:nnn { enumext } { anskey-math-mode }
5814 {
5815     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5816 }
5817 \msg_new:nnn { enumext } { anskey-env-wrong }
5818 {
5819     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5820 }
5821 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5822 {
5823     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5824     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5825 }
5826 \msg_new:nnn { enumext } { command-wrong-place }
5827 {
5828     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5829     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5830 }
5831 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5832 {
5833     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5834     'anskey*' ~ and ~ is ~ being ~ ignored.
5835 }
5836 {
5837     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5838     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5839 }
5840 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5841 {
5842     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5843     'anskey*' ~ and ~ is ~ being ~ ignored.
5844 }
5845 {
5846     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5847     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5848 }
5849 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5850 { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5851 {
5852     The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5853     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5854 }
5855 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5856 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5857 {
5858     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5859     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5860 }

```

Messages used by [keyans](#), [keyans*](#) and [keyansPIC](#) environment.

```

5861 \msg_new:nnn { enumext } { keyans-nested }
5862 {
5863     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5864 }

```

```

5865 \msg_new:nnn { enumext } { keyans-wrong-level }
5866 {
5867   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5868   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5869 }
5870 \msg_new:nnn { enumext } { wrong-place }
5871 {
5872   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5873   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5874 }
5875 \msg_new:nnn { enumext } { keyanspic-nested }
5876 {
5877   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5878 }
5879 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5880 {
5881   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5882   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5883 }
5884 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5885 {
5886   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5887 }
5888 \msg_new:nnnn { enumext } { keyans-unknown-key }
5889 {
5890   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5891   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5892 }
5893 {
5894   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5895   ~ have ~ a ~ key ~ called ~ '#1'.\\
5896   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5897 }
5898 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5899 {
5900   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5901   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5902 }
5903 {
5904   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5905   ~ have ~ a ~ key ~ called ~ '#1'.\\
5906   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5907 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*`. environment.

```

5908 \msg_new:nnnn { enumext } { starred-unknown-key }
5909 {
5910   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5911   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5912 }
5913 {
5914   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5915   ~ have ~ a ~ key ~ called ~ '#1'.\\
5916   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5917 }
5918 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5919 {
5920   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5921   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5922 }
5923 {
5924   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5925   ~ have ~ a ~ key ~ called ~ '#1'.\\
5926   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5927 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5928 \msg_new:nnnn { enumext } { standar-unknown-key }
5929 {
5930   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5931   ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5932 }

```

```

5933 {
5934     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5935     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5936     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5937 }
5938 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5939 {
5940     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5941     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5942 }
5943 {
5944     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5945     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5946     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5947 }

```

Message used by unknown *(keys)* in `\foreachkeyans`.

```

5948 \msg_new:nnnn { enumext } { for-key-unknown }
5949 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5950 {
5951     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5952     Check~that~you~have~spelled~the~key~name~correctly.
5953 }
5954 \msg_new:nnnn { enumext } { for-key-value-unknown }
5955 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5956 {
5957     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5958     Check~that~you~have~spelled~the~key~name~correctly.
5959 }

```

Messages used by `\getkeyans` command.

```

5960 \msg_new:nnn { enumext } { undefined-storage-anskey }
5961 {
5962     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5963 }

```

Messages used by `\miniright` command.

```

5964 \msg_new:nnn { enumext } { missing-miniright }
5965 {
5966     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5967     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5968 }
5969 \msg_new:nnn { enumext } { wrong-miniright-place }
5970 {
5971     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5972     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5973 }
5974 \msg_new:nnn { enumext } { wrong-miniright-use }
5975 {
5976     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5977     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5978 }
5979 \msg_new:nnn { enumext } { wrong-miniright-starred }
5980 {
5981     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5982 }
5983 \msg_new:nnn { enumext } { many-miniright-used }
5984 {
5985     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5986 }

```

Messages used by `\setenumextmeta` command.

```

5987 \msg_new:nnn { enumext } { unknown-set }
5988 {
5989     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5990 }
5991 \msg_new:nnn { enumext } { already-defined }
5992 {
5993     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5994 }
5995 \msg_new:nnn { enumext } { prohibited-unknown }
5996 {
5997     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.

```

5998 }

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

```
5999 \msg_new:nnn { enumext } { nested }
6000 {
6001   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
6002 }
6003 \msg_new:nnn { enumext } { nested-horizontal }
6004 {
6005   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
6006 }
6007 \msg_new:nnn { enumext } { item-joined }
6008 {
6009   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
6010 }
6011 \msg_new:nnn { enumext } { item-joined-columns }
6012 {
6013   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
6014 }
```

13.52 Finish package

Finish package implementation.

```
6015 \file_input_stop:
6016 </package>
```

14 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>*</code>	232
<code>\+</code>	224
<code>\-</code>	224
<code>\\</code> 240, 2962, 4409, 4412, 5633, 5642, 5647, 5667, 5669, 5676, 5678, 5691, 5696, 5701, 5716, 5755, 5757, 5759, 5764, 5765, 5770, 5771, 5789, 5806, 5823, 5828, 5837, 5846, 5852, 5858, 5867, 5872, 5881, 5895, 5905, 5915, 5925, 5935, 5945, 5951, 5957, 5966, 5971, 5976	
A	
<code>above</code>	<u>1721</u>
<code>above*</code>	<u>1721</u>
<code>\addvspace</code> 1288, 1316, 1359, 1362, 1530, 1533, 1630, 1636, 1674, 1680, 1701, 1707, 3837, 3998, 4016, 4294, 4298, 4657, 4672, 4718, 4732	
<code>after</code>	<u>1118</u>
<code>align</code>	<u>669</u>
<code>\Alph</code>	43, <u>48</u>
<code>\Alpha</code>	611, 739, 783, 847, 5338
<code>\alph</code>	43, <u>48</u>
<code>\alpha</code>	612, 737, 5326
<code>\anskey</code>	13, 82, 84, <u>2780</u>
<code>anskey*</code>	14, <u>2890</u>
<code>\anspic</code>	17, <u>112</u> , <u>115</u> , <u>4308</u>
<code>\anspic*</code>	75
<code>\arabic</code>	35, <u>43</u>
<code>\arabic</code>	610, 736, 782, 5314, 5320, 5344
B	
<code>base-fix</code>	<u>976</u>
<code>\baselineskip</code>	<u>56</u>
<code>\baselineskip</code>	992, 1003
<code>before</code>	<u>1118</u>
<code>before*</code>	<u>1118</u>
<code>below</code>	<u>1721</u>
<code>below*</code>	<u>1721</u>
bool commands:	
<code>\bool_gset_false:N</code> 353, 354, 355, 3066, 3068, 4674, 4678, 4734	
<code>\bool_gset_true:N</code> 261, 271, 1221, 2214, 2220, 4643, 4675, 4707, 4735	
<code>\bool_if:NTF</code> . 404, 414, 431, 505, 512, 521, 528, 542, 555, 1743, 1757, 1770, 1781, 1792, 1803, 1814, 1825, 1874, 1891, 1896, 1904, 1931, 1969, 1974, 1981, 1985, 2007, 2012, 2020, 2027, 2058, 2066, 2159, 2402, 2412, 2491, 2515, 2522, 2546, 2644, 2666, 2706, 2730, 2734, 2784, 2803, 2827, 2881, 2892, 2981, 3018, 3082, 3117, 3132, 3207, 3323, 3357, 3393, 3409, 3430, 3569, 3590, 3678, 3688, 3721, 3726, 3792, 3818, 3868, 3926, 3981, 4006, 4227, 4292, 4310, 4329, 4380, 4407, 4636, 4652, 4658, 4701, 4715, 4719, 4808, 4818, 4906, 4912, 4919, 4935, 5028, 5038, 5155, 5162, 5193, 5209, 5240	
<code>\bool_if:nTF</code> 1681, 1708, 3379, 3548, 3636, 4350, 5364, 5508	
<code>\bool_if_p:N</code> 280, 295, 986, 987, 999, 1000, 1653, 2038, 2039, 2047, 2048, 2172, 2198, 2211, 2212, 2217, 2218, 2579, 2589, 2601, 2616, 2617, 2651, 2692, 2693, 3004, 3194, 3195, 3224, 3225, 3237, 3238, 3278, 3279, 3298, 3299, 3582, 3583, 3584, 3765, 3767, 3778, 5184, 5185, 5186	
<code>\bool_lazy_all:nTF</code> 278, 293, 984, 2170, 2196, 2577, 2586, 2599, 2614, 3276, 3296, 3580, 3763, 3776, 5182	
<code>\bool_lazy_and:nnTF</code> 257, 267, 998, 1645, 1652, 2037, 2046, 2210, 2216, 2650, 2657, 2691, 2845, 2857, 3003, 3009, 3193	
<code>\bool_lazy_or:nnTF</code> . . 2099, 2106, 3223, 3236, 5531	
<code>\bool_new:N</code> 30, 31, 32, 33, 34, 35, 36, 37, 60, 69, 93, 98, 99, 104, 105, 108, 127, 139, 140, 147, 153, 154, 156, 160, 162, 163, 180, 192, 194	
<code>\bool_not_p:n</code> 258, 268, 988, 1654, 2588, 2652, 2658, 3005, 3010, 3766, 3779	
<code>\bool_set_eq:NN</code> 3332, 3529, 4859, 5103	
<code>\bool_set_false:N</code> 411, 1010, 2144, 2145, 2177, 2182, 2186, 2190, 2203, 2945, 3557, 3740, 3885, 3934, 4021, 4163, 4224, 4370, 4777, 4803, 4856, 5044, 5099, 5100, 5377, 5378	
<code>\bool_set_true:N</code> . 285, 286, 300, 301, 396, 399, 662, 1025, 1727, 1732, 1994, 2116, 2117, 2434, 2442, 2946, 3326, 3328, 3360, 3362, 3525, 3536, 3550, 3701, 3739, 3772, 3785, 3858, 3931, 3958, 4160, 4352, 4353, 4625, 4690, 4776, 4863, 4870, 4871, 4915, 5042, 5107, 5114, 5115, 5116, 5372, 5373	
box commands:	
<code>\box_dp:N</code> . . 1576, 1577, 1580, 1587, 1600, 1608, 1614, 1622, 4238, 4244, 4294, 4391	
<code>\box_ht:N</code> . . 1359, 1362, 1373, 1374, 1385, 1387, 1402, 1405, 1413, 1414, 1425, 1427, 1442, 1445, 1452, 1453, 1464, 1466, 1481, 1484, 1530, 1533, 1541, 1542, 1550, 1551, 1563, 1565	
<code>\box_ht_plus_dp:N</code> 4233, 4302, 4338	
<code>\box_new:N</code> 66, 149, 150, 187, 193	
<code>\box_use_drop:N</code> 4669, 4730, 4971, 5252	
<code>\box_wd:N</code> 618	
<code>break-col</code>	<u>2750</u>
C	
<code>\c</code>	232, 233, 883, 885, 897, 899
<code>\catcode</code>	2962
<code>\cB</code>	233
<code>\cE</code>	233
<code>\centering</code>	1683, 1710, 4435, 4662, 4723
<code>check-ans</code>	<u>2136</u>
Document class:	
<code>article</code>	49
clist commands:	
<code>\clist_const:Nn</code>	199
<code>\clist_map_function:nN</code>	4418
<code>\clist_map_inline:Nn</code> . 668, 931, 1117, 1132, 1213, 1737	
<code>\clist_map_inline:nn</code> . 45, 56, 74, 82, 95, 107, 142, 171, 198, 646, 699, 719, 1030, 1051, 1227, 1843, 2083, 2150, 2329, 2399, 2431, 2574, 3126, 3451, 3466, 3506, 3665, 3668, 3696, 3708, 3711, 3731, 5484	
<code>\columnbreak</code>	82
<code>\columnbreak</code>	2654
<code>columns</code>	<u>1197</u>
<code>columns-sep</code>	<u>1197</u>
<code>\columnsep</code>	105

`\columnsep` 3813, 3979
`\columnseprule` 105
`\columnseprule` 3816, 3980
 Commands provide by **enumext**:
 `\anskey` . 32, 71, 72, 78, 79, 81, 83–85, 91, 104, 105, 124, 133, 135, 142
 `\anspic*` 32, 33, 75, 79, 91, 92, 115, 133, 135
 `\anspic` 33, 79, 112, 115, 142
 `\foreachkeyans` 138, 145
 `\getkeyans` 79, 133, 145
 `\item*` 32, 33, 75, 79, 91, 92, 95, 99, 125, 126, 131, 133, 135
 `\item` 95, 99, 119, 125, 127, 130
 `\miniright` 31, 54, 62, 63, 106, 107, 145
 `\printkeyans*` 134
 `\printkeyans` 32, 79, 134, 135
 `\setenumextmeta` 138, 145
 `\setenumext` 32, 135–137, 141
 Counters defined by **enumext**:
 `enumXiii` 30, 42
 `enumXii` 30, 42
 `enumXiv` 30, 42
 `enumXi` 30, 42
 `enumXviii` 30, 42
 `enumXvii` 30, 42, 126
 `enumXvi` 30, 42
 `enumXv` 30, 42
 cs commands:
 `\cs_generate_variant:Nn` . 204, 205, 620, 636, 889, 905, 2483, 2488, 2564, 2898, 3655, 4420, 5543
 `\cs_if_exist:NTF` 590
 `\cs_if_exist_p:N` 3585, 5187
 `\cs_if_free:NTF` 2849, 2861
 `\cs_new:Nn` 218
 `\cs_new:Npn` . 236, 1844, 1853, 1861, 2446, 2455, 2463, 5392, 5401, 5410
 `\cs_new_eq:NN` . 380, 381, 386, 387, 416, 417, 420, 421
 `\cs_new_protected:Nn` . 228, 242, 250, 276, 309, 339, 345, 351, 357, 363, 371, 391, 439, 443, 461, 473, 491, 503, 519, 535, 548, 569, 759, 820, 869, 982, 1133, 1137, 1141, 1145, 1149, 1153, 1157, 1161, 1165, 1169, 1173, 1177, 1181, 1185, 1189, 1193, 1228, 1240, 1273, 1290, 1301, 1318, 1344, 1365, 1490, 1516, 1536, 1569, 1591, 1626, 1632, 1738, 1752, 1766, 1777, 1788, 1799, 1810, 1821, 1902, 2005, 2018, 2035, 2056, 2084, 2089, 2114, 2155, 2165, 2208, 2223, 2230, 2239, 2244, 2249, 2254, 2263, 2268, 2273, 2489, 2513, 2520, 2544, 2551, 2565, 2801, 2820, 2836, 2899, 2935, 2966, 3001, 3043, 3064, 3072, 3115, 3130, 3158, 3191, 3219, 3232, 3245, 3274, 3287, 3365, 3375, 3386, 3402, 3418, 3544, 3562, 3596, 3608, 3732, 3761, 3790, 3797, 3827, 3844, 3866, 3888, 3924, 3948, 3965, 3990, 4004, 4025, 4199, 4402, 4416, 4421, 4445, 4455, 4486, 4615, 4634, 4680, 4699, 4763, 4790, 4797, 4806, 4816, 4841, 4982, 5026, 5057, 5063, 5084, 5143, 5263
 `\cs_new_protected:Npn` 206, 210, 214, 424, 588, 605, 615, 621, 740, 784, 852, 876, 890, 1665, 1694, 1870, 1889, 1959, 1992, 2094, 2278, 2400, 2410, 2432, 2440, 2475, 2484, 2640, 2703, 2728, 2766, 2770, 2890, 2921, 2925, 2956, 3092, 3168, 3212, 3319, 3338, 3467, 3471, 3485, 3489, 3507, 3511, 3521, 3533, 3578, 3624, 3658, 3699, 3743, 3944, 4208, 4215, 4222, 4327, 4346, 4376, 4517, 4566, 4780, 4847, 4854, 4868, 4876, 4881, 4891, 5050, 5090, 5097, 5112, 5121, 5138, 5180, 5285, 5298, 5358, 5481, 5493, 5517, 5529, 5567, 5577, 5585, 5607
 `\cs_new_protected_nopar:Nn` . . . 4088, 4132, 4140,

