

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-10-04^{*}

©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 L^AT_EX tagging PDF for creating “simple exercise sheets” along with “multiple choice questions”, storing the “answers” to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

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 <code>label</code> and <code>ref</code>	12
1.3	User interface	3	6.1.2	Keys for <code>wrap</code> and <code>display</code>	12
1.3.1	Internal counters	3	6.1.3	Keys for <code>debug</code> and <code>checking</code>	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	13
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	13
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	14
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	6.6	Printing stored content	17
2.3.1	Keys for <code>\item*</code>	6	6.6.1	The command <code>\getkeyans</code>	17
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.2	The command <code>\foreachkeyans</code>	17
3	The command <code>\setenumext</code>	6	6.6.3	The command <code>\printkeyans</code>	17
4	The command <code>\setenumextmeta</code>	6	7	Full examples	18
5	The keyval system	7	8	Tagged PDF examples	21
5.1	Keys for <code>label</code> and <code>ref</code>	7	9	The way of non-enumerated lists	21
5.2	Keys for spaces	8	10	References	23
5.2.1	Vertical spaces	8	11	Change history	24
5.2.2	Horizontal spaces	9	12	Index of Documentation	25
5.3	Keys for <code>add code</code>	9	13	Implementation	27
5.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	10	14	Index of Implementation	140
5.5	Keys for <code>multicols</code>	10			
5.6	Keys for <code>minipage</code>	10			
5.6.1	The command <code>\miniright</code>	10			
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 L^AT_EX 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 2024-10-04.

[†]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 scontents[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.

1 Introduction

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

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

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

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

*

$(x - 1)^2$

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

*

$3(x + y + z)$

3. True False

(a) $\alpha > \delta$

*

False

(b) L^AT_EXze is cool?

*

Very True!

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

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn't exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete

Or we are interested in referring to a specific question and its “answer”, for example:

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

Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- * (b) Yes, dnf

* (c) i. doesn't exist for now :(

* ii. very good

* iii. obsolete

*

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

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

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- Where what we are interested in the *label* and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L
- 2 / 155

1. B) $x = 5$

2. D)

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

* 5. D) "other note"

*

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.

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

1.1 Description and usage

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

- This package is NOT 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 `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “*extra space*” 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.

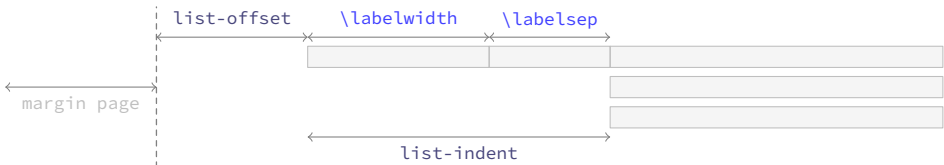


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

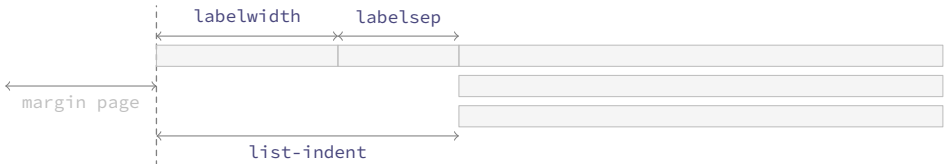


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

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` to print all *stored content*, `\miniright` for `minipage` and `\setenumext` to config all `[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 \TeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§6.1.1) when the key `save-ans` (§6.1) is active.

1.3.6 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package for the `enumext*` and `keyans*` environments, but will not produce the expected links, and if the `mini-env` key is used in `enumext` or `keyans` environments the output will look like the classic way they are displayed in the environment `minipage`.

The best way to solve this 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) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

```
\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}[\langle keyval list \rangle]</code>	<code>\begin{enumext*}[\langle keyval list \rangle]</code>
<code>enumext*</code>	<code>\item \langle item content \rangle</code>	<code>\item \langle item content \rangle</code>
	<code>\item [\langle custom \rangle] \langle item content \rangle</code>	<code>\item [\langle custom \rangle] \langle item content \rangle</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</code>	<code>\item* [\langle symbol \rangle] [\langle offset \rangle] \langle item content \rangle</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[\langle custom \rangle]` 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 `[\langle key = val \rangle]` 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 `enumerate*` environment provided by the `enumitem` package or `task` environment provided by the `task` package, `\item` and `\item[\langle custom \rangle]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` 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.
- 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*`

<code>\item*</code>	<code>\item*</code>
	<code>\item* [\langle symbol \rangle]</code>
	<code>\item* [\langle symbol \rangle] [\langle offset \rangle]</code>

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the numbered `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the `\langle offset \rangle` set by the the *second optional argument*. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\$star$ ‘★’` and the value set by `labelsep` key.

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

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

2.3.1 Keys for \item*

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

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

2.4 The command \item in enumext*

The `\item` command for the `enumext*` environment provides an “first optional argument” `\item($\langle columns \rangle$)` which “joins items” between columns. Let’s consider the following examples adapted directly from the `task` package:

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

1. The first
- ★ 2. The second
3. The third
4. The fourth
- ★ 5. The fifth item is way too long for this and needs three columns
6. The sixth
7. The seventh
- X 8. The eighth item is way too long for this and needs two columns (196.17749pt)
9. The ninth
- Z 10. The tenth (89.28171pt)

3 The command \setenumext

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

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

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

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

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

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

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

`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*. 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 `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 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 = { $\langle code \{ \#1 \} \text{ more code } \rangle$ }` default: *empty*

Wraps the *current* $\langle label \rangle$ defined by `label` key referenced by $\{ \#1 \}$. The $\{ \langle code \rangle \}$ must be passed between braces. This key 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 hash* ‘ $\{ \# \#1 \}$ ’. For example `wrap-label={\fbox{\#1}}` or you can create a command:

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

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

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

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

- By default all the $\langle keys \rangle$ described above are executed inside `\makebox` in the `enumext*` and `keyans*` environments.
- For compatibility with tagged PDF all $\langle keys \rangle$ described above are executed inside `\makebox` in the `enumext` and `keyans` environments, this means that the document output may *not look* the same when `\DocumentMetadata` is active. If you use the `wrap-label` or `wrap-label*` keys you can add conditional code using `\IfDocumentMetadataTF`.

`ref = { $\langle code \{ \backslash\alpha^* | \backslash\Alpha^* | \backslash\arabic^* | \backslash\roman^* | \backslash\Roman^* \} \text{ more code } \rangle$ }` 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{\langle \backslash\alpha^* \rangle}` 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. \TeX will enter *⟨vertical mode⟩* and apply this value to the “top” and “bottom” the environment or nested level.

`itemsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

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

`noitemsep` *⟨value forbidden⟩*

default: *not used*

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

`nosep` *⟨value forbidden⟩*

default: *not used*

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

`base-fix` *⟨value forbidden⟩*

default: *not used*

This is a “meta-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`.

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

`above = {⟨rubber length | rigid length⟩}`

default: *not used*

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

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

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

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

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

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

5.2.2 Horizontal spaces

`itemindent` = { $\langle rigid\ length \rangle$ } default: *0pt*

Extra *horizontal indentation*, beyond `labelsep`, of the “*first line*” off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

`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.

`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 environment `enumext` 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*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

🔗 For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

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

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

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

Executes $\{\langle code \rangle\}$ when “*starting*” the environment. The $\{\langle code \rangle\}$ 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}{\langle arg one \rangle}{\langle arg two \rangle}\{\langle code \rangle\}\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 within the `minipage` environment in which “*item content*” is placed.

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

Execute $\{\langle code \rangle\}$ “*after*” finishing the environment. The $\{\langle code \rangle\}$ 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 multicols

`columns = {⟨integer⟩}` default: 1
Set the *number of columns* to be used by the `multicols` environment within the environment. The value must be a positive integer less than or equal to 10.

`columns-sep = {⟨rigid length⟩}` default: by level
Set the *space between columns* used by the `multicols` environment within the environment. 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.

- The `\footnote{⟨text⟩}` command in the nested levels of `multicols` will not work as expected, prefer the use of `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the after key.

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 L^AT_EX justification is maintained in the `minipage` on the “right side”.

- The `\footnote{⟨text⟩}` command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key (see §1.3.6 for full support).

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

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

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

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

6.1.1 Keys for label and ref

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

`mark-ref` = {`<symbol>`} default: `\textasteriskcentered`
 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 display

`wrap-ans` = {`<code> {#1} more code`} default: `\fbox+\parbox{#1}`
 Wraps the *argument* 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 the *argument* or *body* 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}’.

`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}’.

`show-ans` = {`<true | false>`} default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the `<label>` for `\item*` and `\anspic*` at the place where it is executed. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

`mark-ans` = {`<symbol>`} default: `\textasteriskcentered`
 Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

`mark-pos` = {`<left | right>`} default: `left`
 Sets the *aligned* of the *symbol* defined by `mark-ans` key. 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 `labelsep` key.

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.

`no-store` `<value forbidden>` default: `not used`
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* {`<store name>`} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “*without*” using the `\anskey`, “*without*” use `anskey*`, “*without*” interfering with the `check-ans` key and “*without*” storing an unwanted structure in the *sequence* {`<store name>`}.

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 $\langle content \rangle$ 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: $\$star\$$

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}
```

- | | |
|--|---|
| <ul style="list-style-type: none"> * 1. Text containing our instructions or questions. * <div style="border: 1px solid black; padding: 2px; display: inline-block;">first answer</div> 2. Text containing our instructions or questions. (a) Question. * <div style="border: 1px solid black; padding: 2px; display: inline-block;">second answer</div> | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. * <div style="border: 1px solid black; padding: 2px; display: inline-block;">third answer</div> 4. Text containing our instructions or questions. * <div style="border: 1px solid black; padding: 2px; display: inline-block;">fourth answer</div> |
|--|---|

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 body and it is assumed that each numbered `\item` or `\item*` within the environment in which it is active it has a “*single execution*” unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the `scontents` package, for the correct operation `\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 keys 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` = { $\langle true | false \rangle$ } 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]
    \langle first answer \rangle
  \end{anskey*}
  \item Text containing our instructions or questions.
  \begin{enumext}
    \item Question.
    \begin{anskey*}
      \langle second answer \rangle
    \end{anskey*}
  \end{enumext}
  \item Text containing our instructions or questions.
  \begin{anskey*}
    \langle third answer \rangle
  \end{anskey*}
  \item Text containing our instructions or questions.
  \begin{anskey*}
    \langle fourth answer \rangle
  \end{anskey*}
\end{enumext}
```

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

6.4 The environments `keyans` and `keyans*`

<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.

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

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

<code>\item*</code>	<code>\item*</code>
	<code>\item*[$\langle content \rangle$]</code>

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

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


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

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*[\textit{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}*[\textit{n° upper, n° lower}]\anspic{\textit{drawing}}\anspic*[\textit{content}]{\textit{drawing or tabular}}
```

The `keyanspic` environment is an “*enumerated list*” environment activated by the `save-ans` key that has the same settings as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing drawings or tables with `\label` centered *above* or *below* in a *single line* or *upper and lower* layout. A representation of the output can be seen in the figure 6.

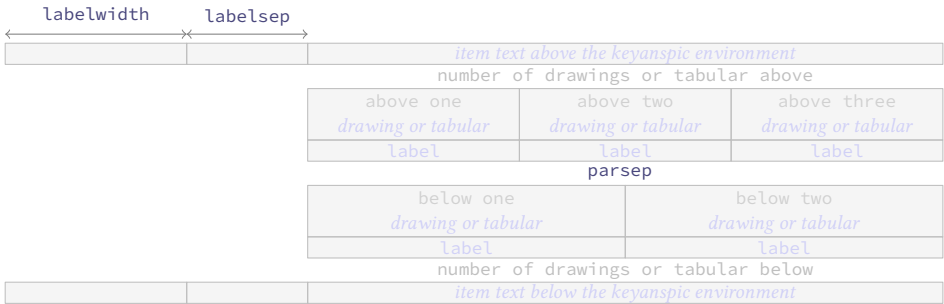


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

When the `keyanspic` environment is used *without arguments* the `\label` are centered *below* the drawings or tabular in a *single line* layout. The *starred argument* ‘*’ places `\label` centered *above* the drawings or tabular. The *optional argument* determines the number drawings or tabular placed at *upper and lower* in the environment. If the *optional argument* or the `\textit{n° lower}` is omitted the drawings or tabular will be put on a *single line*. The vertical separation between “*upper*” and “*lower*” part is controlled by the values set by `parsep` key passed to `keyans` environment.

6.5.1 The command \anspic

\anspic

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

The `\anspic` command take three arguments, the *starred argument* ‘*’ store the current `\label` next to the *optional argument* `\textit{content}` in *sequence* and *prop list* `{\textit{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,nosep]
  \item Question with images.

  \begin{keyanspic}[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.

  \begin{keyanspic}*[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.

  \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.




A)




B)



C)

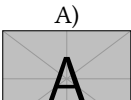


D)

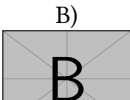


* E)[note]

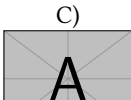
2. Question with images.



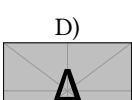
A)




B)



C)




D)




* E)[note]


3. Question with images.




A)




B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

The command `\getkeyans` prints the “stored content” in *prop list* `{<store name>}` defined by `save-ans` key in the `<position>` returned by the `show-pos` key. The “stored content” can only be accessed *after* it is stored, if `{<store name>}` does not exist the command will return an error.

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

`sep = {<code>}` default: *empty*
 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 it will finish executing. 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 <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` environment by default or the `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 $\langle keys \rangle$ set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.
- If we execute `\printkeyans{⟨store name⟩}` and the *sequence* $\{⟨store name⟩\}$ contains any environment `enumext*`, they will start with the $\langle keys \rangle$ set by default unless they are set in the *optional argument* or `save-key` is used to modify it.

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

If we need to set the $\langle keys \rangle$ for the environment `enumext` “saved” in the *sequence* $\{⟨store name⟩\}$ we will use `\setenumext[⟨print , level⟩]{⟨keys⟩}` and if we need to set the $\langle keys \rangle$ 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 \LaTeXe\ is cool? \anskey{Very True!}
  \end{enumext}

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]

2. True False

(a) ~~LaTeXe~~ is cool?

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i. `xsim`

[4]

ii. `exsheets`

[5]

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

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

7 Full examples


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

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

and then you can use the excellent `arara`¹ tool to compile them.

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

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:

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D


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

2. A

3. B

4. A

Example 2

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

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

C) value

B) correct

D) value
2. Second type of questions

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

II. $\alpha = \delta$

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

☒ A

 I only

☒ B

 II only

☒ C

 I and II only

☐ D

 I and III only

☐ E

 I, II, and III
3. Third type of questions

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

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

☒ A

 value

☒ B

 value

☒ C

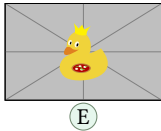
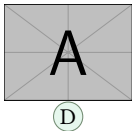
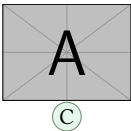
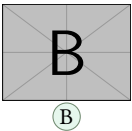
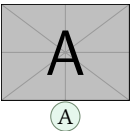
 value

☐ D

 value

☐ E

 value
4. Question with image and label below:



5. Question with image on left side:

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



Test keys

1. B), $x = 5$

2. D

3. C, some note
- * 4. E, A duck

* 5. D, other note

*
- *

*

*

Example 4

A “simple worksheet” using ducks :)

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

- (a) $\alpha > \delta$
- (b) ~~ETX~~ze is cool?

Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class
 - i. `xsim-exam`
 - ii. `xsim`
 - iii. `exsheets`

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- * (b) Yes, dnf

* (c) i. doesn't exist for now :(

* ii. very good

* iii. obsolete

*
- *

*

*

Example 5











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

<div>1</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>3</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>
<div>2</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>4</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>
1. A)	3. B)
2. C)	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` and `hyperref` are required.

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumet*` 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-01.tex` contains the tests for the `enumext` and `enumet*` environments and the support for `minipage` and `multicols` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file  and *tagged* PDF .
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged* PDF .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged* PDF .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta` and `\printkeyans`. Source file  and *tagged* PDF .

9 The way of non-enumerated lists

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

Description indented by label

Here we set `label={}` and we will give a convenient value to `labelsep` and `labelwidth`, for example we can take as reference our *longest label* and pass it as value using:

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

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

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

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

- Something** A short one-line description.
This is an entry *without* a label.
 - Something** A short one-line description.
 - Something long** A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
- If we add `align=right` it will look like this:

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

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

Description with multi-line labels

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

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

Now we just need to set `wrap-label*={\labelbx{#1}}`.

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

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>, 2022.
- [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 $\text{\LaTeX} 3$ 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.

11 Change history

v1.0 2024-10-04 – 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	13
\columnbreak	4, 13	item-join	13
\columnsep	10	item-pos*	13
Commands provide by enumext:		item-star	13
\anskey	11-13	item-sym*	13
\anspic	11, 12, 15, 16	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	17	after	17
\getkeyans	12, 17	before	17
\item*	5-7, 11, 12, 14, 15	sep	17
\item	5-7, 9, 10, 12, 14	start	17
\miniright	10, 11	step	17
\printkeyans	6, 12, 17	stop	17
\setenumextmeta	6	wrapper	17
\setenumext	5-7, 11, 12, 14, 18	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	13
enumXiii	4	force-eol	14
enumXii	4	item-join	13
enumXiv	4	item-pos*	13
enumXi	4	item-star	13
enumXviii	4	item-sym*	13
enumXvii	4	overwrite	14
enumXvi	4	write-env	13
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8
anskey*	11-14, 21	after	9-11
enumet*	21	align	7, 21, 22
enumext*	4-15, 17, 18	base-fix	8
enumext	4-15, 17, 18, 21	before*	9
keyans*	4-9, 11-15, 21	before	9
keyanspic	4, 7, 8, 11-13, 15, 21	below*	9
keyans	4-9, 11-15, 21	below	9
Environments:		check-ans	12
Verbatim	14	columns-sep	4, 10, 21
center	5	columns	4, 8, 10, 21
description	5, 21	first	9
enumerate	1, 3, 5, 23	font	7
figure	5	item-pos*	5, 6
flushleft	5	item-sym*	5, 6
flushright	5	itemindent	9
itemize	5, 21	itemsep	8
keyans	21	labelsep	3-7, 9, 10, 12, 21, 22
list	3, 5, 9, 23	labelwidth	3, 4, 6, 7, 9, 10, 12, 21, 22
minipage	3-5, 8-11, 21, 23	labelwith	5
multicols	3, 4, 10, 21	label	7, 8, 10, 14, 21, 22
quotation	5	list-indent	3, 9
quote	5	list-offset	3, 9, 22
tabbing	5	listparindent	9
table	5	mark-ans	12
task	5	mark-pos	12
trivlist	5	mark-ref	12
verbatim	5	mini-env	4, 8, 10, 11, 21
verse	5	mini-right*	7, 11

mini-right	7, 11, 21	\alph*	7
mini-sep	4, 10, 11	\arabic*	7
no-store	11–13, 21	\roman*	7
noitemsep	8	\labelsep	3, 7
nosep	8, 21	\labelwidth	3, 7
overwrite	13	\linewidth	10
parsep	8, 15	\listparindent	9
partopsep	8		
ref	4, 7, 8, 21	P	
resume*	7, 10, 11	Packages:	
resume	7, 10, 11	enumerate	22
rightmargin	9	enumext	1–5, 7, 15, 21–23
save-ans	4, 6, 10–15, 17, 21	enumitem	3–5, 9, 22, 23
save-key	10, 11, 18	fancyvrb	14
save-ref	4, 7, 12, 13, 17, 21	footnotehyper	5
save-sep	11, 21	geometry	21
series	7, 10, 11	graphicx	21
show-ans	11, 12, 21	hyperref	4, 5, 12, 13, 21–23
show-length	8	l3keys	7
show-pos	11, 12, 17	l3prop	1, 22
start*	10	l3seq	1, 22
start	10	multicol	1, 2, 4, 21, 23
topsep	8, 9	scontents	1, 2, 13, 14, 21
widest	7	task	5, 6
wrap-ans	12	unicode-math	21
wrap-label*	7, 22	xsim	2
wrap-label	7, 21, 22	\parsep	8
wrap-opt	12	\partopsep	8
write-env	14		
		R	
L		\raggedcolumns	4
\label	4	\ref	4
Labels provide by enumext:		\rightmargin	9
\Alph*	7, 14		
\Roman*	7	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}[2024-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-10-04}
7   {1.0}
8   {Enumerate exercise sheets}
```

Finally check if the `multicol` and `scontents` packages are loaded, if not we load it.

```
9 \hook_gput_code:nnn {begindocument} {enumext}
10 {
11   \IfPackageLoadedTF { multicol }
12   {
13     \msg_info:nnn { enumext } { package-load } { multicol }
14   }
15   {
16     \msg_info:nnn { enumext } { package-not-load } { multicol }
17     \RequirePackage{multicol}[2024-05-23]
18   }
19   \IfPackageLoadedTF { scontents }
20   {
21     \msg_info:nnn { enumext } { package-load } { scontents }
22   }
23   {
24     \msg_info:nnn { enumext } { package-not-load } { scontents }
25     \RequirePackage{scontents}
26   }
27 }
```

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.

```

\l__enumext_level_int
\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

```

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

```

28 \int_new:N \l__enumext_level_int
29 \int_new:N \l__enumext_level_h_int
30 \int_new:N \l__enumext_anskey_level_int
31 \int_new:N \l__enumext_keyans_level_int
32 \int_new:N \l__enumext_keyans_level_h_int
33 \int_new:N \l__enumext_keyans_pic_level_int

```

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

```

\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_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).

```

34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_anskey_env_bool
41 \bool_new:N \l__enumext_keyans_env_bool
42 \tl_new:N \g__enumext_start_line_tl
43 \tl_new:N \g__enumext_envir_name_tl
44 \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` (§13.10) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.13).

```

45 \cs_set_protected:Npn \__enumext_tmp:n #1
46 {
47   \tl_new:c { l__enumext_counter_#1_tl }
48 }
49 \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 (§13.13).

```

50 \tl_const:Nn \c__enumext_counter_style_tl
51 { { arabic } { roman } { Roman } { alph } { Alph } }
52 \tl_new:N \l__enumext_ref_key_arg_tl
53 \tl_new:N \l__enumext_ref_the_count_tl
54 \cs_set_protected:Npn \__enumext_tmp:n #1
55 {
56   \tl_new:c { l__enumext_renew_the_count_#1_tl }
57   \tl_new:c { l__enumext_the_counter_#1_tl }
58   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
59 }
60 \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 (§13.24).

```

61 \int_new:N \g__enumext_resume_int
62 \int_new:N \g__enumext_resume_vii_int
63 \tl_new:N \l__enumext_resume_name_tl
64 \bool_new:N \l__enumext_resume_active_bool
65 \tl_new:N \g__enumext_standar_series_tl
66 \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` (§13.14) and `label` (§13.12) keys.

```

67 \dim_new:N \l__enumext_current_widest_dim
68 \tl_new:N \g__enumext_counter_styles_tl
69 \tl_new:N \g__enumext_widest_label_tl
70 \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 (§13.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.37.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72 {
73   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
74   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
75   \dim_new:c { \l__enumext_leftmargin_#1_dim }
76   \dim_new:c { \l__enumext_itemindent_#1_dim }
77 }
78 \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 (§13.21) and `align` key (§13.12).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \skip_new:c { \l__enumext_multicols_above_#1_skip }
82   \skip_new:c { \l__enumext_multicols_below_#1_skip }
83   \skip_new:c { \g__enumext_multicols_right_#1_skip }
84   \str_new:c { \l__enumext_align_label_pos_#1_str }
85 }
86 \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 (§13.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.20, §13.22).