4148, 4826, 4834, 4965, 5069, 5077, 5246
`\cs_new_protected_nopar:Npn` . . 4080, 4096, 4897, 4943, 5199, 5220
`\cs_set:Npn` 2575, 2612, 5291
`\cs_set_eq:NN` . . 3588, 4753, 4754, 4945, 5015, 5016, 5190, 5222
`\cs_set_protected:Nn` 1056, 1072, 1085, 1097
`\cs_set_protected:Npn` . 41, 50, 67, 75, 90, 96, 135, 167, 178, 637, 647, 669, 704, 720, 766, 906, 932, 1012, 1035, 1109, 1118, 1197, 1214, 1721, 1832, 2075, 2136, 2295, 2330, 2418, 2567, 3119, 3440, 3456, 3499, 3656, 3697
`\cs_to_str:N` 607, 630
`\cs_undefine:N` 2838, 2839, 2840, 2841

D

`\d` 224
`\DeclareDocumentEnvironment` 573
 dim commands:
 `\dim_abs:n` 3629, 3634
 `\dim_add:Nn` 3271, 4242, 4480, 4511
 `\dim_compare:nNnTF` . . 1058, 1074, 1087, 1099, 1377, 1389, 1417, 1429, 1456, 1468, 1545, 1553, 1667, 1696, 2708, 2716, 3266, 3626, 3631, 3637, 3643, 3645, 3647, 3802, 3849, 3952, 3969, 4217, 4457, 4473, 4488, 4504, 4617, 4682, 5151
 `\dim_compare:nTF` 2676, 3031, 3891, 4028
 `\dim_eval:n` 992, 4300, 4387
 `\dim_gset_eq:NN` 4626, 4691
 `\dim_gzero:N` 3070, 4677, 4737
 `\dim_new:N` . 63, 70, 71, 72, 92, 131, 132, 144, 151, 152, 186, 188, 189, 195
 `\dim_set:Nn` . 618, 1026, 2710, 2718, 3253, 3257, 3262, 3268, 3355, 3629, 3634, 3636, 3639, 3640, 3644, 3646, 3649, 3650, 3652, 3805, 3852, 3890, 3954, 3971, 4027, 4231, 4336, 4423, 4459, 4466, 4490, 4497, 4552, 4601, 4619, 4684, 4893, 5153
 `\dim_set_eq:NN` 727, 773, 844, 3350, 3667, 3710, 3813, 3979, 4559, 4562, 4563, 4608, 4611, 4612, 4886, 4957, 5234
 `\dim_sub:Nn` 3896, 4033, 4475, 4506
 `\dim_use:N` . 1059, 1067, 1668, 1678, 2554, 2557, 2562, 2720, 3370, 3372, 3425, 3803, 3807, 3808, 3810, 3850, 3855, 3856, 3862, 3893, 3898
 `\dim_zero:N` 3702, 3816, 3980, 4245
 `\dim_zero_new:N` 587
 `\c_zero_dim` 1061, 1075, 1088, 1100, 1668, 1696, 2678, 2708, 2716, 3033, 3253, 3266, 3626, 3631, 3637, 3644, 3803, 3850, 3893, 3952, 3969, 4030, 4217, 4457, 4473, 4488, 4504, 4617, 4682, 5151
`\dimeval` 2364

E

`\end` . . . 2517, 2548, 3834, 3995, 4282, 4437, 5366, 5376, 5384
 end internal commands:
 `\end__enumext_mini_page` . 1676, 1703, 3877, 4015, 4641, 4705, 4731
`\endgroup` 2962
`\endlist` 381
`\endminipage` 387
`enumext` 5, 3902
 enumext internal commands:
 `\l__enumext__resume_name_tl` 67
 `__enumext_add_meta_key:nnn` . . . 138, 5495, 5511, 5512, 5514, 5517
 `__enumext_add_pre_parsep:` . 55, 1238, 1240, 1240

```

\__enumext_after_args_exec: 53, 1133, 1145, 3915
\__enumext_after_args_exec_v: 1149, 1161, 4048
\__enumext_after_args_exec_vii: .. 1165, 1189
\__enumext_after_args_exec_viii: ..... 1193
\__enumext_after_env:nn 88, 89, 91, 108, 121, 128,
210, 210, 561, 565, 2976, 3920, 4650, 4713, 4998
\__enumext_after_hyperref: ... 38, 389, 389, 391
\l__enumext_after_list_args_v_tl ..... 1163
\l__enumext_after_list_args_vii_tl 1191, 4963
\l__enumext_after_list_args_viii_tl .. 1195,
5244
\__enumext_after_list_vii: 121, 124, 4761, 4797,
4797
\__enumext_after_list_viii: ... 129, 5024, 5063,
5063
\__enumext_after_stop_list: 53, 107, 1133, 1141,
3882
\__enumext_after_stop_list_v: 1149, 1157, 4022
\l__enumext_after_stop_list_v_tl ..... 1159
\__enumext_after_stop_list_vii: .. 124, 1165,
1181, 4800
\l__enumext_after_stop_list_vii_tl ... 1183
\__enumext_after_stop_list_viii: . 1185, 5066
\l__enumext_after_stop_list_viii_tl ... 1187
\l__enumext_align_label_pos_v_str 3249, 3614
\l__enumext_align_label_pos_X_str ..... 75
\l__enumext_align_label_vii_str ..... 4932
\l__enumext_align_label_viii_str ..... 5213
\l__enumext_align_label_X_str ..... 178
\c__enumext_all_envs_clist . 199, 668, 931, 1117,
1132, 1213, 1737
\c__enumext_all_families_seq .. 137, 5449, 5475
\l__enumext_anskey_env_bool 36, 87, 30, 286, 301,
2892
\__enumext_anskey_env_clean_vars: . 90, 2997,
3001, 3064
\__enumext_anskey_env_define_keys: 87, 2890,
2899, 2970
\__enumext_anskey_env_exec: 88, 2895, 2966, 2966
\__enumext_anskey_env_make:n 71, 87, 2119, 2890,
2890, 2898
\__enumext_anskey_env_reset_keys: 88, 89, 2890,
2935, 2998
\__enumext_anskey_env_save_keys: .. 89, 2978,
3001, 3001
\__enumext_anskey_env_store: .. 90, 2994, 3001,
3043
\__enumext_anskey_env_unknown:n 87, 2890, 2918,
2921
\__enumext_anskey_env_unknown:nn . 2890, 2923,
2925
\l__enumext_anskey_level_int .. 24, 2822, 2823
\__enumext_anskey_safe_inner: . 85, 2795, 2801,
2820
\__enumext_anskey_safe_inner:n ..... 85
\__enumext_anskey_safe_outer: . 85, 2782, 2801,
2801
\__enumext_anskey_show_wrap_arg:n . 83, 2703,
2703, 2732, 2747
\__enumext_anskey_show_wrap_left:n 84, 2648,
2728, 2728
\__enumext_anskey_unknown:n 84, 2750, 2764, 2766
\__enumext_anskey_unknown:nn . 2750, 2768, 2770
\__enumext_anskey_wrapper:n ..... 2361, 2726

\l__enumext_anspic_above_int . 143, 4424, 4425,
4427
\__enumext_anspic_args:nnn 115, 117, 4308, 4324,
4402
\l__enumext_anspic_args_seq 115, 117, 143, 4322,
4436, 4449
\l__enumext_anspic_below_int . 143, 4424, 4425,
4428
\l__enumext_anspic_body_box ... 143, 4335, 4338
\__enumext_anspic_body_dim:n .. 115, 4308, 4327,
4379
\l__enumext_anspic_body_htdp_dim .. 115, 143,
4336, 4390
__enumext_anspic_exec: ..... 4308
\__enumext_anspic_exec: ... 114, 117, 4277, 4445
\__enumext_anspic_label:nn 116, 4308, 4346, 4382,
4397
\l__enumext_anspic_label_above_bool ... 143,
4160, 4163, 4227, 4292, 4329, 4380, 4407
\l__enumext_anspic_label_box .. 143, 4230, 4233
\l__enumext_anspic_label_htdp_dim . 113, 143,
4231, 4237, 4302, 4389
\__enumext_anspic_label_pos:nnn .. 116, 4308,
4376, 4405
\l__enumext_anspic_label_sep_skip 4170, 4239,
4303, 4392, 4409
\l__enumext_anspic_layout_style_tl 4172, 4447,
4452
\l__enumext_anspic_mini_pos_str .. 143, 4161,
4164, 4434
\l__enumext_anspic_mini_width_dim 143, 4348,
4423, 4434
\__enumext_anspic_print:n 117, 4308, 4416, 4420,
4449, 4452
\__enumext_anspic_row:n .. 117, 4308, 4418, 4421
\__enumext_anspic_start_list_tag: 4104, 4132,
4404
\__enumext_anspic_stop_list_tag: . 4104, 4148,
4414
\__enumext_anspic_stop_start_list_tag: 4104,
4140, 4406
\__enumext_at_begin_document:n .. 38, 206, 206,
378, 384
\l__enumext_base_line_fix_bool 50, 135, 978, 987,
1010, 5372, 5377
\__enumext_before_args_exec: 53, 106, 124, 1133,
1133, 3847
\__enumext_before_args_exec_v: 1149, 1149, 3951
\__enumext_before_args_exec_vii: . 1165, 1165,
4794
\__enumext_before_args_exec_viii: 1169, 5060
\__enumext_before_env:nn 87, 210, 214, 2843, 2855,
2867, 2968
\__enumext_before_keys_exec: .. 53, 1133, 1137,
3912
\__enumext_before_keys_exec_v: 1149, 1153, 4045
\__enumext_before_keys_exec_vii ..... 1165
\__enumext_before_keys_exec_vii: . 1173, 4748
\__enumext_before_keys_exec_viii: 1177, 5010
\__enumext_before_list: .. 106, 3844, 3844, 3906
\__enumext_before_list_v: ... 3948, 3948, 4040
\__enumext_before_list_vii: .. 124, 4743, 4790,
4790
\__enumext_before_list_viii: .. 129, 5006, 5057,