```
87 \int_new:N \g__enumext_minipage_stat_int
88 \skip_new:N \l__enumext_minipage_temp_skip
89 \skip_new:N \l__enumext_minipage_left_skip
90 \skip_new:N \l__enumext_minipage_right_skip
91 \skip_new:N \l__enumext_minipage_after_skip
92 \skip_new:N \g__enumext_minipage_right_skip
93 \skip_new:N \g__enumext_minipage_after_skip
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95 {
96   \dim_new:c { \l__enumext_minipage_left_#1_dim }
97   \bool_new:c { \l__enumext_minipage_active_#1_bool }
98 }
99 \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 (§13.12), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.12). 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 (§13.19).

```
100 \cs_set_protected:Npn \__enumext_tmp:n #1
101 {
102   \bool_new:c { \l__enumext_wrap_label_#1_bool }
103   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
104   \int_new:c { \l__enumext_start_#1_int }
105   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
106   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
107   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
108   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
109   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
110 }
111 \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 (§13.25.1) 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` (§13.29) 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*` (§13.30).

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*` (§13.36) and `\anspic*` (§13.41.2) 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.

```

112 \bool_new:N \l__enumext_store_active_bool
113 \tl_new:N \l__enumext_store_name_tl
114 \tl_new:N \g__enumext_store_name_tl
115 \tl_new:N \l__enumext_store_anskey_arg_tl
116 \tl_new:N \l__enumext_store_anskey_env_tl
117 \tl_new:N \l__enumext_store_anskey_opt_tl
118 \tl_new:N \l__enumext_store_current_label_tl
119 \tl_new:N \l__enumext_store_current_opt_arg_tl
120 \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` (§13.47).

```

121 \tl_new:N \l__enumext_setkey_tmpa_tl
122 \tl_new:N \l__enumext_setkey_tmpb_tl
123 \int_new:N \l__enumext_setkey_tmpa_int
124 \seq_new:N \l__enumext_setkey_tmpa_seq
125 \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 (§13.46) and `\foreachkeyans` command (§13.49).

```

126 \tl_new:N \l__enumext_meta_path_tl
127 \seq_new:N \l__enumext_foreach_print_seq
128 \tl_new:N \l__enumext_foreach_name_prop_tl
129 \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
\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` (§13.46), `show-pos` key (§13.26), `item-sym*` key (§13.34), `save-key` key (§13.26.2) and “*storing structure*”.

```

130 \tl_new:N \l__enumext_print_keyans_starred_tl
131 \bool_new:N \l__enumext_print_keyans_star_bool
132 \str_new:N \l__enumext_mark_position_str
133 \tl_new:N \g__enumext_item_symbol_aux_tl
134 \cs_set_protected:Npn \__enumext_tmp:n #1
135 {
136   \tl_new:c { \l__enumext_print_keyans_#1_tl }
137   \tl_new:c { \l__enumext_store_save_key_#1_tl }
138   \bool_new:c { \l__enumext_store_save_key_#1_bool }
139   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
140 }
141 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_keyans_pic_star_bool
\l__enumext_anspic_mini_pos_str
\g__enumext_keyans_pic_parsep_skip
\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 (§13.41.1).

```

142 \seq_new:N \l__enumext_anspic_args_seq
143 \dim_new:N \l__enumext_anspic_mini_width_dim
144 \int_new:N \l__enumext_anspic_above_int
145 \int_new:N \l__enumext_anspic_below_int
146 \bool_new:N \l__enumext_keyans_pic_star_bool
147 \str_new:N \l__enumext_anspic_mini_pos_str
148 \skip_new:N \g__enumext_keyans_pic_parsep_skip
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.)

Internal variables used by “*internal check answer*” mechanism (§13.25.3) used by the `check-ans` and `no-store` 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 \int_new:N \g__enumext_check_starred_cmd_int
157 \int_new:N \g__enumext_item_anskey_int
158 \int_new:N \g__enumext_item_number_int
159 \bool_new:N \l__enumext_item_number_bool
160 \int_new:N \g__enumext_item_answer_diff_int

```

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

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

161 \bool_new:N \l__enumext_hyperref_bool
162 \bool_new:N \l__enumext_footnotes_key_bool

```

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

Internal variables used by `save-ref` key (§13.26). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the $\langle labels \rangle$ 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` (§13.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

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

Internal variables used for redefinition of `\footnote` (§13.42.4).

```

171 \int_new:N \g__enumext_footnote_int
172 \seq_new:N \g__enumext_footnote_arg_seq
173 \seq_new:N \g__enumext_footnote_int_seq

```

(End of definition for `\g__enumext_footnote_int`, `\g__enumext_footnote_arg_seq`, and `\g__enumext_footnote_int_seq`.)

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

```

174 \cs_set_protected:Npn \__enumext_tmp:n #1
175 {
176   \bool_new:c { l__enumext_item_starred_#1_bool }
177   \int_new:c { l__enumext_item_column_pos_#1_int }
178   \int_new:c { g__enumext_item_count_all_#1_int }
179   \int_new:c { l__enumext_joined_item_#1_int }
180   \int_new:c { l__enumext_joined_item_aux_#1_int }
181   \int_new:c { l__enumext_tmpa_#1_int }
182   \dim_new:c { l__enumext_tmpa_#1_dim }
183   \box_new:c { l__enumext_item_text_#1_box }
184   \dim_new:c { l__enumext_joined_width_#1_dim }
185   \dim_new:c { l__enumext_item_width_#1_dim }
186   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
187   \str_new:c { l__enumext_align_label_#1_str }
188   \bool_new:c { g__enumext_minipage_active_#1_bool }
189   \box_new:c { l__enumext_miniright_code_#1_box }
190   \bool_new:c { g__enumext_minipage_center_#1_bool }
191   \dim_new:c { g__enumext_minipage_right_#1_dim }
192   \skip_new:c { g__enumext_minipage_right_#1_skip }
193 }
194 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

```
\c__enumext_all_envs_clist An internal clist-var variable to run with \__enumext_tmp:n.
195 \clist_const:Nn \c__enumext_all_envs_clist
196 {
197     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
198     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
199 }
```

(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 (§13.46) and `\foreachkeyans` command (§13.49).

```
200 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
201 \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.

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

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

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

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

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

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

(End of definition for `__enumext_level:`.)

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

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

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

`__enumext_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.

```
224 \cs_new_protected:Nn \__enumext_regex_counter_style:
225 {
226     \tl_map_inline:Nn \c__enumext_counter_style_tl
227     {
228         \regex_replace_once:nnN { \c{##1}\* }
229         { \c{##1}\cB{\u{!__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
230     }
231 }
```

(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.

```

232 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
233 {
234     * ~ #2
235     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
236     = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
237 }

```

(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.

```

238 \cs_new_protected:Nn \__enumext_unskip_unkern:
239 {
240     \int_case:nnT { \lastnodetype }
241     {
242         { 11 }
243         {
244             % \typeout{SKIP} \typeout{\the\lastskip}
245             \unskip
246         }
247         { 12 }
248         {
249             % \typeout{KERN} \typeout{\the\lastkern}
250             \unkern
251         }
252     }
253 }

```

(End of definition for `__enumext_unskip_unkern:`.)

13.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`

The function `__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “*true*” only if the environments `enumext` and `enumext*` are nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

`__enumext_is_on_first_level:`

```

254 \cs_new_protected:Nn \__enumext_is_not_nested:
255 {
256     \str_case:en { \@currenvir }
257     {
258         {enumext}
259         {
260             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
261             \bool_lazy_and:nnT
262             { \bool_not_p:n { \g__enumext_standar_bool } }
263             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
264             {
265                 \bool_gset_true:N \g__enumext_standar_bool
266             }
267         }
268         {enumext*}
269         {
270             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
271             \bool_lazy_and:nnT
272             { \bool_not_p:n { \g__enumext_starred_bool } }
273             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
274             {
275                 \bool_gset_true:N \g__enumext_starred_bool
276             }
277         }
278     }
279 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.25.1), `\l__enumext_starred_first_bool` (§13.25.1) and `\l__enumext_anskey_env_bool` (§13.30) 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.

```

280 \cs_new_protected:Nn \__enumext_is_on_first_level:
281 {
282   \bool_lazy_all:nT
283   {
284     { \bool_if_p:N \g__enumext_standar_bool }
285     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
286     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
287   }
288   {
289     \bool_set_true:N \l__enumext_standar_first_bool
290     \bool_set_true:N \l__enumext_anskey_env_bool
291     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
292     \tl_gset:Ne \g__enumext_start_line_tl
293     {
294       on ~ line ~ \exp_not:V \inputlineno
295     }
296   }
297   \bool_lazy_all:nT
298   {
299     { \bool_if_p:N \g__enumext_starred_bool }
300     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
301     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
302   }
303   {
304     \bool_set_true:N \l__enumext_starred_first_bool
305     \bool_set_true:N \l__enumext_anskey_env_bool
306     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
307     \tl_gset:Ne \g__enumext_start_line_tl
308     {
309       on ~ line ~ \exp_not:V \inputlineno
310     }
311   }
312 }

```

(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.

```

313 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
314 {
315   \str_case:en { \@currenvir }
316   {
317     {keyans}
318     {
319       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
320       \tl_set:Ne \l__enumext_check_start_line_env_tl
321       {
322         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
323       }
324     }
325     {keyans*}
326     {
327       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
328       \tl_set:Ne \l__enumext_check_start_line_env_tl
329       {
330         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
331       }
332     }
333     {keyanspic}
334     {
335       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
336       \tl_set:Ne \l__enumext_check_start_line_env_tl
337       {
338         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
339       }
340     }
341   }
342 }

```

(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.

```

343 \cs_new_protected:Nn \__enumext_reset_global_vars:
344 {
345   \__enumext_reset_global_int:
346   \__enumext_reset_global_bool:
347   \__enumext_reset_global_tl:
348 }
349 \cs_new_protected:Nn \__enumext_reset_global_int:
350 {
351   \int_gzero:N \g__enumext_item_number_int
352   \int_gzero:N \g__enumext_item_anskey_int
353   \int_gzero:N \g__enumext_item_answer_diff_int
354 }
355 \cs_new_protected:Nn \__enumext_reset_global_bool:
356 {
357   \bool_gset_false:N \g__enumext_check_ans_key_bool
358   \bool_gset_false:N \g__enumext_standar_bool
359   \bool_gset_false:N \g__enumext_starred_bool
360 }
361 \cs_new_protected:Nn \__enumext_reset_global_tl:
362 {
363   \tl_gclear:N \g__enumext_store_name_tl
364   \tl_gclear:N \g__enumext_start_line_tl
365   \tl_gclear:N \g__enumext_envir_name_tl
366 }

```

(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.

```

367 \cs_new_protected:Nn \__enumext_log_global_vars:
368 {
369   \msg_log:nneeee { enumext } { prop-seq-int-hook }
370   { \g__enumext_store_name_tl }
371   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
372   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
373   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
374 }

```

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.

```

375 \cs_new_protected:Nn \__enumext_log_answer_vars:
376 {
377   \msg_log:nneee { enumext } { item-answer-hook }
378   { \int_use:N \g__enumext_item_number_int }
379   { \int_use:N \g__enumext_item_anskey_int }
380   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
381 }

```

(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`).

```

\__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`, the function `__enumext_item_std:w` is a copy of the `\item` command.

```

382 \__enumext_at_begin_document:n
383 {
384     \cs_new_eq:NN \__enumext_start_list:nn \list
385     \cs_new_eq:NN \__enumext_stop_list: \endlist
386     \NewCommandCopy \__enumext_item_std:w \item
387 }

```

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

```

388 \__enumext_at_begin_document:n
389 {
390     \cs_new_eq:NN \__enumext_minipage:w \minipage
391     \cs_new_eq:NN \__enumext_endminipage: \endminipage
392 }

```

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

13.7 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. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§13.38) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.43)

```

393 \cs_new_protected:Nn \__enumext_internal_mini_page:
394 {
395     \int_compare:nNnT { \__enumext_level_int } = { 0 }
396     {
397         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
398         {
399             \__enumext_minipage:w [ t ] { ##1 }
400             \legacy_if_gset_false:n { @minipage }
401             \skip_vertical:N \c_zero_skip
402         }
403         {
404             \skip_vertical:N \c_zero_skip
405             \__enumext_endminipage:
406         }
407     }
408 }

```

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

13.8 Compatibility with hyperref and footnotehyper

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

```

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

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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”.

```

411 \cs_new_protected:Nn \__enumext_after_hyperref:
412 {
413     \IfPackageLoadedTF { hyperref }
414     {
415         \msg_info:nnn { enumext } { package-load } { hyperref }
416         \bool_set_true:N \l__enumext_hyperref_bool
417         \IfHyperBoolean{hyperfootnotes}
418         {
419             % \typeout{hyperfootnotes=true}
420             \bool_set_true:N \l__enumext_footnotes_key_bool
421         }
422     }

```

```

423         % \typeout{hyperfootnotes=false}
424     }
425 }
426 { }

```

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`.

```

427 \bool_if:NT \l__enumext_footnotes_key_bool
428 {
429     \IfPackageLoadedTF { footnotehyper }
430     {
431         \msg_info:nnn { enumext } { package-load } { footnotehyper }
432     }
433     {
434         % \typeout{No ~ footnotehyper ~ load}
435         % \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
436         \bool_set_false:N \l__enumext_footnotes_key_bool
437     }
438 }

```

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.

```

439 \bool_if:NTF \l__enumext_hyperref_bool
440 {
441     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
442     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
443 }
444 {
445     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
446     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
447 }
448 }

```

(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.

```

449 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
450 {
451     \protected@write \@auxout { }
452     {
453         \token_to_str:N \newlabel {#1}
454         {
455             {#2}
456             \bool_if:NT \l__enumext_hyperref_bool
457             { { \thepage } {#2} {#1} }
458             { }
459         }
460     }
461     \__enumext_hypertarget:nn {#1} { }
462     \__enumext_phantomsection:
463 }

```

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

13.9 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.

```

464 \dim_zero_new:N \itemwidth

```

13.10 Definition of counters

```
\__enumext_define_counters:Nn
\__enumext_define_counters:cn
```

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.

```
465 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
466 {
467   \cs_if_exist:cTF { c@ #2 }
468   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
469   {
470     \tl_set:Nn #1 { #2 }
471     \newcounter { #2 }
472   }
473 }
```

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

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.

```
enumXi    474 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii   475 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii  476 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv   477 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv    478 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi   479 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii  480 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii 481 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
```

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

13.11 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.

```
482 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
483 {
484   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
485   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
486 }
487 \__enumext_register_counter_style:Nn \arabic { 0 }
488 \__enumext_register_counter_style:Nn \Alph { M }
489 \__enumext_register_counter_style:Nn \alph { m }
490 \__enumext_register_counter_style:Nn \Roman { VIII }
491 \__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.

```
492 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
493 {
494   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
495   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
496 }
497 \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.

```
498 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
```

```

499 {
500   \tl_clear_new:N #1
501   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
502   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
503   \tl_map_inline:Nn \g__enumext_counter_styles_tl
504     {
505       \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
506       \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
507       { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
508     }
509   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
510   { \tl_use:N \g__enumext_widest_label_tl }
511   \tl_set_eq:cN { the #2 } #1
512 }
513 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

13.12 Setting keys associated with label

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

```

514 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
515 {
516   \keys_define:nn { enumext / #1 }
517   {
518     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
519     font .value_required:n = true,
520     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
521     labelsep .initial:n = {0.3333em},
522     labelsep .value_required:n = true,
523     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
524     labelwidth .value_required:n = true,
525     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
526     wrap-label .initial:n = {##1},
527     wrap-label .value_required:n = true,
528     wrap-label* .code:n = {
529       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
530       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
531     },
532     wrap-label* .value_required:n = true,
533   }
534 }
535 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

🔗 In this point, the following are set `__enumext_wrapper_label_X:n` which will be used by `__enumext_make_label:` for the different levels of the `enumext` environment and is set to `__enumext_wrapper_label_v:n` which will be used by `__enumext_keyans_make_label:` for `keyans` and `keyanspic` environments.

align The `align` key is implemented differently for “starred” and “non starred” environments.

```

536 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
537 {
538   \keys_define:nn { enumext / #1 }
539   {
540     align .choice:,
541     align / left .code:n =
542       {
543         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
544         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
545         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
546       },
547     align / right .code:n =
548       {
549         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
550         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
551         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
552       },
553     align / center .code:n =
554       {
555         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }

```

```

556         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
557         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
558     },
559     align / unknown .code:n =
560         \msg_error:nneee { enumext } { unknown-choice }
561         { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
562     align .initial:n = left,
563     align .value_required:n = true,
564 }
565 }
566 \clist_map_inline:nn
567 {
568     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
569 }
570 { \__enumext_tmp:nn #1 }

```

For compatibility with \LaTeX tagged PDF we must set `\l__enumext_align_label_pos_X_str`. When tagged PDF is active `\make_label` is redefined and the only way to get the `align` key to work correctly is by using `\makebox`.

```

571 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
572 {
573     \keys_define:nn { enumext / #1 }
574     {
575         align .choice:,
576         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
577         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
578         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
579         align / unknown .code:n =
580             \msg_error:nneee { enumext } { unknown-choice }
581             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
582         align .initial:n = left,
583         align .value_required:n = true,
584     }
585 }
586 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

13.13 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 `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “*label and ref*” system.

13.13.1 Define and set label and ref keys for enumext environment

`label` 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.

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
587 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
588 {
589     \keys_define:nn { enumext / #1 }
590     {
591         label .code:n = {
592             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
593             { l__enumext_counter_#2_tl } {##1}
594             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
595             \l__enumext_current_widest_dim
596         },
597         label .initial:n = #3,
598         label .value_required:n = true,
599         ref .code:n = \__enumext_standar_ref:n {##1},
600         ref .value_required:n = true,
601     }
602 }
603 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
604 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
605 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
606 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

`__enumext_standar_ref:n` The `__enumext_standar_ref:n` first we will pass the key argument to `\l__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 `\l__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will

return the modified `\l__enumext_ref_key_arg_tl` and we make the value of `\l__enumext_ref_the_count_tl` the same as that `\l__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `\l__enumext_renew_the_count_X_tl` with the renewed command.

```

607 \cs_new_protected:Npn \__enumext_standar_ref:n #1
608 {
609   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
610   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
611   {
612     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
613   }
614   {
615     \tl_set_eq:Nc
616     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
617     \__enumext_regex_counter_style:
618     \tl_set_eq:Nc
619     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
620     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
621     {
622       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
623     }
624   }
625 }

```

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

```

626 \cs_new_protected:Nn \__enumext_standar_ref:
627 {
628   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
629   {
630     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
631   }
632 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
633 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
634 {
635   \keys_define:nn { enumext / #1 }
636   {
637     label .code:n = {
638       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
639       { \l__enumext_counter_#2_tl } {##1}
640       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
641       \l__enumext_current_widest_dim
642     },
643     label .initial:n = #3,
644     label .value_required:n = true,
645     ref .code:n = \__enumext_starred_ref:n {##1},
646     ref .value_required:n = true,
647   }
648 }
649 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
650 \__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`.

```

651 \cs_new_protected:Npn \__enumext_starred_ref:n #1
652 {
653   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
654   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
655   {
656     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
657     {
658       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
659     }
660     {
661       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl

```

```

662         \__enumext_regex_counter_style:
663         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
664         \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
665         {
666             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
667         }
668     }
669 }
670 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
671 {
672     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
673     {
674         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
675     }
676     {
677         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
678         \__enumext_regex_counter_style:
679         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
680         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
681         {
682             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
683         }
684     }
685 }
686 }

```

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.

```

687 \cs_new_protected:Nn \__enumext_starred_ref:
688 {
689     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
690     {
691         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
692         {
693             \tl_use:N \l__enumext_renew_the_count_vii_tl
694         }
695     }
696     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
697     {
698         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
699         {
700             \tl_use:N \l__enumext_renew_the_count_viii_tl
701         }
702     }
703 }

```

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

13.13.3 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` and `ref` key. The `keyanspic` environment use the same *label* as the `keyans` environment.

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
704 \keys_define:nn { enumext / keyans }
705 {
706     label .code:n = {
707         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
708         { \l__enumext_counter_v_tl } {#1}
709         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
710         \l__enumext_current_widest_dim
711         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
712         { \l__enumext_counter_vi_tl } {#1}
713         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
714         \l__enumext_current_widest_dim
715     },
716     label .initial:n = \Alph*,
717     label .value_required:n = true,
718     ref .code:n = \__enumext_keyans_ref:n {#1},
719     ref .value_required:n = true,
720 }

```

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

_enumext_keyans_ref:n
_enumext_keyans_ref:

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

```

721 \cs_new_protected:Npn \\_enumext_keyans_ref:n #1
722 {
723   \tl_set:Nn \\_enumext_ref_key_arg_tl {#1}
724   \tl_if_empty:NTF \\_enumext_ref_key_arg_tl
725   {
726     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
727   }
728   {
729     \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_counter_v_tl
730     \\_enumext_regex_counter_style:
731     \tl_set_eq:NN \\_enumext_ref_the_count_tl \\_enumext_the_counter_v_tl
732     \tl_put_right:Ne \\_enumext_renew_the_count_v_tl
733     {
734       \exp_not:N \renewcommand { \exp_not:V \\_enumext_ref_the_count_tl } { \exp_not:V \\_
735     }
736   }
737 }

```

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

```

738 \cs_new_protected:Nn \\_enumext_keyans_ref:
739 {
740   \tl_if_empty:NF \\_enumext_renew_the_count_v_tl
741   {
742     \tl_use:N \\_enumext_renew_the_count_v_tl
743   }
744 }

```

(End of definition for _enumext_keyans_ref:n and _enumext_keyans_ref:.)

13.14 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: _enumext_label_X_tl
#2: _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.

```

745 \cs_new_protected:Npn \\_enumext_start_from:NNn #1 #2 #3
746 {
747   \\_enumext_if_is_int:nTF { #3 }
748   {
749     \int_set:Nn #2 {#3}
750   }
751   {
752     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
753     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
754     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
755     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
756   }
757 }
758 \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: _enumext_label_X_tl
#3: _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.

```

759 \cs_new_protected:Npn \\_enumext_widest_from:nNNn #1 #2 #3 #4
760 {
761   \\_enumext_if_is_int:nTF {#4}
762   {
763     \setcounter{enumX#1} { #4 }

```

```

764     }
765     {
766         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
767         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
768         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
769         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
770     }
771     \__enumext_label_width_by_box:cv
772     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
773 }
774 \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.

```

775 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
776 {
777     \keys_define:nn { enumext / #1 }
778     {
779         start* .code:n = {
780             \__enumext_start_from:ccn
781             { l__enumext_label_#2_tl }
782             { l__enumext_start_#2_int } {##1}
783         },
784         start* .value_required:n = true,
785         start .code:n = {
786             \__enumext_start_from:cce
787             { l__enumext_label_#2_tl }
788             { l__enumext_start_#2_int } { \int_eval:n {##1} }
789         },
790         start .initial:n = 1,
791         start .value_required:n = true,
792         widest .code:n = {
793             \__enumext_widest_from:nccn {#2}
794             { l__enumext_label_#2_tl }
795             { l__enumext_labelwidth_#2_dim } {##1}
796         },
797         widest .value_required:n = true,
798     }
799 }
800 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.15 Setting keys for vertical spaces

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

```

801 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
802 {
803     \keys_define:nn { enumext / #1 }
804     {
805         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
806         topsep .initial:n = {#3},
807         topsep .value_required:n = true,
808         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
809         partopsep .initial:n = {#4},
810         partopsep .value_required:n = true,
811         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
812         parsep .initial:n = {#5},
813         parsep .value_required:n = true,
814         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
815         itemsep .initial:n = {#6},
816         itemsep .value_required:n = true,
817         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
818         noitemsep .value_forbidden:n = true,
819         nosepe .meta:n = {
820             itemsep = 0pt, parsep = 0pt,
821             topsep = 0pt, partopsep = 0pt,
822         },

```

```

823         nosepl      .value_forbidden:n = true,
824     }
825 }

```

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

```

826 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
827 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
828 { 4.0pt plus 2.0pt minus 1.0pt }
829 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
830 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
831 { 2.0pt plus 1.0pt minus 1.0pt }
832 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
833 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
835 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
836 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
837 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
838 { 2.0pt plus 1.0pt minus 1.0pt }
839 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
840 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
841 { 4.0pt plus 2.0pt minus 1.0pt }
842 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
843 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
844 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.16 Setting base-fix key

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

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

```

845 \keys_define:nn { enumext / level-1 }
846 {
847     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
848     base-fix .initial:n = false,
849     base-fix .value_forbidden:n = true,
850 }

```

(End of definition for `base-fix`.)