```


5057
 \l__enumext_before_no_starred_key_v_tl 1155
 \l__enumext_before_no_starred_key_vii_-
 tl 1175
 \l__enumext_before_no_starred_key_viii_-
 tl 1179
 \l__enumext_before_starred_key_v_tl ... 1151
 \l__enumext_before_starred_key_vii_tl . 1167
 \l__enumext_before_starred_key_viii_tl 1171
 __enumext_calc_hspace:NNNNNN 101, 3624, 3624,
 3655, 3660, 3703
 __enumext_check_ans_active: 73, 106, 124, 2155,
 2155, 3848, 4793
 \g__enumext_check_ans_item_tl 93
 \g__enumext_check_ans_key_bool 74, 75, 153, 353,
 2214, 2220, 3082
 \l__enumext_check_ans_key_bool 74, 2140, 2145,
 2211, 2217
 __enumext_check_ans_key_hook: .. 74, 107, 124,
 2208, 2208, 3883, 4801
 __enumext_check_ans_level: 73, 2155, 2161, 2165
 __enumext_check_ans_log: 74, 75, 90, 2254, 2254,
 3086
 __enumext_check_ans_log_msg_greater: 2254,
 2260, 2273
 __enumext_check_ans_log_msg_less: 2254, 2258,
 2263
 __enumext_check_ans_log_msg_same_ok: 2254,
 2259, 2268
 __enumext_check_ans_msg_greater: 2230, 2236,
 2249
 __enumext_check_ans_msg_less: 2230, 2234, 2239
 __enumext_check_ans_msg_same_ok: 2230, 2235,
 2244
 __enumext_check_ans_show: .. 74, 90, 2230, 2230,
 3084
 \l__enumext_check_answers_bool 71, 73, 85, 95, 96,
 153, 2117, 2144, 2159, 2491, 2515, 2522, 2546, 2784,
 2981, 3207, 3323, 3357, 4912
 __enumext_check_starred_cmd:n 36, 75, 93, 128,
 2278, 2278, 4051, 4290, 5023
 \g__enumext_check_starred_cmd_int . 100, 153,
 2281, 2287, 2292, 3542, 4358, 5150
 \l__enumext_check_start_line_env_tl . 36, 153,
 316, 324, 332, 2284, 2290, 2293
 \l__enumext_columns_sep_v_dim 3969, 3971, 3979
 \l__enumext_columns_sep_vii_dim .. 4457, 4459,
 4468, 4480, 4556, 4979
 \l__enumext_columns_sep_viii_dim . 4488, 4490,
 4499, 4511, 4605, 5260
 \l__enumext_columns_v_int 1510, 1528, 1699, 3967,
 3975, 3987, 3992
 \l__enumext_columns_vii_int .. 4462, 4465, 4469,
 4478, 4520, 4524, 4527, 4533, 4539, 4543, 4973, 4987
 \l__enumext_columns_viii_int . 4493, 4496, 4500,
 4509, 4569, 4573, 4576, 4582, 4588, 4592, 5254, 5269
 \l__enumext_counter_i_tl 41, 597
 \l__enumext_counter_ii_tl 41, 598
 \l__enumext_counter_iii_tl 41, 599
 \l__enumext_counter_iv_tl 41, 600
 \c__enumext_counter_style_tl 35, 46, 230
 \g__enumext_counter_styles_tl . 30, 43, 63, 608,
 626
 \l__enumext_counter_v_tl 41, 601, 860
 \l__enumext_counter_vi_tl 41, 602
 \l__enumext_counter_vii_tl 41, 603, 794
 \l__enumext_counter_viii_tl 41, 604, 810
 \l__enumext_current_widest_dim 30, 63, 632, 728,
 774, 845
 __enumext_def_meta_key:nnn ... 138, 5495, 5523,
 5529, 5543
 __enumext_default_item:n ... 3319, 3319, 3383
 __enumext_define_counters:Nn 30, 588, 588, 597,
 598, 599, 600, 601, 602, 603, 604
 __enumext_endminipage: . 38, 378, 387, 582, 4671,
 4967, 5248
 \g__enumext_envir_name_tl 36, 30, 287, 302, 361,
 2087, 2092, 2102, 2242, 2247, 2252, 2266, 2271, 2276
 \l__enumext_envir_name_tl 35, 36, 98, 30, 256, 266,
 315, 323, 331, 3461, 4195, 5891, 5894, 5901, 5904,
 5911, 5914, 5921, 5924, 5930, 5934, 5940, 5944, 6001,
 6005
 __enumext_execute_after_env: 37, 71, 74, 75, 86,
 90, 3072, 3072, 3922, 5000
 __enumext_fake_item_indent: . 1056, 1056, 3687
 \l__enumext_fake_item_indent_v_dim 1075, 1080
 \l__enumext_fake_item_indent_v_tl 1077, 3526,
 3530, 3537
 __enumext_fake_item_indent_vii: . 1056, 1085,
 3720
 \l__enumext_fake_item_indent_vii_dim . 1088,
 1092
 \l__enumext_fake_item_indent_vii_tl .. 1090,
 4962
 __enumext_fake_item_indent_viii: 1056, 1097,
 3725
 \l__enumext_fake_item_indent_viii_dim 1100,
 1104
 \l__enumext_fake_item_indent_viii_tl . 1102,
 5239
 \l__enumext_fake_item_indent_X_tl 96
 __enumext_fake_make_label_vii:n . 126, 4897,
 4897, 4954
 __enumext_fake_make_label_viii:n 5180, 5199,
 5231
 __enumext_filter_first_level:n .. 136, 5392,
 5392, 5426, 5437
 __enumext_filter_first_level_key:n 136, 5392,
 5397, 5401
 __enumext_filter_first_level_pair:nn . 136,
 5392, 5398, 5410
 __enumext_filter_save_key:n .. 78, 2407, 2415,
 2438, 2444, 2446, 2446, 5311, 5317, 5323, 5329, 5335,
 5341
 __enumext_filter_save_key_key:n .. 78, 2446,
 2451, 2455
 __enumext_filter_save_key_pair:nn 79, 2446,
 2452, 2463
 __enumext_filter_series:n 66, 1844, 1844, 1882,
 1894, 1899
 __enumext_filter_series_key:n 66, 1844, 1849,
 1853
 __enumext_filter_series_pair:nn .. 66, 1844,
 1850, 1861
 __enumext_first_item_tmp_vii: 122, 124, 4753,
 4826, 4826
 __enumext_first_item_tmp_viii: .. 130, 5015,
 5069, 5069
 \g__enumext_footnote_standar_arg_seq .. 172,
 456, 467, 470

[\g__enumext_footnote_standar_int](#) [172](#), [450](#), [453](#), [455](#), [458](#)
[\g__enumext_footnote_standar_int_seq](#) .. [172](#), [458](#), [463](#), [466](#), [471](#)
[\g__enumext_footnote_starred_arg_seq](#) .. [172](#), [486](#), [497](#), [500](#)
[\g__enumext_footnote_starred_int](#) [172](#), [480](#), [483](#), [485](#), [488](#)
[\g__enumext_footnote_starred_int_seq](#) .. [172](#), [488](#), [493](#), [496](#), [501](#)
[__enumext_footnotes_key_bool](#) [38](#)
[\l__enumext_footnotes_key_bool](#) [33](#), [38](#), [162](#), [399](#), [404](#), [411](#), [512](#), [528](#), [542](#), [555](#)
[__enumext_footnotetext:nn](#) .. [439](#), [439](#), [468](#), [498](#)
[__enumext_foreach_add_body:n](#) . [139](#), [5544](#), [5604](#), [5607](#)
[\l__enumext_foreach_after_tl](#) [5548](#), [5616](#)
[\l__enumext_foreach_before_tl](#) [5546](#), [5611](#)
[\g__enumext_foreach_default_keys_tl](#) [138](#), [122](#), [5566](#), [5587](#)
[__enumext_foreach_keyans:nn](#) .. [139](#), [5544](#), [5583](#), [5585](#)
[\l__enumext_foreach_name_prop_tl](#) . [122](#), [5589](#), [5614](#)
[\l__enumext_foreach_print_seq](#) [122](#), [5599](#), [5605](#), [5609](#)
[\l__enumext_foreach_sep_tl](#) [5558](#), [5605](#)
[\l__enumext_foreach_start_int](#) [5550](#), [5601](#)
[\l__enumext_foreach_step_int](#) [5554](#), [5602](#)
[\l__enumext_foreach_stop_int](#) . [5552](#), [5594](#), [5596](#), [5603](#)
[__enumext_foreach_wrapper:n](#) [5556](#), [5612](#)
[__enumext_getkeyans:nn](#) .. [134](#), [5280](#), [5294](#), [5298](#)
[__enumext_getkeyans_aux:n](#) [134](#), [5280](#), [5282](#), [5285](#)
[\l__enumext_hyperref_bool](#) . [33](#), [38](#), [39](#), [162](#), [396](#), [414](#), [431](#), [2693](#), [3195](#), [4906](#)
[__enumext_hypertarget:nn](#) [39](#), [389](#), [416](#), [420](#), [436](#)
[__enumext_if_is_int:n](#) [222](#)
[__enumext_if_is_int:nTF](#) [222](#), [878](#), [892](#)
[__enumext_internal_mini_page:](#) [41](#), [104](#), [123](#), [569](#), [569](#), [3735](#), [4766](#)
[__enumext_is_not_nested:](#) . [30](#), [35](#), [104](#), [123](#), [250](#), [250](#), [3734](#), [4765](#)
[__enumext_is_on_first_level:](#) . [30](#), [36](#), [104](#), [123](#), [250](#), [276](#), [3741](#), [4778](#)
[\g__enumext_item_anskey_int](#) [85](#), [93](#), [153](#), [348](#), [375](#), [376](#), [2227](#), [2642](#), [3209](#)
[__enumext_item_answer_diff:](#) .. [74](#), [75](#), [90](#), [2223](#), [2223](#), [3079](#)
[\g__enumext_item_answer_diff_int](#) . [74](#), [75](#), [153](#), [349](#), [2225](#), [2232](#), [2256](#)
[\l__enumext_item_column_pos_vii_int](#) [125](#), [4527](#), [4533](#), [4539](#), [4543](#), [4550](#), [4837](#), [4973](#), [4976](#)
[\l__enumext_item_column_pos_viii_int](#) .. [130](#), [4576](#), [4582](#), [4588](#), [4592](#), [4599](#), [5080](#), [5254](#), [5257](#)
[\l__enumext_item_column_pos_X_int](#) [178](#)
[\g__enumext_item_count_all_vii_int](#) [125](#), [4551](#), [4838](#), [4987](#), [4995](#)
[\g__enumext_item_count_all_viii_int](#) [130](#), [4600](#), [5081](#), [5268](#), [5277](#)
[\g__enumext_item_count_all_X_int](#) [178](#)
[\g__enumext_item_number_bool](#) [153](#)
[\l__enumext_item_number_bool](#) [73](#), [160](#), [2177](#), [2182](#), [2186](#), [2190](#), [2203](#), [2827](#), [2881](#), [3326](#), [3360](#), [4915](#)
[\g__enumext_item_number_int](#) .. [73](#), [153](#), [347](#), [374](#), [376](#), [2176](#), [2181](#), [2185](#), [2189](#), [2202](#), [2227](#), [3325](#), [3359](#), [4914](#)
[__enumext_item_peek_args_vii:](#) [125](#), [4834](#), [4839](#), [4841](#)
[__enumext_item_peek_args_viii:](#) .. [130](#), [5077](#), [5082](#), [5084](#)
[__enumext_item_starred_exec:](#) . [96](#), [3338](#), [3365](#), [3407](#), [3428](#)
[__enumext_item_starred_exec:nn](#) .. [3338](#), [3338](#), [3381](#)
[\l__enumext_item_starred_vii_bool](#) [4856](#), [4870](#), [4919](#)
[\l__enumext_item_starred_viii_bool](#) [5099](#), [5114](#), [5209](#), [5240](#)
[\l__enumext_item_starred_X_bool](#) [178](#)
[__enumext_item_std:w](#) [38](#), [95](#), [96](#), [99](#), [378](#), [382](#), [3329](#), [3335](#), [3363](#), [3526](#), [3530](#), [3537](#)
[\g__enumext_item_symbol_aux_tl](#) . [96](#), [126](#), [3343](#), [3346](#), [3371](#), [3415](#), [3435](#)
[\g__enumext_item_symbol_aux_vii_tl](#) [4878](#), [4921](#), [4924](#), [4928](#), [4930](#)
[\g__enumext_item_symbol_aux_X_tl](#) [178](#)
[\l__enumext_item_symbol_sep_vii_dim](#) .. [4886](#), [4893](#), [4927](#), [4929](#)
[\l__enumext_item_symbol_vii_tl](#) [4924](#)
[\l__enumext_item_text_vii_box](#) [4946](#), [4971](#)
[\l__enumext_item_text_viii_box](#) ... [5223](#), [5252](#)
[\l__enumext_item_text_X_box](#) [178](#)
[\l__enumext_item_width_vii_dim](#) ... [4466](#), [4475](#), [4554](#), [4562](#), [4563](#)
[\l__enumext_item_width_viii_dim](#) .. [4497](#), [4506](#), [4603](#), [4611](#), [4612](#)
[\l__enumext_item_width_X_dim](#) [178](#)
[\l__enumext_item_wrap_key_bool](#) [100](#), [153](#), [3279](#), [3299](#), [3550](#), [3557](#), [3584](#), [4352](#), [4370](#), [5100](#), [5115](#), [5186](#)
[\l__enumext_itemindent_X_dim](#) [67](#)
[\l__enumext_itemsep_i_skip](#) ... [1371](#), [1378](#), [1381](#), [1383](#), [1390](#), [1394](#), [1397](#), [1399](#), [1539](#), [1546](#), [1548](#), [1549](#), [1554](#), [1558](#), [1560](#), [1561](#)
[\l__enumext_itemsep_ii_skip](#) .. [1411](#), [1418](#), [1421](#), [1423](#), [1430](#), [1434](#), [1437](#), [1439](#)
[\l__enumext_itemsep_iii_skip](#) . [1450](#), [1457](#), [1460](#), [1462](#), [1469](#), [1473](#), [1476](#), [1478](#)
[\l__enumext_itemsep_vii_skip](#) [4993](#)
[\l__enumext_itemsep_viii_skip](#) [5275](#)
[\l__enumext_joined_item_aux_vii_int](#) .. [4548](#), [4549](#), [4550](#), [4551](#), [4557](#)
[\l__enumext_joined_item_aux_viii_int](#) . [4597](#), [4598](#), [4599](#), [4600](#), [4606](#)
[\l__enumext_joined_item_aux_X_int](#) [178](#)
[__enumext_joined_item_vii:w](#) .. [125](#), [4834](#), [4844](#), [4845](#), [4847](#)
[\l__enumext_joined_item_vii_int](#) .. [4519](#), [4520](#), [4523](#), [4525](#), [4531](#), [4536](#), [4541](#), [4546](#), [4548](#), [4554](#)
[__enumext_joined_item_viii:w](#) . [130](#), [5077](#), [5087](#), [5088](#), [5090](#)
[\l__enumext_joined_item_viii_int](#) . [4568](#), [4569](#), [4572](#), [4574](#), [4580](#), [4585](#), [4590](#), [4595](#), [4597](#), [4603](#)
[\l__enumext_joined_item_X_int](#) [178](#)
[\l__enumext_joined_width_vii_dim](#) . [4552](#), [4559](#), [4562](#), [4948](#), [4956](#)
[\l__enumext_joined_width_viii_dim](#) [4601](#), [4608](#), [4611](#), [5225](#), [5233](#)
[\l__enumext_joined_width_X_dim](#) [178](#)

```

\__enumext_keyans_addto_prop:n 91, 3092, 3092,
    3539, 4355
\__enumext_keyans_addto_seq:n . 92, 3168, 3168,
    3541, 4357
\__enumext_keyans_addto_seq_link: 3168, 3189,
    3191, 5149
\__enumext_keyans_default_item:n .. 99, 3521,
    3521, 3558
\l__enumext_keyans_env_bool 30, 3766, 3779, 3931,
    4021
\__enumext_keyans_fake_item_indent: .. 1056,
    1072, 3677
\l__enumext_keyans_level_h_int .. 129, 24, 803,
    829, 2811, 2873, 3146, 4772, 5032, 5033
\l__enumext_keyans_level_int .. 24, 1659, 2807,
    2869, 3141, 3289, 3930, 3935, 4318
\__enumext_keyans_make_label: . 100, 3562, 3562,
    3675
\__enumext_keyans_make_label_box: 3562, 3566,
    3571, 3608
\__enumext_keyans_make_label_std: 3562, 3574,
    3596
\__enumext_keyans_mini_right_cmd:n 63, 1661,
    1694, 1694
\__enumext_keyans_mini_set_vskip: ..... 60
\__enumext_keyans_minipage_add_space: 1490,
    1516, 3960
\__enumext_keyans_minipage_set_skip: . 1490,
    1490, 1518
\__enumext_keyans_multi_addvspace: 1290, 1301,
    3984
\__enumext_keyans_multi_set_vskip: 56, 1290,
    1290, 1303
\__enumext_keyans_multicols_start: 3948, 3963,
    3965
\__enumext_keyans_multicols_stop: 1698, 3948,
    3990, 4019
\__enumext_keyans_name_and_start: 30, 36, 129,
    309, 309, 3932, 4206, 5037
\__enumext_keyans_parse_keys:n 3944, 3944, 4039
\__enumext_keyans_pic_arg_two: 113, 4199, 4222,
    4253
\l__enumext_keyans_pic_level_int .. 24, 1640,
    2815, 2877, 3095, 3136, 3171, 4201, 4202
\__enumext_keyans_pic_parse_keys:n 4199, 4208,
    4252
\__enumext_keyans_pic_safe_exec: . 113, 4199,
    4199, 4251
\__enumext_keyans_pic_skip_abs:N . 113, 4199,
    4215, 4226
\__enumext_keyans_pos_mark_set: 94, 3245, 3245,
    3282, 3314
\__enumext_keyans_pre_itemsep_skip: .. 1490,
    1509, 1536
\__enumext_keyans_redefine_item: . 100, 3544,
    3544, 3674
\__enumext_keyans_ref: ..... 47, 852, 869, 3676
\__enumext_keyans_ref:n ..... 47, 849, 852, 852
\__enumext_keyans_safe_exec: . 3924, 3924, 4038
\__enumext_keyans_save_item_opt:n 93, 99, 3212,
    3212, 3535, 4354
\__enumext_keyans_set_item_width: 109, 4025,
    4025, 4047
\__enumext_keyans_show_ans: 94, 3245, 3274, 3601,
    3616, 4359
\__enumext_keyans_show_item_opt: 93, 99, 3212,
    3219, 3538, 4367
\__enumext_keyans_show_item_opt_viii: .. 93,
    3212, 3232, 5242
\__enumext_keyans_show_pos: 94, 3245, 3287, 3602,
    3617, 4360
\__enumext_keyans_starred_item:n .. 99, 3533,
    3533, 3553
\__enumext_keyans_starred_item_star: .. 131,
    5112, 5143, 5211
\__enumext_keyans_store_ref: .. 91, 3115, 3115,
    3540, 4356, 5147
\__enumext_keyans_store_ref_aux_i: 92, 3115,
    3127, 3130
\__enumext_keyans_store_ref_aux_ii: 92, 3115,
    3156, 3158
\__enumext_keyans_unknown_keys:n . 3456, 3462,
    3467, 4196
\__enumext_keyans_unknown_keys:nn 3456, 3469,
    3471
\__enumext_keyans_wrapper_label:n ..... 100
\__enumext_keyans_wrapper_label_viii:n 5180,
    5180, 5216
\__enumext_keyans_wrapper_item_v:n 3585, 3588
\__enumext_keyans_wrapper_item_viii:n 5187,
    5191
\__enumext_keyans_wrapper_label:n 3562, 3578,
    3604, 3619, 4364
\__enumext_keyans_wrapper_opt_v:n .... 3227
\__enumext_keyans_wrapper_opt_viii:n .. 3240
\l__enumext_label_copy_i_tl .. 2608, 3134, 3139,
    3144, 3149
\l__enumext_label_copy_v_tl ..... 3144
\l__enumext_label_copy_vi_tl ..... 3139
\l__enumext_label_copy_vii_tl 2584, 2595, 2624,
    3134
\l__enumext_label_copy_viii_tl ..... 3149
\l__enumext_label_copy_X_tl ..... 164
\l__enumext_label_fill_left_v_tl .... 3600
\l__enumext_label_fill_left_X_tl ..... 96
\l__enumext_label_fill_right_v_tl .... 3605
\l__enumext_label_fill_right_X_tl .... 96
\l__enumext_label_font_style_v_tl 3603, 3618,
    4363, 4371
\l__enumext_label_font_style_vii_tl ... 4934
\l__enumext_label_font_style_viii_tl .. 5215
\l__enumext_label_i_tl ..... 720
\l__enumext_label_ii_tl ..... 720
\l__enumext_label_iii_tl ..... 720
\l__enumext_label_iv_tl ..... 720
\__enumext_label_style:Nnn 30, 43, 621, 621, 636,
    725, 771, 840, 842
\l__enumext_label_v_tl 92, 837, 3100, 3176, 3248,
    4042, 4230
\l__enumext_label_vi_tl 92, 837, 3097, 3173, 4364,
    4372
\l__enumext_label_vii_tl . 766, 4865, 4888, 4895
\l__enumext_label_viii_tl 766, 5109, 5141, 5145
\l__enumext_label_width_by_box .. 63, 617, 618
\__enumext_label_width_by_box:Nn 43, 615, 615,
    620, 632, 902, 3247
\l__enumext_labelsep_v_dim ... 3268, 3974, 4242,
    4366
\l__enumext_labelsep_vii_dim . 2710, 4461, 4471,

```

4555, 4830, 4886, 4941, 4950
 \l__enumext_labelsep_viii_dim 4492, 4502, 4604,
 5073, 5153, 5218, 5227
 \l__enumext_labelwidth_v_dim . 845, 3258, 3263,
 3284, 3316, 3614, 3974, 4242, 4361
 \l__enumext_labelwidth_vii_dim ... 2713, 4461,
 4470, 4555, 4830, 4932, 4949
 \l__enumext_labelwidth_viii_dim .. 4492, 4501,
 4604, 5073, 5160, 5177, 5213, 5226
 \l__enumext_leftmargin_tmp_v_bool . 113, 4224
 \l__enumext_leftmargin_tmp_X_bool 67
 \l__enumext_leftmargin_tmp_X_dim 67
 \l__enumext_leftmargin_X_dim 67
 __enumext_level: 218, 218, 749, 752, 753, 761, 763,
 1059, 1063, 1067, 1135, 1139, 1143, 1147, 1230, 1232,
 1234, 1236, 1278, 1280, 1282, 1284, 1288, 1322, 1328,
 1333, 1335, 1338, 1341, 1354, 1357, 1668, 1672, 1678,
 1741, 1743, 1745, 1748, 1755, 1757, 1759, 1762, 2402,
 2404, 2406, 2434, 2435, 2437, 2493, 2501, 2505, 2509,
 2720, 2724, 3328, 3329, 3333, 3334, 3335, 3343, 3351,
 3352, 3355, 3362, 3363, 3367, 3370, 3372, 3406, 3408,
 3409, 3411, 3414, 3425, 3426, 3429, 3430, 3432, 3772,
 3785, 3792, 3800, 3803, 3805, 3807, 3808, 3809, 3810,
 3813, 3818, 3824, 3830, 3837, 3850, 3852, 3855, 3856,
 3858, 3862, 3868, 3893, 3898, 3909, 3911
 \l__enumext_level_h_int 123, 24, 259, 282, 296, 787,
 822, 1647, 2173, 2193, 2603, 2847, 2859, 3780, 4767,
 4768
 \l__enumext_level_int 104, 24, 220, 269, 281, 297,
 571, 1242, 1367, 1646, 2167, 2199, 2580, 2590, 2596,
 2602, 2609, 2618, 2623, 2846, 2858, 3074, 3691, 3736,
 3737, 3748, 3756, 3770, 3783, 3814, 3939, 4314, 4810,
 4820, 5045, 5931, 5935, 5941, 5945
 __enumext_list_arg_two_i: 3656
 __enumext_list_arg_two_ii: 3656
 __enumext_list_arg_two_iii: 3656
 __enumext_list_arg_two_iv: 3656
 __enumext_list_arg_two_v: 100, 3656, 4044, 4225
 __enumext_list_arg_two_vii: 3697, 4747
 __enumext_list_arg_two_viii: 3697, 5009
 \l__enumext_listoffset_v_dim . 3976, 4030, 4033
 \l__enumext_listparindent_vii_dim 4957, 4961
 \l__enumext_listparindent_viii_dim 5234, 5238
 __enumext_log_answer_vars: . 37, 363, 371, 3081
 __enumext_log_global_vars: . 37, 363, 363, 3080
 __enumext_make_label: 97, 3386, 3386, 3685
 __enumext_make_label_box: ... 3386, 3390, 3395,
 3418
 __enumext_make_label_std: ... 3386, 3398, 3402
 \l__enumext_mark_answer_sym_tl 80, 2346, 2559,
 2736, 3270, 5157, 5164
 \l__enumext_mark_answer_sym_v_tl . 3270, 3302
 \l__enumext_mark_answer_sym_viii_tl ... 5157
 \l__enumext_mark_position_str 126, 2352, 2353,
 2354, 2557, 3272, 5158, 5175
 \l__enumext_mark_position_v_str .. 126, 3272
 \l__enumext_mark_position_viii_str 126, 5158,
 5175
 \l__enumext_mark_ref_sym_tl .. 2334, 2698, 3203
 \l__enumext_mark_sep_tmpa_dim 126, 3248, 3258,
 3263
 \l__enumext_mark_sep_tmpb_dim 126, 3253, 3257,
 3262, 3271
 \l__enumext_mark_sym_sep_dim . 2349, 2708, 2710,
 2713, 2716, 2718
 \l__enumext_mark_sym_sep_v_dim ... 3266, 3268,
 3271, 3284, 3316
 \l__enumext_mark_sym_sep_viii_dim 5151, 5153,
 5160, 5177
 \l__enumext_meta_path_tl . 122, 5519, 5520, 5522,
 5523
 \c__enumext_meta_paths_prop 138, 5495
 __enumext_mini_addvspace_vii: 62, 1626, 1626,
 4629
 __enumext_mini_addvspace_viii: 62, 1626, 1632,
 4694
 __enumext_mini_env* 569
 __enumext_mini_page 1678, 1705, 3862, 3961, 4631,
 4696, 4717
 __enumext_mini_right_cmd:n 63, 1663, 1665, 1665
 __enumext_mini_set_vskip_vii: 61, 1569, 1569,
 1628
 __enumext_mini_set_vskip_viii: 61, 1569, 1591,
 1634
 __enumext_minipage:w 38, 378, 386, 576, 4654, 4956,
 5233
 \l__enumext_minipage_active_v_bool 3958, 3981,
 4006
 \g__enumext_minipage_active_vii_bool .. 121,
 4643, 4652, 4674
 \l__enumext_minipage_active_vii_bool . 4625,
 4636
 \g__enumext_minipage_active_viii_bool 4707,
 4715, 4734
 \l__enumext_minipage_active_viii_bool 4690,
 4701
 \g__enumext_minipage_active_X_bool ... 178
 \l__enumext_minipage_active_X_bool 83
 __enumext_minipage_add_space: . 57, 106, 1318,
 1344, 3860
 \g__enumext_minipage_after_skip 83, 1573, 1585,
 4672, 4732
 \l__enumext_minipage_after_skip .. 57, 107, 83,
 1331, 1371, 1373, 1378, 1381, 1385, 1390, 1394, 1397,
 1401, 1413, 1418, 1421, 1425, 1430, 1434, 1437, 1441,
 1452, 1457, 1460, 1464, 1469, 1473, 1476, 1480, 1492,
 1506, 1539, 1541, 1546, 1548, 1550, 1554, 1558, 1560,
 1562, 1593, 1606, 1620, 1674, 1701, 4016
 \g__enumext_minipage_center_vii_bool . 4658,
 4675
 \g__enumext_minipage_center_viii_bool 4719,
 4735
 \g__enumext_minipage_center_X_bool ... 178
 \l__enumext_minipage_hsep_v_dim 3956
 \l__enumext_minipage_hsep_vii_dim 4623
 \l__enumext_minipage_hsep_viii_dim ... 4688
 \l__enumext_minipage_left_skip 83, 1493, 1571,
 1576, 1580, 1594, 1598, 1612, 1630, 1636
 \l__enumext_minipage_left_v_dim .. 3954, 3961
 \l__enumext_minipage_left_vii_dim 4619, 4631
 \l__enumext_minipage_left_viii_dim 4684, 4696
 \l__enumext_minipage_left_X_dim 83
 \g__enumext_minipage_right_skip 83, 1572, 1577,
 1581, 4657, 4718
 \l__enumext_minipage_right_skip . 57, 83, 1320,
 1326, 1331, 1333, 1335, 1494, 1495, 1501, 1506, 1507,
 1508, 1513, 1595, 1602, 1616, 1680, 1707
 \l__enumext_minipage_right_v_dim . 1696, 1705,
 3952, 3956