`__enumext_nested_base_line_fix:` The function `__enumext_nested_base_line_fix:` passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.38) will be responsible for applying the *baseline correction* and adjusting the `\keys` for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§13.46).

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

```

851 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
852 {
853     \bool_lazy_all:nT
854     {
855         { \bool_if_p:N \l__enumext_starred_first_bool }
856         { \bool_if_p:N \l__enumext_base_line_fix_bool }
857         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
858     }
859     {
860         \mode_leave_vertical:
861         \vspace { -\dim_eval:n { \baselineskip + \parsep } }
862     }

```

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

```

863     \bool_lazy_and:nnT
864     { \bool_if_p:N \l__enumext_starred_first_bool }
865     { \bool_if_p:N \l__enumext_print_keyans_star_bool }
866     {

```

```

867     \mode_leave_vertical:
868     \skip_vertical:n { -\baselineskip }
869     \skip_vertical:N \c_zero_skip
870 }

```

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

```

871     \keys_set:nn { enumext / level-1 }
872     {
873         topsep = 0pt, above = 0pt, above* = 0pt,
874     }
875     \bool_set_false:N \l__enumext_base_line_fix_bool
876 }

```

(End of definition for `__enumext_nested_base_line_fix:`.)

13.17 Setting keys for horizontal spaces

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

```

877 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
878 {
879     \keys_define:nn { enumext / #1 }
880     {
881         itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
882         itemindent .value_required:n = true,
883         rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
884         rightmargin .value_required:n = true,
885         listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
886         listparindent .value_required:n = true,
887         list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
888         list-offset .value_required:n = true,
889         list-indent .code:n =
890             \bool_set_true:c { \l__enumext_leftmargin_tmp_#2_bool }
891             \dim_set:cn { \l__enumext_leftmargin_tmp_#2_dim } {##1},
892         list-indent .value_required:n = true,
893     }
894 }
895 \clist_map_inline:nn
896 {
897     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
898 }
899 { \__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.

```

900 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
901 {
902     \keys_define:nn { enumext / #1 }
903     {
904         itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
905         itemindent .value_required:n = true,
906         rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
907         rightmargin .value_required:n = true,
908         listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
909         listparindent .value_required:n = true,
910         list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
911         list-offset .value_required:n = true,
912         list-indent .meta:n = { list-offset = ##1 },
913         list-indent .value_required:n = true,
914     }
915 }
916 \clist_map_inline:nn
917 {
918     {enumext*}{vii}, {keyans*}{viii}
919 }
920 { \__enumext_tmp:nn #1 }

```


13.17.1 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.

```

921 \cs_set_protected:Nn \__enumext_fake_item_indent:
922 {
923   \dim_compare:nNnT
924     { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
925     >
926     { \c_zero_dim }
927   {
928     \tl_set:ce { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
929     {
930       \exp_not:N \mode_leave_vertical:
931       \exp_not:n { \skip_horizontal:n }
932       { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
933       \ignorespaces
934     }
935   }
936 }
937 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
938 {
939   \dim_compare:nNnT
940     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
941   {
942     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
943     {
944       \exp_not:N \mode_leave_vertical:
945       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
946       \ignorespaces
947     }
948   }
949 }
950 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
951 {
952   \dim_compare:nNnT
953     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
954   {
955     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
956     {
957       \exp_not:N \mode_leave_vertical:
958       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
959       \ignorespaces
960     }
961   }
962 }
963 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
964 {
965   \dim_compare:nNnT
966     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
967   {
968     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
969     {
970       \exp_not:N \mode_leave_vertical:
971       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
972       \ignorespaces
973     }
974   }
975 }

```

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

13.18 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.

```

976 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
977 {

```

```

978     \keys_define:nn { enumext / #1 }
979     {
980         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
981         show-length .initial:n = false,
982     }
983 }
984 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for show-length.)

13.19 Setting before, after and first keys

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

```

before  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
before* {
after    \keys_define:nn { enumext / #1 }
first    {
985     {
986         \keys_define:nn { enumext / #1 }
987         {
988             before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
989             before .value_required:n = true,
990             before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
991             before* .value_required:n = true,
992             after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
993             after .value_required:n = true,
994             first .tl_set:c = { l__enumext_after_list_args_#2_tl },
995             first .value_required:n = true,
996         }
997     }
998 }
999 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for before and others.)

13.19.1 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>}.
 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>}}.

```

1000 \cs_new_protected:Nn \__enumext_before_args_exec:
1001 {
1002     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1003 }

```

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>}}.

```

1004 \cs_new_protected:Nn \__enumext_before_keys_exec:
1005 {
1006     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1007 }

```

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

```

1008 \cs_new_protected:Nn \__enumext_after_stop_list:
1009 {
1010     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1011 }

```

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.

```

1012 \cs_new_protected:Nn \__enumext_after_args_exec:
1013 {
1014     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1015 }

```

(End of definition for __enumext_before_args_exec: and others.)

13.19.2 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:
1016 \cs_new_protected:Nn __enumext_before_args_exec_v:
1017 {
1018   \tl_use:N \l__enumext_before_starred_key_v_tl
1019 }
1020 \cs_new_protected:Nn __enumext_before_keys_exec_v:
1021 {
1022   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1023 }
1024 \cs_new_protected:Nn __enumext_after_stop_list_v:
1025 {
1026   \tl_use:N \l__enumext_after_stop_list_v_tl
1027 }
1028 \cs_new_protected:Nn __enumext_after_args_exec_v:
1029 {
1030   \tl_use:N \l__enumext_after_list_args_v_tl
1031 }

```

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

13.19.3 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:
1032 \cs_new_protected:Nn __enumext_before_args_exec_vii:
1033 {
1034   \tl_use:N \l__enumext_before_starred_key_vii_tl
1035 }
1036 \cs_new_protected:Nn __enumext_before_args_exec_viii:
1037 {
1038   \tl_use:N \l__enumext_before_starred_key_viii_tl
1039 }
1040 \cs_new_protected:Nn __enumext_before_keys_exec_vii:
1041 {
1042   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1043 }
1044 \cs_new_protected:Nn __enumext_before_keys_exec_viii:
1045 {
1046   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1047 }
1048 \cs_new_protected:Nn __enumext_after_stop_list_vii:
1049 {
1050   \tl_use:N \l__enumext_after_stop_list_vii_tl
1051 }
1052 \cs_new_protected:Nn __enumext_after_stop_list_viii:
1053 {
1054   \tl_use:N \l__enumext_after_stop_list_viii_tl
1055 }
1056 \cs_new_protected:Nn __enumext_after_args_exec_vii:
1057 {
1058   \tl_use:N \l__enumext_after_list_args_vii_tl
1059 }
1060 \cs_new_protected:Nn __enumext_after_args_exec_viii:
1061 {
1062   \tl_use:N \l__enumext_after_list_args_viii_tl
1063 }

```

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

13.20 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.

```

1064 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1065 {
1066   \keys_define:nn { enumext / #1 }
1067   {
1068     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1069     mini-env .value_required:n = true,
1070     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },

```

```

1071     mini-sep    .initial:n = 0.3333em,
1072     mini-sep    .value_required:n = true,
1073     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1074     columns-sep .value_required:n = true,
1075     columns     .int_set:c = { l__enumext_columns_#2_int },
1076     columns     .initial:n = 1,
1077     columns     .value_required:n = true,
1078   }
1079 }
1080 \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.

```

1081 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1082 {
1083   \keys_define:nn { enumext / #1 }
1084   {
1085     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1086     mini-right .value_required:n = true,
1087     mini-right* .code:n = {
1088       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1089       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1090     },
1091     mini-right* .value_required:n = true,
1092   }
1093 }
1094 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.21 Adjustment of vertical spaces for multicols

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



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

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

🔗 I should make it clear that the implementation here is a “bit questionable”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values 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.

13.21.1 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*.

```

1095 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1096 {
1097   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1098   {
1099     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1100   }
1101   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1102   {
1103     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1104   }

```

```

1105   \__enumext_add_pre_parsep:
1106   }

```

(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*.

```

1107 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1108 {
1109   \int_case:nn { \l__enumext_level_int }
1110   {
1111     { 2 }{
1112       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1113       {
1114         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1115         {
1116           \l__enumext_parsep_i_skip
1117         }
1118       }
1119     }
1120     { 3 }{
1121       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1122       {
1123         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1124         {
1125           \l__enumext_parsep_ii_skip
1126         }
1127       }
1128     }
1129     { 4 }{
1130       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1131       {
1132         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1133         {
1134           \l__enumext_parsep_iii_skip
1135         }
1136       }
1137     }
1138   }
1139 }

```

(End of definition for __enumext_add_pre_parsep:.)

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

```

1140 \cs_new_protected:Nn \__enumext_multi_addvspace:
1141 {
1142   \__enumext_multi_set_vskip:
1143   \mode_if_vertical:T
1144   {
1145     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1146     {
1147       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1148     }
1149     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1150     {
1151       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1152     }
1153   }
1154   \par\nopagebreak
1155   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1156 }

```

(End of definition for __enumext_multi_addvspace:.)

13.21.2 Adjustment of vertical spaces for multicols in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

The function `__enumext_keyans_multi_set_vskip:` will take care of determining the “adjusted spaces” 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`.

```

1157 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1158 {
1159   \skip_set:Nn \l__enumext_multicols_above_v_skip
1160   {
1161     \l__enumext_topsep_v_skip
1162   }
1163   \skip_set:Nn \l__enumext_multicols_below_v_skip
1164   {
1165     \l__enumext_topsep_v_skip
1166   }
1167 }
1168 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1169 {
1170   \__enumext_keyans_multi_set_vskip:
1171   \mode_if_vertical:T
1172   {
1173     \skip_add:Nn \l__enumext_multicols_above_v_skip
1174     {
1175       \skip_use:N \l__enumext_partopsep_v_skip
1176     }
1177     \skip_add:Nn \l__enumext_multicols_below_v_skip
1178     {
1179       \skip_use:N \l__enumext_partopsep_v_skip
1180     }
1181   }
1182   \par\nopagebreak
1183   \addvspace{ \l__enumext_multicols_above_v_skip }
1184 }

```

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

13.22 Adjustment of vertical spaces for minipage

When nesting a “list environment” within the `\minipage` environment, the values of the “vertical spaces” are lost. Graphically it can be seen like in the figure 8.



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

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

Here there are several complications that we must circumvent, the `\minipage` environment eliminates the “top” spaces, the `\multicols` environment can be nested in the `\minipage` environment, the “top” and “bottom” spaces are affected when `\topsep=0pt` and to this is added the `\partopsep` parameter that comes into action according to whether \TeX is in *horizontal mode* or *vertical mode*. Depending on these cases, small adjustments must be made using `\vspace` and `\addvspace` to obtain the “desired vertical spacing”.

- Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `\minipage` environment is quite complicated, even more if `\multicols` is nested. The setting of the values was more “trial and error” (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.

13.22.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

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

```

1185 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1186 {
1187   \skip_set:Nn \l__enumext_minipage_right_skip

```



```

1188     {
1189       \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1190     }
1191     \mode_if_vertical:T
1192     {
1193       \skip_add:Nn \l__enumext_minipage_right_skip
1194       {
1195         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1196       }
1197     }
1198     \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:`.

```

1199     \skip_set_eq:cN
1200     { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1201     \skip_set_eq:cN
1202     { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1203     \__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`.