`\g__enumext_minipage_right_vii_dim` 120, 4627, 4654, 4677
`\l__enumext_minipage_right_vii_dim` 120, 4617, 4622, 4628
`\g__enumext_minipage_right_viii_dim` .. 4692, 4717, 4737
`\l__enumext_minipage_right_viii_dim` .. 4682, 4687, 4693
`\g__enumext_minipage_right_X_dim` 178
`\g__enumext_minipage_right_X_skip` 178
`__enumext_minipage_set_skip:` . 57, 1318, 1318, 1346
`\g__enumext_minipage_stat_int` .. 106, 83, 1685, 1712, 3859, 3870, 3875, 3959, 4008, 4013
`\l__enumext_minipage_temp_skip` 83, 1392, 1402, 1405, 1432, 1442, 1445, 1471, 1481, 1484, 1556, 1563, 1565
`\l__enumext_miniright_code_vii_box` 4665, 4669
`\g__enumext_miniright_code_vii_tl` 121, 4660, 4667, 4676
`\l__enumext_miniright_code_viii_box` .. 4726, 4730
`\g__enumext_miniright_code_viii_tl` 4721, 4728, 4736
`\l__enumext_miniright_code_X_box` 178
`\l__enumext_mode_box_bool` 641, 3393, 3569
`__enumext_multi_addvspace:` 56, 105, 1273, 1273, 3821
`__enumext_multi_set_vskip:` 55, 1228, 1228, 1275
`\l__enumext_multicols_above_ii_skip` ... 1247
`\l__enumext_multicols_above_iii_skip` .. 1256
`\l__enumext_multicols_above_iv_skip` ... 1265
`\l__enumext_multicols_above_v_skip` 1292, 1306, 1316, 1507
`\l__enumext_multicols_above_X_skip` 75
`\l__enumext_multicols_below_ii_skip` .. 1374, 1383, 1387, 1399, 1404
`\l__enumext_multicols_below_iii_skip` . 1414, 1423, 1427, 1439, 1444
`\l__enumext_multicols_below_iv_skip` .. 1453, 1462, 1466, 1478, 1483
`\l__enumext_multicols_below_v_skip` 1296, 1310, 1508, 1542, 1549, 1551, 1561, 1564, 3998
`\l__enumext_multicols_below_X_skip` 75
`\g__enumext_multicols_right_X_skip` 75
`__enumext_multicols_start:` 105, 106, 3797, 3797, 3864
`__enumext_multicols_stop:` 106, 1670, 3827, 3827, 3880
`__enumext_nested_base_line_fix:` 50, 104, 976, 982, 3752
`__enumext_newlabel:nn` 33, 39, 82, 424, 424, 2634, 3162
`\l__enumext_newlabel_arg_one_tl` 33, 39, 82, 92, 164, 2627, 2635, 2697, 3151, 3163, 3201
`\l__enumext_newlabel_arg_two_tl` 33, 39, 81, 164, 2583, 2593, 2606, 2621, 2636, 3138, 3143, 3148, 3164
`__enumext_parse_foreach_keys:n` .. 5544, 5560, 5577
`__enumext_parse_foreach_keys:nn` . 5544, 5567, 5579
`__enumext_parse_keys:n` 50, 67, 3743, 3743, 3905
`__enumext_parse_keys_vii:n` 67, 4742, 4780, 4780
`__enumext_parse_keys_viii:n` . 5005, 5050, 5050
`__enumext_parse_save_key:n` 78, 2427, 2432, 2432
`__enumext_parse_save_key_vii:n` 78, 2422, 2432, 2440
`__enumext_parse_series:n` .. 67, 104, 123, 1870, 1870, 3751, 4786
`__enumext_parse_store_keys:n` 104
`\l__enumext_parsep_i_skip` 1245, 1249
`\l__enumext_parsep_ii_skip` 1254, 1258
`\l__enumext_parsep_iii_skip` 1263, 1267
`\l__enumext_parsep_vii_skip` 4958
`\l__enumext_parsep_viii_skip` 5235
`\l__enumext_partopsep_v_skip` . 1308, 1312, 1503, 1526
`\l__enumext_partopsep_viii_skip` 1604
`__enumext_phantomsection:` 39, 389, 417, 421, 437
`__enumext_pre_itemsep_skip:` 57, 58, 1336, 1365, 1365
`__enumext_print_footnote:` .. 439, 461, 525, 530
`__enumext_print_footnote_mini:` 439, 491, 552, 557
`__enumext_print_footnote_standar:` 503, 519, 583
`__enumext_print_footnote_starred:` 503, 548, 563, 567
`__enumext_print_keyans_box:NN` 80, 2551, 2551, 2564, 2712, 2723, 3283, 3315, 5159, 5176
`\l__enumext_print_keyans_i_tl` 5318, 5350
`\l__enumext_print_keyans_ii_tl` ... 5324, 5351
`\l__enumext_print_keyans_iii_tl` .. 5330, 5352
`\l__enumext_print_keyans_iv_tl` ... 5336, 5353
`\l__enumext_print_keyans_star_bool` . 50, 135, 126, 988, 1000, 5373, 5378
`\l__enumext_print_keyans_starred_tl` 134, 135, 126, 5312, 5371
`\l__enumext_print_keyans_vii_tl` 134, 5342, 5354
`\l__enumext_print_keyans_X_tl` 126
`__enumext_printkeyans:nnn` 135, 5347, 5355, 5358
`__enumext_redefine_item:` . 96, 3375, 3375, 3684
`\l__enumext_ref_key_arg_tl` 45, 46, 233, 742, 743, 755, 786, 789, 799, 805, 815, 854, 855, 865
`\l__enumext_ref_the_count_tl` . 45, 46, 749, 752, 755, 794, 796, 799, 810, 812, 815, 860, 862, 865
`__enumext_regex_counter_style:` .. 35, 45, 228, 228, 750, 795, 811, 861
`__enumext_register_counter_style:Nn` .. 605, 605, 610, 611, 612, 613, 614
`__enumext_remove_extra_parsep_vii:` .. 4760, 4982, 4982
`__enumext_remove_extra_parsep_viii:` . 5022, 5263, 5263
`__enumext_renew_footnote:` .. 439, 443, 509, 514
`__enumext_renew_footnote_mini:` 439, 473, 539, 544
`__enumext_renew_footnote_standar:` 503, 503, 575
`__enumext_renew_footnote_starred:` 503, 535, 4952, 5229
`\l__enumext_renew_the_count_v_tl` 863, 871, 873
`\l__enumext_renew_the_count_vii_tl` 797, 824, 826
`\l__enumext_renew_the_count_viii_tl` 813, 831, 833
`\l__enumext_renew_the_count_X_tl` 46
`__enumext_rescan_anskey_env:n` .. 88, 90, 2890, 2956, 3051, 3059


```

__enumext_reset_global_bool: . . 339, 342, 351
__enumext_reset_global_int: . . . 339, 341, 345
__enumext_reset_global_tl: . . . . 339, 343, 357
__enumext_reset_global_vars: . 37, 90, 339, 339,
    3089
__enumext_resume_active_bool 67, 69, 57, 1874,
    1994
__enumext_resume_counter: . . 68, 69, 1992, 1998,
    2005
__enumext_resume_counter:n . 67, 69, 1963, 1968,
    1992, 1992, 2062, 2070
__enumext_resume_counter_save_ans: . . 69, 70,
    1992, 2003, 2035
__enumext_resume_counter_series: . 69, 1992,
    2001, 2018
__enumext_resume_int . . . 57, 1915, 2009, 2010
__enumext_resume_last:n . . 67, 1870, 1876, 1889
__enumext_resume_name_tl 57, 1911, 1919, 1922,
    1938, 1946, 1949, 1995, 1996, 2024, 2031
__enumext_resume_save_counter: . 67, 107, 124,
    1902, 1902, 3886, 4804
__enumext_resume_series:n . 68, 1838, 1959, 1959
__enumext_resume_starred: . 70, 1839, 2056, 2056
__enumext_resume_vii_int 57, 1942, 2014, 2015
__enumext_rightmargin_vii_dim . . 4473, 4477,
    4482
__enumext_rightmargin_viii_dim . 4504, 4508,
    4513
__enumext_safe_exec: . . 41, 104, 3732, 3732, 3904
__enumext_safe_exec_vii: . 41, 4741, 4763, 4763
__enumext_safe_exec_viii: 129, 5004, 5026, 5026
__enumext_second_part: . . 107, 3866, 3866, 3918
__enumext_second_part_v: . . . 3948, 4004, 4052
__enumext_series_name_tl . . . . . 69
__enumext_series_str . 67, 104, 123, 1836, 1872,
    1880, 1881, 1883, 1885, 1906, 1909, 1913, 1933, 1936,
    1940, 3747, 4784
__enumext_set_error:nn . . . . . 5454, 5491, 5493
__enumext_set_item_width: 107, 3888, 3888, 3914
__enumext_set_parse:n . . . . . 5454, 5465, 5481
__enumext_setkey_tmpa_int . . . 117, 5458, 5462
__enumext_setkey_tmpa_seq . . . 117, 5456, 5466,
    5472, 5474, 5476, 5488
__enumext_setkey_tmpa_tl . . . . 117, 5464, 5468
__enumext_setkey_tmpb_seq . . . 117, 5457, 5460,
    5464, 5465
__enumext_setkey_tmpb_tl 117, 5483, 5485, 5486
__enumext_show_answer_bool . 2321, 2340, 2730,
    3224, 3237, 3278, 3583, 5155, 5185
__enumext_show_length:nnn . . 52, 236, 236, 5702,
    5703, 5704, 5705, 5706, 5707, 5708, 5709, 5710, 5711,
    5717, 5718, 5719, 5720, 5721, 5722, 5723, 5724, 5725,
    5726
__enumext_show_pos_tmp_int . 126, 3291, 3294,
    3309
__enumext_show_position_bool . . . 2324, 2343,
    2734, 3225, 3238, 3298, 5162
__enumext_standar_bool 35, 104, 30, 258, 261, 280,
    354, 505, 521, 1904, 1969, 1981, 2007, 2020, 2058,
    2198, 2212, 2588, 2601, 2616, 3767
__enumext_standar_bool 104, 107, 30, 1654, 2589,
    3739, 3885, 4777
__enumext_standar_first_bool 36, 104, 30, 285,
    1891, 2038, 2100, 2107
__enumext_standar_item_vii:w . 125, 4834, 4852,
    4854
__enumext_standar_item_viii:w 130, 5077, 5095,
    5097
__enumext_standar_ref: . . . . 45, 740, 759, 3686
__enumext_standar_ref:n . . . . 45, 732, 740, 740
__enumext_standar_series_tl . 57, 1893, 1894,
    2060, 2063
__enumext_standar_unknown_keys:n 3499, 3503,
    3507
__enumext_standar_unknown_keys:nn 3499, 3509,
    3511
__enumext_starred_bool 35, 123, 30, 268, 271, 295,
    355, 1653, 1931, 1974, 1985, 2012, 2027, 2066, 2172,
    2218, 2579, 3132, 4678
__enumext_starred_bool 123, 124, 129, 30, 2617,
    2652, 2658, 2706, 3005, 3010, 3740, 4776, 4803, 5038,
    5042
__enumext_starred_columns_set_vii: . . 4455,
    4455, 4751
__enumext_starred_columns_set_viii: . 4455,
    4486, 5013
__enumext_starred_first_bool 36, 123, 30, 300,
    986, 999, 1896, 2047, 2100, 2107
__enumext_starred_item_vii:w . 125, 4834, 4851,
    4868
__enumext_starred_item_vii_aux_i:w . . 4834,
    4873, 4876
__enumext_starred_item_vii_aux_ii:w . 4834,
    4874, 4879, 4881
__enumext_starred_item_vii_aux_iii:w 4834,
    4884, 4891
__enumext_starred_item_viii:w 130, 131, 5094,
    5112, 5112
__enumext_starred_item_viii_aux_i:w . . 131,
    5112, 5118, 5121
__enumext_starred_item_viii_aux_ii:w . 131,
    5112, 5119, 5136, 5138
__enumext_starred_joined_item_vii:n 119, 125,
    4517, 4517, 4849
__enumext_starred_joined_item_viii:n . 119,
    130, 4517, 4566, 5092
__enumext_starred_ref: . . . . 46, 784, 820, 3717
__enumext_starred_ref:n . . . . 46, 778, 784, 784
__enumext_starred_series_tl . 57, 1898, 1899,
    2068, 2071
__enumext_starred_unknown_keys:n 3481, 3483,
    3485
__enumext_starred_unknown_keys:nn 3481, 3487,
    3489
__enumext_start_from:NNn 48, 876, 876, 889, 911,
    917
__enumext_start_i_int . . . . . 2010, 2022, 2041
__enumext_start_item_tmp_vii: 122, 4754, 4834,
    4834
__enumext_start_item_tmp_viii: . . 5016, 5077,
    5077
__enumext_start_item_vii:w 125, 127, 4860, 4865,
    4888, 4895, 4943, 4943
__enumext_start_item_viii:w . . 130, 5104, 5109,
    5141, 5220, 5220
__enumext_start_line_tl 36, 30, 288, 303, 360,
    2242, 2247, 2252, 2266, 2271, 2276

```

```

\__enumext_start_list:nn 38, 101, 378, 380, 3908,
4041, 4745, 5007
\__enumext_start_list_tag:n .. 4054, 4080, 4953,
5230
\__enumext_start_mini_vii: 124, 4615, 4615, 4795
\__enumext_start_mini_viii: ... 129, 4680, 4680,
5061
\__enumext_start_save_ans_msg: 71, 2084, 2084,
2109
\__enumext_start_store_level: 104, 3761, 3761,
3907
\__enumext_start_store_level_vii: 124, 4744,
4806, 4806
\l__enumext_start_vii_int ... 2015, 2029, 2050
\l__enumext_start_X_int ..... 96
\__enumext_stop_item_tmp_vii: 122, 124, 125, 127,
4753, 4759, 4836, 4945
\__enumext_stop_item_tmp_viii: 130, 5015, 5021,
5079, 5222
\__enumext_stop_item_vii: 127, 4943, 4945, 4965
\__enumext_stop_item_viii: ... 5220, 5222, 5246
\__enumext_stop_list: 38, 121, 124, 378, 381, 3832,
3840, 3994, 4001, 4638, 4646, 4703, 4710
\__enumext_stop_list_tag:n ... 4054, 4096, 4968,
5249
\__enumext_stop_mini_vii: 121, 124, 4615, 4634,
4799
\__enumext_stop_mini_viii: 129, 4680, 4699, 5065
\__enumext_stop_save_ans_msg: . 71, 2084, 2089,
3078
\__enumext_stop_start_list_tag: .. 4054, 4088,
4955, 5232
\__enumext_stop_store_level: .. 105, 106, 3790,
3790, 3833, 3841
\__enumext_stop_store_level_vii: .. 121, 124,
4639, 4647, 4806, 4816
\l__enumext_store_active_bool 32, 71, 108, 2039,
2048, 2116, 2803, 3765, 3778, 3926, 3934, 4310, 4808,
4818, 5028, 5044
\__enumext_store_active_keys:n 77, 78, 104, 2400,
2400, 3758
\__enumext_store_active_keys_vii:n 77, 78, 123,
2400, 2410, 4787
\__enumext_store_addto_prop:n 79, 91, 2475, 2475,
2483, 2643, 3113, 5146
\__enumext_store_addto_seq:n 79, 93, 2484, 2484,
2488, 2495, 2509, 2517, 2526, 2540, 2548, 2701, 3206
\l__enumext_store_anskey_arg_tl 32, 82, 83, 108,
2649, 2654, 2656, 2661, 2668, 2671, 2681, 2686, 2689,
2695, 2701
\__enumext_store_anskey_code:n 82, 85, 90, 2640,
2640, 2796, 3049, 3057
\l__enumext_store_anskey_env_tl .. 32, 88, 108,
2979, 2983, 2989, 3051, 3059
\l__enumext_store_anskey_opt_tl .. 32, 89, 108,
2980, 3007, 3013, 3020, 3026, 3036, 3046, 3055
\__enumext_store_anskey_safe_outer: .... 85
\g__enumext_store_columns_break_bool . 2903,
3004, 3066
\l__enumext_store_columns_break_bool . 2651,
2752
\l__enumext_store_current_label_tl 32, 91-93,
131, 108, 3094, 3097, 3100, 3106, 3111, 3113, 3170,
3173, 3176, 3182, 3187, 3197, 3206, 5123, 5128, 5132,
5145, 5146, 5148
\l__enumext_store_current_label_tmp_tl . 32,
108
\l__enumext_store_current_opt_arg_tl . 32, 93,
131, 108, 3216, 3221, 3228, 3234, 3241, 5134
\__enumext_store_internal_ref: .. 81, 82, 2565,
2565, 2646
\g__enumext_store_item_join_int .. 2906, 3011,
3015, 3067
\l__enumext_store_item_join_int .. 2659, 2663,
2755
\g__enumext_store_item_star_bool . 2908, 3018,
3068
\l__enumext_store_item_star_bool . 2666, 2757
\g__enumext_store_item_symbol_sep_dim 2913,
3033, 3038, 3070
\l__enumext_store_item_symbol_sep_dim 2678,
2683, 2762
\g__enumext_store_item_symbol_tl . 2911, 3024,
3028, 3069
\l__enumext_store_item_symbol_tl . 2669, 2673,
2760
\l__enumext_store_keyans_item_opt_sep_v_-
tl ..... 3104, 3108, 3180, 3184
\l__enumext_store_keyans_item_opt_sep_-
viii_tl ..... 5126, 5130
\__enumext_store_level_close: . 79, 2489, 2513,
3794
\__enumext_store_level_close_vii: . 80, 2520,
2544, 4822
\__enumext_store_level_open: 79, 105, 2489, 2489,
3773, 3786
\__enumext_store_level_open_vii: .. 80, 2520,
2520, 4812
\g__enumext_store_name_tl 32, 71, 108, 359, 366,
367, 368, 369, 2092, 2118, 2241, 2246, 2251, 2265,
2270, 2275, 3076
\l__enumext_store_name_tl 32, 71, 73, 108, 1925,
1928, 1952, 1955, 2043, 2052, 2087, 2096, 2097, 2118,
2119, 2120, 2122, 2123, 2125, 2127, 2128, 2130, 2132,
2133, 2157, 2477, 2479, 2486, 2629, 2630, 2742, 2985,
3153, 3154, 3308, 5170
\l__enumext_store_ref_key_bool 82, 2337, 2644,
2692, 3117, 3194
\l__enumext_store_save_key_vii_bool .. 2412,
2442
\l__enumext_store_save_key_vii_tl 2414, 2415,
2443, 2444, 2524, 2532, 2536, 2540
\l__enumext_store_save_key_X_bool .. 77, 126
\l__enumext_store_save_key_X_tl .. 77, 78, 126
\l__enumext_store_upper_level_X_bool .. 126
\__enumext_storing_exec: 71, 87, 2094, 2110, 2114
\__enumext_storing_set:n .. 71, 2079, 2094, 2094
\l__enumext_the_counter_v_tl ..... 862
\l__enumext_the_counter_vii_tl ..... 796
\l__enumext_the_counter_viii_tl ..... 812
\l__enumext_the_counter_X_tl ..... 46
\__enumext_tmp:n 41, 45, 50, 56, 67, 74, 75, 82, 90, 95,
96, 107, 135, 142, 167, 171, 178, 198, 637, 646, 1832,
1843, 2075, 2083, 2136, 2154, 2330, 2399, 2418, 2431,
2567, 2574, 2575, 2596, 2609, 2612, 2623, 3119, 3126,
3456, 3466, 3499, 3506, 3656, 3696, 3697, 3731
\__enumext_tmp:nn 647, 668, 669, 703, 704, 719, 906,
931, 1012, 1034, 1035, 1055, 1109, 1117, 1118, 1132,
1197, 1213, 1214, 1227, 1721, 1737, 2295, 2329, 3440,
3455

```


<code>__enumext_tmp:nnn</code>	720, 736, 737, 738, 739, 766, 782, 783
<code>__enumext_tmp:nnnnn</code>	932, 957, 960, 963, 965, 967, 970, 973
<code>__enumext_tmp:w</code>	5291, 5294
<code>\l__enumext_tmpa_vii_int</code>	4465, 4468, 4477, 4508
<code>\l__enumext_tmpa_viii_int</code>	4496, 4499
<code>\l__enumext_tmpa_X_dim</code>	178
<code>\l__enumext_tmpa_X_int</code>	178
<code>\l__enumext_topsep_v_skip</code>	1294, 1298, 1497, 4303
<code>\l__enumext_topsep_vii_skip</code>	1574, 1583, 1587
<code>\l__enumext_topsep_viii_skip</code>	1596, 1618, 1622
<code>__enumext_undefine_anskey_env:</code>	86, 90, 2836, 2836, 3087
<code>__enumext_unskip_unkern:</code>	35, 242, 242, 1347, 1519, 3835, 3836, 3876, 3996, 3997, 4014, 4959, 4960, 5236, 5237
<code>\l__enumext_vspace_a_star_v_bool</code>	1770
<code>\l__enumext_vspace_a_star_vii_bool</code>	1792
<code>\l__enumext_vspace_a_star_viii_bool</code>	1803
<code>\l__enumext_vspace_a_star_X_bool</code>	96
<code>__enumext_vspace_above:</code>	64, 106, 1738, 1738, 3846
<code>__enumext_vspace_above_v:</code>	65, 1766, 1766, 3950
<code>\l__enumext_vspace_above_v_skip</code>	1768, 1772, 1774
<code>__enumext_vspace_above_vii:</code>	65, 124, 1788, 1788, 4792
<code>\l__enumext_vspace_above_vii_skip</code>	1790, 1794, 1796
<code>__enumext_vspace_above_viii:</code>	65, 1788, 1799, 5059
<code>\l__enumext_vspace_above_viii_skip</code>	1801, 1805, 1807
<code>\l__enumext_vspace_b_star_v_bool</code>	1781
<code>\l__enumext_vspace_b_star_vii_bool</code>	1814
<code>\l__enumext_vspace_b_star_viii_bool</code>	1825
<code>\l__enumext_vspace_b_star_X_bool</code>	96
<code>__enumext_vspace_below:</code>	64, 107, 1752, 1752, 3884
<code>__enumext_vspace_below_v:</code>	65, 1777, 1777, 4023
<code>\l__enumext_vspace_below_v_skip</code>	1779, 1783, 1785
<code>__enumext_vspace_below_vii:</code>	65, 124, 1810, 1810, 4802
<code>\l__enumext_vspace_below_vii_skip</code>	1812, 1816, 1818
<code>__enumext_vspace_below_viii:</code>	65, 1810, 1821, 5067
<code>\l__enumext_vspace_below_viii_skip</code>	1823, 1827, 1829
<code>__enumext_widest_from:nNNn</code>	48, 890, 890, 905, 924
<code>\g__enumext_widest_label_tl</code>	30, 43, 63, 625, 629, 633
<code>\l__enumext_wrap_label_opt_v_bool</code>	3529
<code>\l__enumext_wrap_label_opt_vii_bool</code>	125, 4859
<code>\l__enumext_wrap_label_opt_viii_bool</code>	130, 5103
<code>\l__enumext_wrap_label_opt_X_bool</code>	96
<code>\l__enumext_wrap_label_v_bool</code>	3525, 3529, 3536, 3582, 3590, 4353
<code>\l__enumext_wrap_label_vii_bool</code>	125, 4859, 4863, 4871, 4935
<code>\l__enumext_wrap_label_viii_bool</code>	130, 5103, 5107, 5116, 5184, 5193
<code>\l__enumext_wrap_label_X_bool</code>	96
<code>__enumext_wrapper_label_v:n</code>	3588, 3592, 4372
<code>__enumext_wrapper_label_vii:n</code>	4937
<code>__enumext_wrapper_label_viii:n</code>	5191, 5195
<code>\l__enumext_write_aux_file_tl</code>	33, 82, 92, 164, 2632, 2638, 3160, 3166
<code>enumext*</code>	5, 4739
<code>enumXi</code>	588
<code>enumXii</code>	588
<code>enumXiii</code>	588
<code>enumXiv</code>	588
<code>enumXv</code>	588
<code>enumXvi</code>	588
<code>enumXvii</code>	588
<code>enumXviii</code>	588
Environments provide by <code>enumext</code> :	
<code>anskey*</code>	32, 71, 76, 78, 81, 83, 84, 86–88, 90, 104, 105, 124, 133, 135, 140, 142
<code>enumext*</code>	29, 30, 33–35, 39–43, 46, 48–52, 54, 61, 62, 65–68, 70–73, 76–82, 84–86, 89, 91, 92, 97, 98, 103–105, 110, 118, 119, 121, 122, 124, 126–130, 132–136, 138, 141, 144, 146
<code>enumext</code>	29, 30, 34, 35, 39–43, 45–50, 52–57, 60, 62–64, 66–68, 70–73, 76–79, 81, 82, 84–86, 89, 91, 92, 95–99, 101, 102, 105, 107, 108, 113, 118, 120, 123, 124, 126, 129, 134–136, 138, 141, 142, 144
<code>keyans*</code>	29, 30, 32–36, 39–42, 46–52, 54, 61, 62, 65, 71, 72, 75, 76, 79, 86, 91, 93, 98, 103, 110, 118, 120, 128, 129, 141, 143, 146
<code>keyanspic</code>	29, 30, 32, 33, 36, 42, 47, 71, 72, 75, 79, 86, 91–93, 98, 110–115, 117, 143
<code>keyans</code>	29, 30, 32, 33, 35, 36, 39, 40, 42, 43, 47–50, 52, 54, 56, 60, 62–65, 71, 72, 75, 76, 79, 86, 91–94, 98–102, 108, 110, 112, 113, 116, 120, 130, 141, 143
Environments:	
<code>center</code>	118
<code>description</code>	97, 118
<code>enumerate</code>	118
<code>flushleft</code>	118
<code>flushright</code>	118
<code>itemize</code>	118
<code>list</code>	34, 38, 84, 97, 101, 106, 107, 110, 112–114, 118, 121
<code>lrbox</code>	127
<code>minipage</code>	34, 38, 40, 41, 54, 56–58, 112, 115, 117, 118, 121, 127
<code>multicols</code>	54–58, 63, 105–107
<code>quotation</code>	118
<code>quote</code>	118
<code>scontents</code>	87, 89
<code>tabbing</code>	118
<code>trivlist</code>	118
<code>verbatim</code>	118
<code>verse</code>	118
exp commands:	
<code>\exp_after:wN</code>	5294
<code>\exp_args:Ne</code>	3048, 3056, 3755, 5282
<code>\exp_args:NV</code>	2768, 2923, 3469, 3487, 3509, 5579
<code>\exp_not:N</code>	54, 628, 755, 799, 815, 865, 1065, 1068, 1079, 1080, 1081, 1092, 1093, 1104, 1105, 2697, 2739, 2740, 3199, 3305, 3306, 5167, 5168, 5291
<code>\exp_not:n</code>	290, 305, 318, 326, 334, 694, 714, 755, 799, 815, 865, 1066, 1859, 1868, 2308, 2357, 2461, 2473, 2635, 2663, 2673, 2683, 2697, 2698, 3015, 3028, 3038, 3163, 3201, 3203, 4167, 5408, 5418, 5611, 5616

F

\fbox 2364

\fboxrule 2364

\fboxsep 2364

file commands:

 \file_input_stop: 6015

first 1118

font 647

\footnote 39

\footnote 39, 445, 475

\footnotemark 455, 485

\footnotesize 2740, 3306, 5168

\footnotetext 441

\foreachkeyans 18, 138, 5544

G

\getkeyans 18, 133, 5280

group commands:

 \group_begin: 2738, 2783, 2958, 3045, 3304, 5166, 5349

 \group_end: 2745, 2799, 3062, 3312, 5173, 5356

H

\hbadness 4970, 5251

hbox commands:

 \hbox_overlap_left:n 2555, 3371, 4928

 \hbox_set:Nn 617, 4230

 \hbox_set_end: 4969, 5250

 \hbox_set_to_wd:Nnw 4946, 5223

\hfill 677, 682, 688, 689, 1677, 1704, 2697, 3199, 4642, 4706

hook commands:

 \hook_gput_code:nnn 5, 208, 212, 216, 389

 \hook_gremove_code:nn 89, 2974

 \hook_gset_rule:nnnn 390

 \hook_if_empty:nTF 2972

\hyperlink 83, 93

\hyperlink 2697, 3199

\hypertarget 39

\hypertarget 416

I

\IfDocumentMetadataTF .. 507, 523, 537, 550, 3388, 3564, 4082, 4090, 4098, 4134, 4142, 4150, 4254, 4263, 4271, 4278, 4283, 4331, 4340, 4430, 4438, 4640, 4704, 4750, 4758, 4904, 5012, 5020

\IfHyperBoolean 397

\IfPackageLoadedTF 7, 15, 393, 406

\ignorespaces .. 1068, 1081, 1093, 1105, 4243, 4755, 4832, 4865, 4888, 4895, 4941, 4961, 5017, 5075, 5109, 5141, 5218, 5238

\inputlineno 290, 305, 318, 326, 334

int commands:

 \int_add:Nn 4550, 4599

 \int_case:nn ... 1242, 1367, 2167, 2193, 2232, 2256

 \int_case:nnTF 244

 \int_compare:nNnTF .. 571, 787, 803, 822, 829, 1337, 1356, 1510, 1528, 1640, 1659, 1671, 1699, 2280, 2286, 2807, 2811, 2815, 2823, 2869, 2873, 2877, 3074, 3095, 3136, 3141, 3146, 3171, 3289, 3737, 3748, 3770, 3783, 3799, 3814, 3829, 3870, 3935, 3939, 3967, 3992, 4008, 4202, 4314, 4318, 4520, 4530, 4546, 4569, 4579, 4595, 4768, 4772, 4810, 4820, 4972, 4984, 5033, 5045, 5253, 5265, 5462, 5594

 \int_compare_p:nNn ... 259, 269, 281, 282, 296, 297, 1646, 1647, 2173, 2199, 2580, 2590, 2602, 2603, 2618, 2659, 2846, 2847, 2858, 2859, 3011, 3780

 \int_decr:N 4549, 4598

\int_eval:n .. 376, 919, 2479, 2630, 2740, 3154, 3306, 3671, 3716, 4538, 4587, 5168

\int_from_alph:n 884, 898

\int_from_roman:n 886, 900

\int_gadd:Nn 4551, 4600

\int_gdecr:N 2176, 2181, 2185, 2189, 2202

\int_gincr:N 2009, 2014, 2642, 3209, 3325, 3359, 3542, 3859, 3959, 4358, 4838, 4914, 5081, 5150

\int_gset:Nn 453, 483, 2225

\int_gset_eq:NN .. 450, 480, 1908, 1915, 1921, 1927, 1935, 1942, 1948, 1954

\int_gzero:N . 347, 348, 349, 1685, 1712, 2292, 3067, 3875, 4013, 4995, 5277

\int_if_exist:NTF 1883, 1919, 1925, 1946, 1952, 2130

\int_incr:N 2822, 3291, 3736, 3930, 4201, 4767, 4837, 5032, 5080

\int_mod:nn 4986, 5267

\int_new:N . 24, 25, 26, 27, 28, 29, 57, 58, 83, 100, 119, 133, 145, 146, 157, 158, 159, 161, 172, 173, 181, 182, 183, 184, 185, 1885, 2133