```

1204     \int_compare:nNnT
1205     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1206     {
1207       \skip_zero:N \topskip
1208       \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1209     }
1210 }

```

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*.

```

1211 \cs_new_protected:Nn \__enumext_minipage_add_space:
1212 {
1213   \__enumext_minipage_set_skip:
1214   \__enumext_unskip_unkern:
1215   \mode_if_vertical:TF
1216   {
1217     \nopagebreak\nointerlineskip
1218   }
1219   {
1220     \par\nopagebreak\nointerlineskip
1221     \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1222   }
1223   \int_compare:nNnTF
1224   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1225   {
1226     \addvspace{ 0.445\box_ht:N \strutbox }
1227   }
1228   {
1229     \addvspace{ 0.250\box_ht:N \strutbox }
1230   }
1231 }

```

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

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

```

1232 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1233 {
1234   \int_case:nn { \l__enumext_level_int }
1235   {
1236     { 2 }{
1237       \skip_if_eq:nnTF
1238       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1239       {
1240         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1241         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }

```

```

1242     }
1243     {
1244         \dim_compare:nNnT
1245         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1246         {
1247             \skip_sub:Nn
1248             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1249             \skip_sub:Nn
1250             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1251             \skip_add:Nn
1252             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1253             \skip_add:Nn
1254             \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1255         }
1256         \dim_compare:nNnT
1257         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1258         {
1259             \skip_set:Nn \l__enumext_minipage_temp_skip
1260             {
1261                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1262             }
1263             \skip_sub:Nn
1264             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1265             \skip_sub:Nn
1266             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1267             \skip_add:Nn
1268             \l__enumext_minipage_after_skip
1269             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1270             \skip_add:Nn
1271             \l__enumext_multicols_below_ii_skip
1272             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1273         }
1274     }
1275 }
1276 { 3 }{
1277     \skip_if_eq:nnTF
1278     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1279     {
1280         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1281         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1282     }
1283     {
1284         \dim_compare:nNnT
1285         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1286         {
1287             \skip_sub:Nn
1288             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1289             \skip_sub:Nn
1290             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1291             \skip_add:Nn
1292             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1293             \skip_add:Nn
1294             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1295         }
1296         \dim_compare:nNnT
1297         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1298         {
1299             \skip_set:Nn \l__enumext_minipage_temp_skip
1300             {
1301                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1302             }
1303             \skip_sub:Nn
1304             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1305             \skip_sub:Nn
1306             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1307             \skip_add:Nn
1308             \l__enumext_minipage_after_skip
1309             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1310             \skip_add:Nn
1311             \l__enumext_multicols_below_iii_skip
1312             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }

```

```

1313         }
1314     }
1315 }
1316 { 4 }{
1317     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1318     {
1319         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1320         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1321     }
1322     {
1323         \dim_compare:nNnT
1324         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1325         {
1326             \skip_sub:Nn
1327             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1328             \skip_sub:Nn
1329             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1330             \skip_add:Nn
1331             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1332             \skip_add:Nn
1333             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1334         }
1335         \dim_compare:nNnT
1336         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1337         {
1338             \skip_set:Nn \l__enumext_minipage_temp_skip
1339             {
1340                 \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1341             }
1342             \skip_sub:Nn
1343             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1344             \skip_sub:Nn
1345             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1346             \skip_add:Nn
1347             \l__enumext_minipage_after_skip
1348             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1349             \skip_add:Nn
1350             \l__enumext_multicols_below_iv_skip
1351             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1352         }
1353     }
1354 }
1355 }
1356 }

```

(End of definition for `__enumext_pre_itemsep_skip`.)

13.22.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

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

```

1357 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1358 {
1359     \skip_zero:N \l__enumext_minipage_after_skip
1360     \skip_zero:N \l__enumext_minipage_left_skip
1361     \skip_zero:N \l__enumext_minipage_right_skip
1362     \skip_set:Nn \l__enumext_minipage_right_skip
1363     {
1364         \l__enumext_topsep_v_skip
1365     }
1366     \mode_if_vertical:T
1367     {
1368         \skip_add:Nn \l__enumext_minipage_right_skip
1369         {
1370             \l__enumext_partopsep_v_skip
1371         }
1372     }
1373     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1374     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1375     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1376     \__enumext_keyans_pre_itemsep_skip:

```

```

1377 \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
1378 {
1379   \skip_zero:N \topskip
1380   \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1381 }
1382 }
1383 \cs_new_protected:Nn \l__enumext_keyans_minipage_add_space:
1384 {
1385   \l__enumext_keyans_minipage_set_skip:
1386   \l__enumext_unskip_unkern:
1387   \mode_if_vertical:TF
1388   {
1389     \nopagebreak\nointerlineskip
1390   }
1391   {
1392     \par\nopagebreak\nointerlineskip
1393     \skip_zero:N \l__enumext_partopsep_v_skip
1394   }
1395   \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }
1396   {
1397     \addvspace{ 0.445\box_ht:N \strutbox }
1398   }
1399   {
1400     \addvspace{ 0.250\box_ht:N \strutbox }
1401   }
1402 }
1403 \cs_new_protected:Nn \l__enumext_keyans_pre_itemsep_skip:
1404 {
1405   \skip_if_eq:nnTF
1406   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1407   {
1408     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1409     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1410   }
1411   {
1412     \dim_compare:nNt
1413     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1414     {
1415       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1416       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1417       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1418       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1419     }
1420     \dim_compare:nNt
1421     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1422     {
1423       \skip_set:Nn \l__enumext_minipage_temp_skip
1424       {
1425         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1426       }
1427       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1428       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1429       \skip_add:Nn \l__enumext_minipage_after_skip
1430       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1431       \skip_add:Nn \l__enumext_multicols_below_v_skip
1432       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1433     }
1434   }
1435 }

```

(End of definition for `\l__enumext_keyans_minipage_set_skip:`, `\l__enumext_keyans_minipage_add_space:`, and `\l__enumext_keyans_pre_itemsep_skip:`.)

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

`\l__enumext_mini_set_vskip_vii:`
`\l__enumext_mini_set_vskip_viii:`

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\l__enumext_mini_page` environment in `enumext*` and `keyans*`.

```

1436 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1437 {
1438   \skip_zero_new:N \l__enumext_minipage_left_skip
1439   \skip_gzero_new:N \l__enumext_minipage_right_skip

```

```

1440 \skip_gzero_new:N \g__enumext_minipage_after_skip
1441 \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1442 {
1443   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1444   \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1445 }
1446 {
1447   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1448   \skip_gset:Nn \g__enumext_minipage_right_skip
1449   {
1450     \l__enumext_topsep_vii_skip
1451   }
1452   \skip_gset:Nn \g__enumext_minipage_after_skip
1453   {
1454     0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1455   }
1456 }
1457 }
1458 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1459 {
1460   \skip_zero_new:N \l__enumext_minipage_after_skip
1461   \skip_zero_new:N \l__enumext_minipage_left_skip
1462   \skip_zero_new:N \l__enumext_minipage_right_skip
1463   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1464   {
1465     \skip_set:Nn \l__enumext_minipage_left_skip
1466     {
1467       0.5\box_dp:N \strutbox
1468     }
1469     \skip_set:Nn \l__enumext_minipage_right_skip
1470     {
1471       \l__enumext_partopsep_viii_skip
1472     }
1473     \skip_set:Nn \l__enumext_minipage_after_skip
1474     {
1475       1.6\box_dp:N \strutbox
1476     }
1477   }
1478   {
1479     \skip_set:Nn \l__enumext_minipage_left_skip
1480     {
1481       0.5875\box_dp:N \strutbox
1482     }
1483     \skip_set:Nn \l__enumext_minipage_right_skip
1484     {
1485       \l__enumext_topsep_viii_skip
1486     }
1487     \skip_set:Nn \l__enumext_minipage_after_skip
1488     {
1489       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1490     }
1491   }
1492 }

```

(End of definition for `__enumext_mini_set_vskip_vii:` and `__enumext_mini_set_vskip_viii:`)

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_page` environment on the *left side* when the *mini-right* key is active in the `enumext*` and `keyans*` environments.

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

```

1493 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1494 {
1495   \__enumext_mini_set_vskip_vii:
1496   \par\nopagebreak
1497   \addvspace { \l__enumext_minipage_left_skip }
1498 }
1499 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1500 {
1501   \__enumext_mini_set_vskip_viii:
1502   \par\nopagebreak

```

```

1503   \addvspace { \l__enumext_minipage_left_skip }
1504 }

```

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

13.22.4 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 \LaTeX 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.

```

1505 \NewDocumentCommand \miniright { s }
1506 {
1507   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1508   {
1509     \msg_error:nnn { enumext } { wrong-miniright-place }
1510   }
1511   % outside
1512   \bool_lazy_and:nnT
1513   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1514   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1515   {
1516     \msg_error:nnn { enumext } { wrong-miniright-place }
1517   }
1518   % starred env
1519   \bool_if:NT \l__enumext_starred_bool
1520   {
1521     \msg_error:nnn { enumext } { wrong-miniright-starred }
1522   }
1523   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1524   {
1525     \__enumext_keyans_mini_right_cmd:n {#1}
1526   }
1527   { \__enumext_mini_right_cmd:n {#1} }
1528 }

```

(End of definition for `\miniright`. This function is documented on page 10.)

`__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.

```

1529 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1530 {
1531   \dim_compare:nNtF
1532   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1533   {
1534     \__enumext_multicols_stop:
1535     \int_compare:nNtF
1536     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1537     {
1538       \par\addvspace{ \l__enumext_minipage_after_skip }
1539     }
1540     \end__enumext_mini_page
1541     \hfill
1542     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1543     \par\nointerlineskip
1544     \addvspace { \l__enumext_minipage_right_skip }
1545     \bool_if:nF {#1}
1546     {
1547       \centering
1548     }
1549     \int_gzero:N \g__enumext_minipage_stat_int
1550   }

```

```

1551     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1552 % paranoia
1553 \RenewDocumentCommand \miniright { s }
1554 {
1555     \msg_error:nn { enumext } { many-miniright-used }
1556 }
1557 }

```

(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.

```

1558 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1559 {
1560     \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1561     {
1562         \__enumext_keyans_multicols_stop:
1563         \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1564         {
1565             \par\addvspace{ \__enumext_minipage_after_skip }
1566         }
1567         \end__enumext_mini_page
1568         \hfill
1569         \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1570         \par\nointerlineskip
1571         \addvspace { \__enumext_minipage_right_skip }
1572         \bool_if:nF {#1}
1573         {
1574             \centering
1575         }
1576         \int_gzero:N \g__enumext_minipage_stat_int
1577     }
1578     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1579 % paranoia
1580 \RenewDocumentCommand \miniright { s }
1581 {
1582     \msg_error:nn { enumext } { many-miniright-used }
1583 }
1584 }

```

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

13.23 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.

```

1585 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1586 {
1587     \keys_define:nn { enumext / #1 }
1588     {
1589         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1590         above .value_required:n = true,
1591         above* .code:n = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1592             \keys_set:nn { enumext / #1 } { above = {#1} },
1593         above* .value_required:n = true,
1594         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1595         below .value_required:n = true,
1596         below* .code:n = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1597             \keys_set:nn { enumext / #1 } { below = {#1} },
1598         below* .value_required:n = true,
1599     }
1600 }
1601 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.23.1 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.

```

1602 \cs_new_protected:Nn \__enumext_vspace_above:
1603 {
1604   \skip_if_eq:nnF
1605   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1606   {
1607     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1608     {
1609       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1610     }
1611     {
1612       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1613     }
1614   }
1615 }

```

(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.

```

1616 \cs_new_protected:Nn \__enumext_vspace_below:
1617 {
1618   \skip_if_eq:nnF
1619   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1620   {
1621     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1622     {
1623       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1624     }
1625     {
1626       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1627     }
1628   }
1629 }

```

(End of definition for `__enumext_vspace_below:`.)