\int_set:Nn 880, 884, 886, 2022, 2029, 2041, 2050, 2959, 4424, 4425, 4465, 4496, 4519, 4525, 4541, 4568, 4574, 4590, 4970, 5251, 5458, 5596

\int_set_eq:NN 2010, 2015, 4548, 4597

\int_sign:n 2227

\int_step_function:nnN 2596, 2609, 2623

\int_step_function:nnnN 5600

\int_step_inline:nn 5510

\int_step_inline:nnn 4426

\int_to_roman:n 220, 2576, 2613

\int_use:N 369, 374, 375, 1338, 1357, 1672, 2024, 2031, 2043, 2052, 3671, 3691, 3716, 3756, 3800, 3809, 3824, 3830, 4523, 4524, 4536, 4572, 4573, 4585, 5931, 5935, 5941, 5945

\int_zero:N 3294, 4976, 5257

\item . 95, 99, 124, 127, 130, 132, 382, 2497, 2503, 2528, 2534, 2656, 3173, 3176, 3377, 3546, 4258, 4259, 4752, 4754, 5014, 5016, 5148

\item* 5, 16, 75, 3544

item-join 2750

item-pos* 2750, 3440

item-star 2750

item-sym* 2750, 3440

\itemindent 102

\itemindent 101

itemindent 1012

\itemsep 4247

\itemwidth . 587, 2364, 3890, 3896, 4027, 4033, 4559, 4563, 4608, 4612

K

keyans 15, 4036

keyans* 15, 5002

keyanspic 16, 4249

Keys for \anskey provide by **enumext**:

 break-col 82, 84, 87–89

 item-join 82, 84, 87–89

 item-pos* 83, 84, 87–89

 item-star 83, 84, 87–89

 item-sym* 83, 84, 87–89

Keys for anskey* provide by **enumext**:

 break-col 82, 84, 87–89

 item-join 82, 84, 87–89

 item-pos* 83, 84, 87–89

 item-star 83, 84, 87–89

item-sym* 83, 84, 87–89

Keys for environments provide by [enumext](#):

above* 31, 50, 64, 65, 106, 124

above 31, 50, 64, 65, 106, 124, 129

after 52, 53, 107, 124, 129

align 31, 44, 94, 95, 97, 100, 126, 140

base-fix 50, 66, 78, 104

before* 52, 53, 106, 124, 129

before 52, 53

below* 31, 64, 65, 107, 124

below 31, 64, 65, 107, 124, 129

check-ans . 33, 34, 36, 71, 72, 74, 75, 79, 90, 93, 107, 108, 124, 128, 142

columns-sep 54, 105, 128

columns 31, 54, 64, 105

first 52, 53, 127

font 43, 97, 100, 116, 126

item-pos* 96, 98

item-sym* 32, 96, 98

itemindent 31, 50, 51, 95, 96, 99, 100, 127

itemsep 49, 103, 128

label-pos 112, 113, 115, 116

label-sep 112

labelsep 43, 102, 126

labelwidth 42, 43, 45–48, 102, 126

label 30, 42, 43, 45, 48, 113, 118

layout-sep 112

layout-sty 112, 117

layout-top 112

lisparindent 103

list-indent 31, 50, 51, 113

list-offset 50, 51, 107, 109

listparindent 50, 127

mark-ans* 76, 94

mark-ans 76, 79, 84

mark-pos* 76, 94

mark-pos 32, 76, 140

mark-ref 76, 79, 81, 83

mark-sep* 76, 94

mark-sep 32, 76, 79, 131

mini-env 31, 39–41, 54, 63, 64, 79, 106, 118, 120, 122, 124, 129

mini-right* 31, 34, 54, 79, 121, 122, 124

mini-right 31, 34, 54, 62, 79, 121, 122, 124

mini-sep 31, 54, 79, 106

mode-box 43, 95, 97, 100, 101

no-store 33, 71–73, 78, 85, 95, 96

noitemsep 49

nosep 49

parindent 103

parsep 49, 103, 113, 127

partopsep 49

ref 30, 35, 45–47, 141

resume* 30, 66, 67, 70–72, 78, 107, 124, 136

resume 30, 37, 66–72, 78, 79, 107, 124, 136

rightmargin 50, 118

save-ans 32, 37, 66–71, 73, 74, 77–79, 85–87, 90–92, 99, 108, 115, 126, 129–131, 133, 134, 136, 142

save-key 32, 66, 78, 79, 104, 123

save-pos 79

save-ref 33, 39, 76, 79, 81–83, 91, 93, 100, 131

save-sep 76, 79, 91, 131

series 30, 66–70, 79, 104, 107, 123, 124, 136

show-ans 32, 76, 79, 80, 82, 84, 93, 94, 116, 131

show-length 35, 52, 141

show-pos 32, 76, 80, 82, 84, 93, 116, 131

start* 31, 48, 66

start 31, 35, 48, 66

store-key 77

topsep 49, 50, 113

widest 30, 35, 48

wrap-ans* 33, 76, 79, 100, 116

wrap-ans 42, 76, 79, 80, 83

wrap-label* 31, 43, 95, 97, 99, 100, 125, 126, 130

wrap-label 31, 43, 95–97, 99, 100, 113, 116, 125, 126, 130

wrap-opt 76, 79, 93, 99, 116

wrap-sep 83

keys commands:

\keys_define:nn 639, 649, 671, 706, 722, 768, 837, 908, 934, 976, 1014, 1037, 1111, 1120, 1199, 1216, 1723, 1834, 2077, 2138, 2297, 2332, 2420, 2425, 2750, 2901, 2937, 3442, 3458, 3481, 3501, 4156, 5308, 5420, 5536, 5544

\keys_if_exist_p:nn 5532, 5533

\l_keys_key_str 84, 87, 2768, 2923, 3469, 3487, 3509, 5579, 5687

\keys_precompile:nnN .. 135, 204, 204, 5310, 5316, 5322, 5328, 5334, 5340, 5562

\keys_set:nn . 663, 993, 1005, 1222, 1728, 1733, 1971, 1976, 2063, 2071, 2368, 2369, 2373, 2374, 2378, 2379, 2383, 2384, 2388, 2389, 2393, 2394, 2788, 3750, 3755, 3946, 4174, 4176, 4178, 4180, 4182, 4184, 4186, 4188, 4190, 4192, 4212, 4785, 5054, 5424, 5429, 5430, 5431, 5432, 5435, 5440, 5441, 5442, 5443, 5444, 5445, 5446, 5478, 5588

\keys_set_known:nn 3055

keyval commands:

\keyval_parse:NNn 1848, 2450, 5396

L

label 720, 766, 837

label-pos 4156

label-sep 4156

Labels provide by [enumext](#):

\Alph* 42, 43

\Roman* 42, 43

\alph* 42, 43

\arabic* 35, 42, 43

\roman* 42, 43

labelsep 647

\labelwidth 43

labelwidth 647

\lastnodetype 244

layout-sep 4156

layout-sty 4156

layout-top 4156

\leftmargin 102

\leftmargin 101, 4242

legacy commands:

\legacy_if:nTF 4899, 4902, 5201, 5204

\legacy_if_gset_false:n 577, 4655

\legacy_if_set_false:n 4901, 5203

\legacy_if_set_true:n 4864, 4887, 4894, 4908, 5108, 5140

\linewidth 106

\linewidth 3854, 3890, 3956, 4027, 4423, 4468, 4499, 4621, 4686

\list 380

list-indent 1012

list-offset 1012

<code>\listparindent</code>	4245
<code>listparindent</code>	<u>1012</u>
M	
<code>\makebox</code>	118
<code>\makebox</code>	2557, 3424, 3614, 4348, 4361, 4932, 5213
<code>\makelabel</code>	95, 97, 100, 118
<code>\makelabel</code>	95, 99, 3404, 3420, 3598, 3610
<code>mark-ans</code>	<u>2330</u> , <u>4156</u>
<code>mark-ans*</code>	<u>2295</u> , <u>2330</u>
<code>mark-pos</code>	<u>2330</u> , <u>4156</u>
<code>mark-pos*</code>	<u>2295</u> , <u>2330</u>
<code>mark-ref</code>	<u>2330</u>
<code>mark-sep</code>	<u>2330</u> , <u>4156</u>
<code>mark-sep*</code>	<u>2295</u> , <u>2330</u>
<code>mini-env</code>	<u>1197</u>
<code>mini-sep</code>	<u>1197</u>
<code>\minipage</code>	386
<code>\miniright</code>	11, 62, <u>1638</u> , 1689, 1716, 3873, 4011
mode commands:	
<code>\mode_if_math:TF</code>	2831, 2885
<code>\mode_if_vertical:TF</code>	1276, 1304, 1324, 1348, 1499, 1520
<code>\mode_leave_vertical:</code>	991, 1002, 1065, 1079, 2553, 3369, 4926
<code>mode-box</code>	<u>637</u>
msg commands:	
<code>\msg_error:nn</code> ..	1691, 1718, 2792, 2825, 2829, 2883, 2991, 3937, 3941, 4204, 4261, 4316, 4770, 5035, 5047, 5447, 5506
<code>\msg_error:nnn</code> ..	745, 791, 807, 857, 1642, 1649, 1656, 1687, 1714, 1983, 1987, 2102, 2774, 2833, 2851, 2863, 2871, 2875, 2879, 2887, 2929, 3475, 3493, 3515, 4774, 5040, 5296, 5305, 5389, 5494, 5525, 5534, 5571, 5592
<code>\msg_error:nnnn</code> ..	2777, 2805, 2809, 2813, 2817, 2932, 3478, 3496, 3518, 3928, 4312, 4320, 5030, 5368, 5574
<code>\msg_error:nnnnn</code>	693, 713, 2307, 2356, 4166
<code>\msg_fatal:nn</code>	3738
<code>\msg_fatal:nnn</code>	591
<code>\msg_info:nnn</code>	9, 12, 17, 20, 395, 408
<code>\msg_line_context:</code> ..	5652, 5657, 5662, 5691, 5696, 5701, 5716, 5731, 5735, 5739, 5743, 5747, 5751, 5758, 5765, 5771, 5785, 5789, 5794, 5798, 5802, 5806, 5811, 5815, 5819, 5823, 5828, 5863, 5867, 5872, 5877, 5881, 5886, 5962, 5966, 5971, 5976, 5981, 5985, 5989, 5993, 5997, 6001, 6005, 6009, 6013
<code>\msg_log:nnn</code>	2122, 2127, 2132
<code>\msg_log:nnnnn</code>	373, 2265, 2270, 2275
<code>\msg_log:nnnnnn</code>	365
<code>\msg_new:nnn</code> ..	5619, 5623, 5627, 5631, 5636, 5649, 5654, 5659, 5664, 5673, 5681, 5685, 5689, 5694, 5699, 5714, 5729, 5733, 5737, 5741, 5745, 5749, 5753, 5762, 5768, 5774, 5778, 5782, 5787, 5792, 5796, 5800, 5804, 5809, 5813, 5817, 5821, 5826, 5861, 5865, 5870, 5875, 5879, 5884, 5960, 5964, 5969, 5974, 5979, 5983, 5987, 5991, 5995, 5999, 6003, 6007, 6011
<code>\msg_new:nnnn</code> ..	5640, 5831, 5840, 5849, 5855, 5888, 5898, 5908, 5918, 5928, 5938, 5948, 5954
<code>\msg_term:nnnn</code> ..	2086, 2091, 3680, 3690, 3722, 3727
<code>\msg_term:nnnnn</code>	2246
<code>\msg_warning:nn</code>	3872, 4010
<code>\msg_warning:nnnn</code> ..	2283, 2289, 3628, 3633, 4522, 4535, 4571, 4584
<code>\msg_warning:nnnnn</code>	2241, 2251
<code>\multicolsep</code>	105

<code>\multicolsep</code>	1341, 1513, 3820, 3983
N	
<code>\NeedsTeXFormat</code>	3
<code>\NewCommandCopy</code>	382
<code>\newcounter</code>	594
<code>\NewDocumentCommand</code> ..	1638, 2780, 4308, 5280, 5347, 5454, 5503, 5581
<code>\NewDocumentEnvironment</code> ..	3902, 4036, 4249, 4739, 5002
<code>\newenvsc</code>	2894
<code>\newlabel</code>	39
<code>\newlabel</code>	428
<code>no-store</code>	<u>2136</u>
<code>\noindent</code>	3861, 4630, 4695, 4975, 5256
<code>\nointerlineskip</code> ..	1350, 1353, 1522, 1525, 1679, 1706, 4630, 4695
<code>noitemsep</code>	<u>932</u>
<code>\nopagebreak</code> ..	1287, 1315, 1350, 1353, 1522, 1525, 1629, 1635
<code>\normalfont</code>	2739, 3305, 5167
<code>nosep</code>	<u>932</u>

P	
Packages:	
<code>caption</code>	121
<code>enumext</code>	29, 42, 45, 71, 75, 97, 102, 112, 140
<code>enumitem</code>	42
<code>expl3</code>	118
<code>footnotehyper</code>	38, 40, 41
<code>hyperref</code>	33, 34, 38, 39, 83, 93, 126, 140
<code>latex-lab-block</code>	38
<code>ltxcmd</code>	38
<code>ltsockets</code>	110
<code>lua-visual-debug</code>	57
<code>multicol</code>	29, 140
<code>scontents</code>	29, 86, 87
<code>shortlst</code>	118, 122, 127
<code>tagpdf</code>	110
<code>\par</code> ..	1287, 1315, 1353, 1525, 1629, 1635, 1674, 1679, 1701, 1706, 2705, 3837, 3998, 4016, 4294, 4297, 4443, 4657, 4672, 4718, 4732, 4975, 5256
para commands:	
<code>\para_end:</code>	4992, 5274
<code>\parbox</code>	2364
<code>\parindent</code>	4957, 5234
<code>\parsep</code>	55, 113
<code>\parsep</code>	992, 3713, 4226, 4235
<code>parsep</code>	<u>932</u>
<code>\parskip</code>	4958, 5235
<code>\partopsep</code>	3714, 4014, 4246
<code>partopsep</code>	<u>932</u>
peek commands:	
<code>\peek_meaning:NTF</code> ..	4843, 4857, 4872, 4883, 5086, 5101, 5117
<code>\peek_meaning_remove:NTF</code>	4850, 5093
<code>\peek_remove_spaces:n</code>	3551
<code>\phantomsection</code>	39
<code>\phantomsection</code>	417
prg commands:	
<code>\prg_do_nothing:</code>	421
<code>\prg_new_protected_conditional:Npnn</code> ..	222
<code>\prg_replicate:nn</code>	239
<code>\prg_return_false:</code>	226
<code>\prg_return_true:</code>	225
<code>\printkeyans</code>	19, 134, <u>5347</u>
prop commands:	
<code>\prop_const_from_keyval:Nn</code>	5495

\prop_count:N 367, 2479, 2630, 2742, 3154, 3308, 5170, 5597

\prop_get:NnNTF 5521

\prop_gput_if_not_in:Nnn 2477

\prop_if_exist:NTF 2120, 5300, 5590

\prop_item:Nn 5302, 5614

\prop_new:N 2123

\ProvidesExplPackage 4

R

\raggedcolumns 3823, 3986

\raisebox 4385

\ref 81, 91

ref 720, 766, 837

\refstepcounter 4911, 5206

regex commands:

\regex_match:nnTF .. 224, 883, 885, 897, 899, 2987

\regex_replace_once:nnN 232

\renewcommand 755, 799, 815, 865

\RenewDocumentCommand . 445, 475, 1689, 1716, 3377, 3404, 3420, 3546, 3598, 3610, 4259

\RequirePackage 13, 21

resume 1832

resume* 1832

rightmargin 1012

\Roman 43, 48

\Roman 613

\roman 43, 48

\roman 614, 738, 5332

S

\s 2988

save-ans 2075

save-key 2418

save-ref 2330

save-sep 2295, 2330, 4156

scan commands:

\scan_stop: 4258, 4752, 5014, 5291, 5294

scontents internal commands:

\l_scontents_fname_out_tl 2947

__scontents_parse_environment_keys:n . 2953

__scontents_rescan_tokens:n 2960

\l_scontents_storing_bool 2945

\l_scontents_writing_bool 2946

seq commands:

\seq_clear:N 5456, 5599

\seq_const_from_clist:Nn 5449

\seq_count:N 368, 4449, 5460

\seq_gclear:N 470, 471, 500, 501

\seq_gput_right:Nn 456, 457, 486, 487, 2486

\seq_if_empty:NTF 463, 493, 5362, 5474

\seq_if_exist:NTF 2125, 5360

\seq_if_in:NnTF 5366

\seq_item:Nn 2985, 4436

\seq_map_function:NN 5465

\seq_map_inline:Nn 5375, 5383, 5475, 5476

\seq_map_pairwise_function:NNN 465, 495

\seq_new:N 120, 121, 123, 143, 174, 175, 176, 177, 2128

\seq_pop_left:NN 5464

\seq_put_right:Nn 4322, 5472, 5488, 5609

\seq_set_from_clist:Nn 5457

\seq_set_map_e:NNn 5466

\seq_use:Nn 204, 205, 5605

series 1832

\setcounter 894, 898, 900, 3671, 3716, 4291

\setenumext 6, 136, 5454

\setenumextmeta 6, 138, 5495

show-ans 2295, 2330, 4156

show-length 1109

show-pos 2295, 2330, 4156

skip commands:

\skip_add:Nn 1247, 1256, 1265, 1278, 1282, 1306, 1310, 1326, 1384, 1386, 1400, 1403, 1424, 1426, 1440, 1443, 1463, 1465, 1479, 1482, 1501, 1550, 1551, 1562, 1564, 4235, 4244

\skip_gset:Nn 1577, 1581, 1585

\skip_gzero_new:N 1572, 1573

\skip_horizontal:N .. 1080, 1092, 1104, 4929, 4941, 4979, 5218, 5260

\skip_horizontal:n .. 1066, 2554, 2562, 3370, 3372, 4366, 4828, 4927, 4961, 5071, 5238

\skip_if_eq:nnTF 1245, 1254, 1263, 1370, 1410, 1450, 1538, 1574, 1596, 1740, 1754, 1768, 1779, 1790, 1801, 1812, 1823

\skip_new:N ... 77, 78, 79, 84, 85, 86, 87, 88, 89, 196

\skip_set:Nn 1230, 1234, 1292, 1296, 1320, 1373, 1374, 1392, 1413, 1414, 1432, 1452, 1453, 1471, 1495, 1541, 1542, 1556, 1576, 1580, 1598, 1602, 1606, 1612, 1616, 1620, 4219

\skip_set_eq:NN 1331, 1332, 1334, 1341, 1506, 1507, 1508, 1513, 3669, 3712, 3713, 4958, 5235

\skip_sub:Nn 1380, 1382, 1396, 1398, 1420, 1422, 1436, 1438, 1459, 1461, 1475, 1477, 1548, 1549, 1560, 1561

\skip_use:N 1232, 1236, 1280, 1284, 1288, 1308, 1312, 1322, 1328, 1741, 1745, 1748, 1755, 1759, 1762, 3837

\skip_vertical:N . 578, 581, 1004, 4656, 4670, 4994, 5276

\skip_vertical:n 1003, 4993, 5275

\skip_zero:N 1340, 1354, 1492, 1493, 1494, 1512, 1526, 3714, 3820, 3983, 4246, 4247

\skip_zero_new:N 1571, 1593, 1594, 1595

\c_zero_skip . 578, 581, 1004, 1245, 1254, 1263, 1411, 1450, 1574, 1596, 1741, 1755, 1768, 1779, 1790, 1801, 1812, 1823, 4656, 4670, 4994, 5276

\small 5315, 5321, 5327, 5333, 5339, 5345

\smash 3422, 3612

socket commands:

\socket_assign_plug:nn .. 4084, 4092, 4100, 4136, 4144, 4152

\socket_new:nn 4054, 4104

\socket_new_plug:nnn 4055, 4063, 4071, 4105, 4113, 4122

\socket_use:n 4137, 4145, 4153

\socket_use:nn 4085, 4093, 4101

start 906

start* 906

start-list-tags 4054, 4104

\stepcounter 449, 479, 4229, 4378

stop-list-tags 4054, 4104

stop-start-tags 4054, 4104

str commands:

\c_backslash_str 2833, 5652, 5657, 5662, 5667, 5669, 5671, 5676, 5678, 5776, 5780, 5784, 5794, 5798, 5806, 5807, 5811, 5823, 5824, 5828, 5829, 5850, 5852, 5856, 5858, 5886, 5949, 5951, 5955, 5957, 5966, 5967, 5971, 5976, 5977, 5981, 5985, 5989

\c_colon_str 2629, 3153, 5291

\c_left_brace_str 5757, 5764, 5770

\c_right_brace_str 5757, 5764, 5770

\str_case:nn 252, 311, 3249

<code>\str_case:nnTF</code>	1855, 1863, 2457, 2465, 5403, 5412
<code>\str_clear:N</code>	3747, 4784
<code>\str_count:n</code>	239
<code>\str_if_empty:NTF</code>	1872, 1913, 1940
<code>\str_if_eq:nnTF</code>	3672, 3718, 5505
<code>\str_if_in:nnTF</code>	5287
<code>\str_new:N</code>	80, 128, 129, 130, 148, 191
<code>\str_set:Nn</code>	678, 684, 690, 709, 710, 711, 2303, 2304, 2305, 2352, 2353, 2354, 4161, 4164
<code>\str_set_eq:NN</code>	3272, 5158, 5175
<code>\str_use:N</code>	3426
<code>\strut</code>	3422, 3612
<code>\strutbox</code>	1359, 1362, 1373, 1374, 1385, 1387, 1402, 1405, 1413, 1414, 1425, 1427, 1442, 1445, 1452, 1453, 1464, 1466, 1481, 1484, 1530, 1533, 1541, 1542, 1550, 1551, 1563, 1565, 1576, 1577, 1580, 1587, 1600, 1608, 1614, 1622, 4238, 4244, 4294, 4302, 4391
T	
tag commands:	
<code>\tag_mc_begin:n</code>	4061, 4111, 4120
<code>\tag_mc_begin_pop:n</code>	4077, 4129, 4286, 4288
<code>\tag_mc_end:</code>	4065, 4115, 4124
<code>\tag_mc_end_push:</code>	4058, 4108, 4274
<code>\tag_resume:n</code>	4057, 4107, 4265, 4273, 4342, 4440, 4640, 4704
<code>\tag_struct_begin:n</code>	4059, 4060, 4067, 4068, 4069, 4109, 4110, 4117, 4118, 4119, 4275
<code>\tag_struct_end:n</code>	4066, 4073, 4074, 4075, 4076, 4116, 4125, 4126, 4127, 4128, 4285, 4287, 4758, 5020
<code>\tag_suspend:n</code>	4078, 4130, 4256, 4267, 4280, 4333, 4432, 4750, 5012
<code>\tag_tool:n</code>	4266
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@auxout</code>	426
<code>\@currentenv</code>	252, 311
<code>\protected@write</code>	426
tex commands:	
<code>\tex_newlinechar:D</code>	2959
text commands:	
<code>\text_expand:n</code>	5283
<code>\textasteriskcentered</code>	2300, 2347
<code>\textborn</code>	3446
<code>\textreferencemark</code>	2335
<code>\thepage</code>	432
tl commands:	
<code>\c_space_tl</code>	3228, 3241, 5701, 5716, 5739, 5743, 5930, 5931, 5940, 5941, 6001, 6005
<code>\tl_clear:N</code>	676, 683, 2293, 2404, 2414, 2435, 2443, 2649, 2979, 2980, 3094, 3170, 5123
<code>\tl_clear_new:N</code>	623
<code>\tl_const:Nn</code>	46, 607
<code>\tl_gclear:N</code>	359, 360, 361, 1893, 1898, 3069, 3415, 3435, 4676, 4736, 4930
<code>\tl_gclear_new:N</code>	1880
<code>\tl_gput_right:Nn</code>	608
<code>\tl_greplace_all:Nnn</code>	629
<code>\tl_gset:Nn</code>	287, 288, 302, 303, 1881, 1894, 1899, 2118, 2983, 3346, 4878
<code>\tl_gset_eq:NN</code>	625, 3342, 4923
<code>\tl_if_blank:NTF</code>	2772, 2790, 2927, 3473, 3491, 3513, 4921, 5569
<code>\tl_if_empty:NTF</code>	743, 761, 789, 805, 824, 831, 855, 871, 1906, 1911, 1933, 1938, 1996, 2060, 2068, 2097, 2157, 2493, 2524, 2669, 3024, 3046, 3076, 3104, 3180, 3221, 3234, 3367, 4447, 5126, 5486

<code>\tl_if_empty:NTF</code>	1961
<code>\tl_if_exist:NTF</code>	1966
<code>\tl_if_novalue:NTF</code>	447, 477, 2786, 3102, 3178, 3214, 3321, 3340, 3348, 3523, 3745, 4210, 4782, 5052, 5124
<code>\tl_map_inline:Nn</code>	230, 626
<code>\tl_new:N</code>	38, 39, 40, 43, 48, 49, 52, 53, 59, 61, 62, 64, 65, 101, 102, 103, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 122, 124, 125, 126, 134, 137, 138, 155, 164, 165, 166, 169, 190
<code>\tl_put_left::Ne</code>	3013
<code>\tl_put_left:Nn</code>	2501, 2532, 2654, 3007, 3020, 3026, 3036, 4660, 4721, 5145, 5148
<code>\tl_put_right:Nn</code>	624, 753, 797, 813, 863, 2505, 2536, 2583, 2593, 2606, 2621, 2627, 2632, 2656, 2661, 2668, 2671, 2681, 2686, 2689, 2695, 3097, 3100, 3106, 3111, 3138, 3143, 3148, 3151, 3160, 3173, 3176, 3182, 3187, 3197, 5128, 5132
<code>\tl_remove_all:Nn</code>	5485
<code>\tl_remove_once:Nn</code>	2571, 3123
<code>\tl_replace_all:Nnn</code>	628, 5520
<code>\tl_reverse:N</code>	2570, 2572, 3122, 3124
<code>\tl_set:Nn</code>	54, 256, 266, 315, 316, 323, 324, 331, 332, 593, 677, 682, 688, 689, 742, 786, 854, 1063, 1077, 1090, 1102, 1995, 2096, 2405, 2415, 2436, 2444, 2736, 2947, 3216, 3302, 3461, 4195, 5134, 5164, 5483, 5519, 5589
<code>\tl_set_eq:NN</code>	634, 748, 751, 794, 796, 810, 812, 860, 862, 2569, 3121, 3134, 3270, 5157
<code>\tl_to_str:n</code>	1966, 1972, 1977, 5283
<code>\tl_trim_spaces:n</code>	624, 5472, 5483, 5489, 5505
<code>\tl_use:N</code>	630, 633, 763, 826, 833, 873, 1135, 1139, 1143, 1147, 1151, 1155, 1159, 1163, 1167, 1171, 1175, 1179, 1183, 1187, 1191, 1195, 2559, 2576, 2584, 2595, 2608, 2613, 2624, 3329, 3335, 3363, 3406, 3408, 3414, 3429, 3526, 3530, 3537, 3600, 3603, 3605, 3618, 3909, 4042, 4363, 4371, 4667, 4728, 4934, 4962, 4963, 5215, 5239, 5244, 5350, 5351, 5352, 5353, 5354, 5371, 5468, 5587
token commands:	
<code>\token_to_str:N</code>	428
<code>\topsep</code>	4014, 4244
<code>topsep</code>	<u>932</u>
<code>\topskip</code>	1340, 1512
U	
<code>\u</code>	233, 2988
<code>\unkern</code>	247
<code>unknown</code>	<u>2750</u> , <u>3456</u> , <u>3481</u> , <u>3499</u>
<code>\unskip</code>	246
use commands:	
<code>\use:N</code>	240, 3411, 3432, 3911
<code>\use:n</code>	1846, 2448, 5289, 5394
<code>\use_none:nn</code>	420, 5526
<code>\usecounter</code>	3670, 3715
V	
<code>\value</code>	1909, 1915, 1922, 1928, 1936, 1942, 1949, 1955
vbox commands:	
<code>\vbox_set:Nn</code>	4335
<code>\vbox_set_top:Nn</code>	4665, 4726
<code>\vspace</code>	992, 1745, 1748, 1759, 1762, 1772, 1774, 1783, 1785, 1794, 1796, 1805, 1807, 1816, 1818, 1827, 1829
W	
<code>widest</code>	<u>906</u>
<code>wrap-ans</code>	<u>2330</u>
<code>wrap-ans*</code>	<u>2295</u> , <u>2330</u> , <u>4156</u>

wrap-label	<u>647</u>		Z	
wrap-label*	<u>647</u>	\z	2988
wrap-opt	<u>2295</u> , <u>2330</u> , <u>4156</u>			