13.23.2 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.

```

1630 \cs_new_protected:Nn \__enumext_vspace_above_v:
1631 {
1632   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1633   {
1634     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1635     {
1636       \vspace*{ \__enumext_vspace_above_v_skip }
1637     }
1638     { \vspace { \__enumext_vspace_above_v_skip } }
1639   }
1640 }

```

(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.

```

1641 \cs_new_protected:Nn \__enumext_vspace_below_v:
1642 {
1643   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1644   {
1645     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1646     {
1647       \vspace*{ \__enumext_vspace_below_v_skip }
1648     }
1649     { \vspace { \__enumext_vspace_below_v_skip } }
1650   }
1651 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

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.

```

1652 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1653 {
1654   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1655   {
1656     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1657     {
1658       \vspace*{ \l__enumext_vspace_above_vii_skip }
1659     }
1660     { \vspace { \l__enumext_vspace_above_vii_skip } }
1661   }
1662 }
1663 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1664 {
1665   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1666   {
1667     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1668     {
1669       \vspace*{ \l__enumext_vspace_above_viii_skip }
1670     }
1671     { \vspace { \l__enumext_vspace_above_viii_skip } }
1672   }
1673 }

```

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

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

```

1674 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1675 {
1676   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1677   {
1678     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1679     {
1680       \vspace*{ \l__enumext_vspace_below_vii_skip }
1681     }
1682     { \vspace { \l__enumext_vspace_below_vii_skip } }
1683   }
1684 }
1685 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1686 {
1687   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1688   {
1689     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1690     {
1691       \vspace*{ \l__enumext_vspace_below_viii_skip }
1692     }
1693     { \vspace { \l__enumext_vspace_below_viii_skip } }
1694   }
1695 }

```

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

13.24 Setting series, resume and resume* keys

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the `<keys>` passed to the *optional argument* of the “first level” of the environments `enumext` and `enumext*`, but, discarding some specific `<keys>`. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in `chat-Tex-SX`

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

```

series
resume
resume*
1696 \cs_set_protected:Npn \__enumext_tmp:n #1
1697 {
1698   \keys_define:nn { enumext / #1 }
1699   {
1700     series .str_set:N = \l__enumext_series_str,
1701     series .value_required:n = true,
1702     resume .code:n = \__enumext_resume_series:n {##1},
1703     resume* .code:n = \__enumext_resume_starred:,

```

```

1704         resume* .value_forbidden:n = true,
1705     }
1706 }
1707 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.24.1 Internal functions for series key

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1708 \cs_new:Npn \__enumext_filter_series:n #1
1709 {
1710     \use:e
1711     {
1712         \keyval_parse:NNn
1713         \__enumext_filter_series_key:n
1714         \__enumext_filter_series_pair:nn {#1}
1715     }
1716 }

```

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

```

1717 \cs_new:Npn \__enumext_filter_series_key:n #1
1718 {
1719     \str_case:nnF {#1}
1720     {
1721         { resume } {} { resume* } {} { base-fix } {}
1722     }
1723     { , { \exp_not:n {#1} } }
1724 }

```

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

```

1725 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1726 {
1727     \str_case:nnF {#1}
1728     {
1729         { series } {} { resume } {} { start } {}
1730         { start* } {} { save-ans } {} { save-key } {}
1731     }
1732     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
1733 }

```

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

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 (§13.38) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.43).

```

1734 \cs_new_protected:Npn \__enumext_parse_series:n #1
1735 {
1736     \str_if_empty:NTF \l__enumext_series_str
1737     {
1738         \bool_if:NF \l__enumext_resume_active_bool
1739         {
1740             \__enumext_resume_last:n {#1}
1741         }
1742     }
1743     {
1744         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1745         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1746         { \__enumext_filter_series:n {#1} }
1747         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1748         {

```

```

1749         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1750     }
1751 }
1752 }

```

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.

```

1753 \cs_new_protected:Npn \__enumext_resume_last:n #1
1754 {
1755     \bool_if:NT \l__enumext_standar_first_bool
1756     {
1757         \tl_gclear:N \g__enumext_standar_series_tl
1758         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1759     }
1760     \bool_if:NT \l__enumext_starred_first_bool
1761     {
1762         \tl_gclear:N \g__enumext_starred_series_tl
1763         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1764     }
1765 }

```

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

13.24.2 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_⟨series name⟩_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 (§13.38) and the *enumext** environment definition (§13.43).

```

1766 \cs_new_protected:Npn \__enumext_resume_save_counter:
1767 {
1768     \bool_if:NT \g__enumext_standar_bool
1769     {
1770         \tl_if_empty:NF \l__enumext_series_str
1771         {
1772             \int_gset_eq:cN
1773             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1774         }
1775         \tl_if_empty:NTF \l__enumext_resume_name_tl
1776         {
1777             \str_if_empty:NT \l__enumext_series_str
1778             {
1779                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1780             }
1781         }
1782         {
1783             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1784             {
1785                 \int_gset_eq:cN
1786                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1787             }
1788         }
1789         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1790         {
1791             \int_gset_eq:cN
1792             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1793         }
1794     }
1795     \bool_if:NT \g__enumext_starred_bool
1796     {
1797         \tl_if_empty:NF \l__enumext_series_str
1798         {
1799             \int_gset_eq:cN
1800             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}

```

```

1801     }
1802     \tl_if_empty:NTF \l__enumext_resume_name_tl
1803     {
1804         \str_if_empty:NT \l__enumext_series_str
1805         {
1806             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1807         }
1808     }
1809     {
1810         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1811         {
1812             \int_gset_eq:cN
1813             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1814         }
1815     }
1816     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1817     {
1818         \int_gset_eq:cN
1819         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1820     }
1821 }
1822 }

```

(End of definition for __enumext_resume_save_counter:.)

13.24.3 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.

```

1823 \cs_new_protected:Npn \__enumext_resume_series:n #1
1824 {
1825     \tl_if_empty:NTF {#1}
1826     {
1827         \__enumext_resume_counter:n { }
1828     }
1829     {
1830         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1831         {
1832             \__enumext_resume_counter:n {#1}
1833             \bool_if:NT \g__enumext_standar_bool
1834             {
1835                 \keys_set:nv { enumext / level-1 }
1836                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1837             }
1838             \bool_if:NT \g__enumext_starred_bool
1839             {
1840                 \keys_set:nv { enumext / enumext* }
1841                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1842             }
1843         }
1844         {
1845             \bool_if:NT \g__enumext_standar_bool
1846             {
1847                 \msg_error:nnn { enumext } { unknown-series } {#1}
1848             }
1849             \bool_if:NT \g__enumext_starred_bool
1850             {
1851                 \msg_error:nnn { enumext } { unknown-series } {#1}
1852             }
1853         }
1854     }
1855 }

```

(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 $\{\langle series\ name\rangle\}$. 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= $\{\langle series\ name\rangle\}$* 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*.

```

1856 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1857 {
1858   \bool_set_true:N \l__enumext_resume_active_bool
1859   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1860   \tl_if_empty:NTF \l__enumext_resume_name_tl
1861   {
1862     \__enumext_resume_counter:
1863   }
1864   {
1865     \__enumext_resume_counter_series:
1866   }
1867   \__enumext_resume_counter_save_ans:
1868 }

```

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.

```

1869 \cs_new_protected:Nn \__enumext_resume_counter:
1870 {
1871   \bool_if:NT \g__enumext_standar_bool
1872   {
1873     \int_gincr:N \g__enumext_resume_int
1874     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1875   }
1876   \bool_if:NT \g__enumext_starred_bool
1877   {
1878     \int_gincr:N \g__enumext_resume_vii_int
1879     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1880   }
1881 }

```

The function `__enumext_resume_counter_series:` will be executed when the *resume= $\{\langle series\ name\rangle\}$* 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.

```

1882 \cs_new_protected:Nn \__enumext_resume_counter_series:
1883 {
1884   \bool_if:NT \g__enumext_standar_bool
1885   {
1886     \int_set:Nn \l__enumext_start_i_int
1887     {
1888       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1889     }
1890   }
1891   \bool_if:NT \g__enumext_starred_bool
1892   {
1893     \int_set:Nn \l__enumext_start_vii_int
1894     {
1895       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1896     }
1897   }
1898 }

```

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.

```

1899 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1900 {
1901   \bool_lazy_and:nnT
1902   { \bool_if_p:N \l__enumext_standar_first_bool }
1903   { \bool_if_p:N \l__enumext_store_active_bool }
1904   {
1905     \int_set:Nn \l__enumext_start_i_int
1906     {
1907       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1908     }
1909   }
1910   \bool_lazy_and:nnT

```

```

1911     { \bool_if_p:N \__enumext_starred_first_bool }
1912     { \bool_if_p:N \__enumext_store_active_bool }
1913     {
1914         \int_set:Nn \__enumext_start_vii_int
1915         {
1916             \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1917         }
1918     }
1919 }

```

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

13.24.4 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.

```

1920 \cs_new_protected:Nn \__enumext_resume_starred:
1921 {
1922     \bool_if:NT \g__enumext_standar_bool
1923     {
1924         \tl_if_empty:NF \g__enumext_standar_series_tl
1925         {
1926             \__enumext_resume_counter:n { }
1927             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1928         }
1929     }
1930     \bool_if:NT \g__enumext_starred_bool
1931     {
1932         \tl_if_empty:NF \g__enumext_starred_series_tl
1933         {
1934             \__enumext_resume_counter:n { }
1935             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1936         }
1937     }
1938 }

```

(End of definition for `__enumext_resume_starred:`.)

13.25 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.

13.25.1 Setting save-ans key

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

```

1939 \cs_set_protected:Npn \__enumext_tmp:n #1
1940 {
1941     \keys_define:nn { enumext / #1 }
1942     {
1943         save-ans .code:n = \__enumext_storing_set:n {##1},
1944         save-ans .value_required:n = true,
1945     }
1946 }
1947 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

13.25.2 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:`.

```

1948 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1949 {
1950     \msg_term:nnVV { enumext } { save-ans-log }
1951     \g__enumext_envir_name_tl \__enumext_store_name_tl
1952 }
1953 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1954 {

```



```

1955     \msg_term:nnVV { enumext } { save-ans-log-hook }
1956     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1957 }

```

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

```

\__enumext_storing_set:n
\__enumext_storing_exec:

```

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

```

1958 \cs_new_protected:Npn \__enumext_storing_set:n #1
1959 {
1960     \tl_set:Nx \l__enumext_store_name_tl {#1}
1961     \tl_if_empty:NTF \l__enumext_store_name_tl
1962     {
1963         \bool_lazy_or:nnT
1964         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1965         {
1966             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1967         }
1968     }
1969     {
1970         \bool_lazy_or:nnT
1971         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1972         {
1973             \__enumext_start_save_ans_msg:
1974             \__enumext_storing_exec:
1975         }
1976     }
1977 }

```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_bool which activates the use of the `\anskey` command and the `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to “true” the variable \l__enumext_check_answers_bool used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, copy $\langle store\ name \rangle$ into the variable \g__enumext_store_name_tl and execute the function __enumext_anskey_env_make:V creating the environment `anskey*` (§13.30).

```

1978 \cs_new_protected:Nn \__enumext_storing_exec:
1979 {
1980     \bool_set_true:N \l__enumext_store_active_bool
1981     \bool_set_true:N \l__enumext_check_answers_bool
1982     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1983     \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

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

```

1984     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1985     {
1986         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1987         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1988     }
1989     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1990     {
1991         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1992         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1993     }
1994     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1995     {
1996         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1997         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1998     }
1999 }

```

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

13.25.3 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.

13.25.4 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 2000 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 2001 {
2002     \keys_define:nn { enumext / #1 }
2003     {
2004         check-ans .bool_set:N = \__enumext_check_ans_key_bool,
2005         check-ans .initial:n = false,
2006         check-ans .value_required:n = true,
2007         no-store .code:n = {
2008             \bool_set_false:N \__enumext_check_answers_bool
2009             \bool_set_false:N \__enumext_check_ans_key_bool
2010         },
2011         no-store .value_forbidden:n = true,
2012     }
2013 }
2014 \clist_map_inline:nn
2015 {
2016     level-1, level-2, level-3, level-4, enumext*
2017 }
2018 { \__enumext_tmp:n {#1} }
```

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

13.25.5 Set-up check answer mechanism

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

```

2019 \cs_new_protected:Nn \__enumext_check_ans_active:
2020 {
2021     \tl_if_empty:NF \l__enumext_store_name_tl
2022     {
2023         \bool_if:NT \__enumext_check_answers_bool
2024         {
2025             \__enumext_check_ans_level:
2026         }
2027     }
2028 }
```

The function `__enumext_check_ans_level:` will decrement by “one” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite and set `\l__enumext_item_number_bool` to “false”.

```

2029 \cs_new_protected:Nn \__enumext_check_ans_level:
2030 {
2031   \int_case:nn { \l__enumext_level_int }
2032   {
2033     { 1 }{
2034       \bool_lazy_all:nT
2035       {
2036         { \bool_if_p:N \g__enumext_starred_bool }
2037         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2038       }
2039       {
2040         \int_gdecr:N \g__enumext_item_number_int
2041         \bool_set_false:N \l__enumext_item_number_bool
2042       }
2043     }
2044     { 2 }{
2045       \int_gdecr:N \g__enumext_item_number_int
2046       \bool_set_false:N \l__enumext_item_number_bool
2047     }
2048     { 3 }{
2049       \int_gdecr:N \g__enumext_item_number_int
2050       \bool_set_false:N \l__enumext_item_number_bool
2051     }
2052     { 4 }{
2053       \int_gdecr:N \g__enumext_item_number_int
2054       \bool_set_false:N \l__enumext_item_number_bool
2055     }
2056   }

```

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.

```

2057   \int_case:nn { \l__enumext_level_h_int }
2058   {
2059     { 1 }{
2060       \bool_lazy_all:nT
2061       {
2062         { \bool_if_p:N \g__enumext_standar_bool }
2063         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2064       }
2065       {
2066         \int_gdecr:N \g__enumext_item_number_int
2067         \bool_set_false:N \l__enumext_item_number_bool
2068       }
2069     }
2070   }
2071 }

```

(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.

```

2072 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2073 {
2074   \bool_lazy_and:nnT
2075   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2076   { \bool_if_p:N \g__enumext_standar_bool }
2077   {
2078     \bool_gset_true:N \g__enumext_check_ans_key_bool
2079   }
2080   \bool_lazy_and:nnT
2081   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2082   { \bool_if_p:N \g__enumext_starred_bool }
2083   {
2084     \bool_gset_true:N \g__enumext_check_ans_key_bool
2085   }
2086 }

```

(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:`.

```
2087 \cs_new_protected:Nn \__enumext_item_answer_diff:
2088 {
2089   \int_gset:Nn \g__enumext_item_answer_diff_int
2090   {
2091     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2092   }
2093 }
```

(End of definition for `__enumext_item_answer_diff:`.)

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

```
2094 \cs_new_protected:Nn \__enumext_check_ans_show:
2095 {
2096   \int_case:nn { \g__enumext_item_answer_diff_int }
2097   {
2098     { -1 } { \__enumext_check_ans_msg_less: }
2099     { 0 } { \__enumext_check_ans_msg_same_ok: }
2100     { 1 } { \__enumext_check_ans_msg_greater: }
2101   }
2102 }
2103 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2104 {
2105   \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2106   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2107 }
2108 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2109 {
2110   \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2111   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2112 }
2113 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2114 {
2115   \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2116   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2117 }
```

(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:`.

```
2118 \cs_new_protected:Nn \__enumext_check_ans_log:
2119 {
2120   \int_case:nn { \g__enumext_item_answer_diff_int }
2121   {
2122     { -1 } { \__enumext_check_ans_log_msg_less: }
2123     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2124     { 1 } { \__enumext_check_ans_log_msg_greater: }
2125   }
2126 }
2127 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2128 {
2129   \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2130   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2131 }
2132 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2133 {
2134   \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2135   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2136 }
```

```

2137 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2138 {
2139     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2140     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2141 }

```

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

13.25.6 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.

```

2142 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2143 {
2144     \int_compare:nNnT
2145     { \g__enumext_check_starred_cmd_int } = { 0 }
2146     {
2147         \msg_warning:nnnV
2148         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2149     }
2150     \int_compare:nNnT
2151     { \g__enumext_check_starred_cmd_int } > { 1 }
2152     {
2153         \msg_warning:nnnV
2154         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2155     }
2156     \int_gzero:N \g__enumext_check_starred_cmd_int
2157     \tl_clear:N \l__enumext_check_start_line_env_tl
2158 }

```

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

13.26 Keys and functions associated with storage

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

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

2159 \cs_set_protected:Npn \__enumext_tmp:n #1
2160 {
2161     \keys_define:nn { enumext / #1 }
2162     {
2163         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2164         wrap-ans .initial:n =
2165             {
2166                 \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2167             },
2168         wrap-ans .value_required:n = true,
2169         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2170         wrap-opt .initial:n = [{##1}],
2171         wrap-opt .value_required:n = true,
2172         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2173         save-sep .initial:n = {, ~ },
2174         save-sep .value_required:n = true,
2175         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2176         mark-ans .initial:n = \textasteriskcentered,
2177         mark-ans .value_required:n = true,
2178         mark-pos .choice:,
2179         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2180         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2181         mark-pos / unknown .code:n =
2182             \msg_error:nneee { enumext } { unknown-choice }
2183             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2184         mark-pos .initial:n = right,
2185         mark-pos .value_required:n = true,
2186         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2187         show-ans .initial:n = false,
2188         show-ans .value_required:n = true,
2189         show-pos .bool_set:N = \l__enumext_show_position_bool,
2190         show-pos .initial:n = false,
2191         show-pos .value_required:n = true,
2192         mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,

```

```

2193     mark-ref .initial:n = \textasteriskcentered,
2194     mark-ref .value_required:n = true,
2195     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2196     save-ref .initial:n = false,
2197     save-ref .value_required:n = true,
2198   }
2199 }
2200 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

(End of definition for wrap-ans and others.)

mark-pos For the **keyans** and **keyans*** environments we will only add the keys mark-pos, show-ans and show-pos.

```

show-ans 2201 \cs_set_protected:Npn \l__enumext_tmp:n #1
show-pos 2202 {
2203   \keys_define:nn { enumext / #1 }
2204   {
2205     mark-pos .choice:,
2206     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2207     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2208     mark-pos .initial:n = right,
2209     mark-pos .value_required:n = true,
2210     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2211     show-ans .initial:n = false,
2212     show-ans .value_required:n = true,
2213     show-pos .bool_set:N = \l__enumext_show_position_bool,
2214     show-pos .initial:n = false,
2215     show-pos .value_required:n = true,
2216   }
2217 }
2218 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

(End of definition for mark-pos, show-ans, and show-pos.)

13.26.1 Store optional arguments of the environments

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

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

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

```

2219 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2220 {
2221   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2222   {
2223     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2224     \tl_set:ce
2225       { l__enumext_store_save_key_ \__enumext_level: _tl }
2226       { \__enumext_filter_save_key:n {#1} }
2227   }
2228 }
2229 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2230 {
2231   \bool_if:NF \l__enumext_store_save_key_vii_bool
2232   {
2233     \tl_clear:N \l__enumext_store_save_key_vii_tl
2234     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2235   }
2236 }

```

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

13.26.2 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.

```

2237 \cs_set_protected:Npn \__enumext_tmp:n #1
2238 {
2239   \keys_define:nn { enumext / enumext* }
2240   {
2241     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2242     save-key .value_required:n = true,
2243   }
2244   \keys_define:nn { enumext / #1 }
2245   {
2246     save-key .code:n = \__enumext_parse_save_key:n {##1},
2247     save-key .value_required:n = true,
2248   }
2249 }
2250 \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*`.

```

2251 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2252 {
2253   \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2254   \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2255   \tl_set:ce
2256   { l__enumext_store_save_key_ \__enumext_level: _tl }
2257   { \__enumext_filter_save_key:n {#1} }
2258 }
2259 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2260 {
2261   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2262   \tl_clear:N \l__enumext_store_save_key_vii_tl
2263   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2264 }

```

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

13.26.3 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.

```

2265 \cs_new:Npn \__enumext_filter_save_key:n #1
2266 {
2267   \use:e
2268   {
2269     \keyval_parse:NNn
2270     \__enumext_filter_save_key_key:n
2271     \__enumext_filter_save_key_pair:nn {#1}
2272   }
2273 }

```

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.

```

2274 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2275 {
2276   \str_case:nnF {#1}
2277   {
2278     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2279   }
2280   { , { \exp_not:n {#1} } }
2281 }

```


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

```

2282 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2283 {
2284   \str_case:nnF {#1}
2285   {
2286     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2287     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2288     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2289     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2290     { mini-right* } {}
2291   }
2292   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2293 }

```

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

13.26.4 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2294 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2295 {
2296   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2297   {
2298     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2299   }
2300   { #1 }
2301 }
2302 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

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

13.26.5 Function for storing content in sequence

```

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

```

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

The form in which the `{\content}` 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.

```

2303 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2304 {
2305   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2306 }
2307 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }

```

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

13.26.6 Functions for storing structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

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

```

2308 \cs_new_protected:Nn \__enumext_store_level_open:
2309 {
2310   \bool_if:NT \__enumext_check_answers_bool
2311   {
2312     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2313     {
2314       \__enumext_store_addto_seq:n
2315       {
2316         \item \begin{enumext}
2317       }
2318     }
2319     {
2320       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2321       {

```

```

2322         \item \begin{enumext} [
2323         ]
2324         \tl_put_right:cn { l__enumext_store_save_key_ \_tl }
2325         {
2326         ]
2327         }
2328         \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \_tl }
2329     }
2330 }
2331 }
2332 \cs_new_protected:Nn \__enumext_store_level_close:
2333 {
2334     \bool_if:NT \l__enumext_check_answers_bool
2335     {
2336         \__enumext_store_addto_seq:n { \end{enumext} }
2337     }
2338 }

```

(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.

```

2339 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2340 {
2341     \bool_if:NT \l__enumext_check_answers_bool
2342     {
2343         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2344         {
2345             \__enumext_store_addto_seq:n
2346             {
2347                 \item \begin{enumext*}
2348             }
2349         }
2350         {
2351             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2352             {
2353                 \item \begin{enumext*}[
2354             }
2355             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2356             {
2357                 ]
2358             }
2359             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2360         }
2361     }
2362 }
2363 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2364 {
2365     \bool_if:NT \l__enumext_check_answers_bool
2366     {
2367         \__enumext_store_addto_seq:n { \end{enumext*} }
2368     }
2369 }

```

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

13.26.7 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

```

2370 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2371 {
2372     \mode_leave_vertical:
2373     \skip_horizontal:n { -\dim_use:N #2 }
2374     \makebox[0pt][ r ]
2375     {
2376         \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2377         {
2378             \tl_use:N \l__enumext_mark_answer_sym_tl

```

```

2379     }
2380   }
2381   \skip_horizontal:n { \dim_use:N #2 }
2382 }
2383 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

13.27 The internal label and ref

The function __enumext_store_internal_ref: handles the “*internal label and ref*” system used by the `save-ref` and `mark-ref` keys for `\anskey` will allow to execute `\ref{⟨store name : position⟩}` and will return `1.(a).i.A.`

__enumext_store_internal_ref:

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

```

2384 \cs_new_protected:Nn \__enumext_store_internal_ref:
2385 {
2386   \cs_set_protected:Npn \__enumext_tmp:n ##1
2387   {
2388     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2389     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2390     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2391     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2392   }
2393   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2394   \cs_set:Npn \__enumext_tmp:n ##1
2395   { . \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.

```

2396   \bool_lazy_all:nT
2397   {
2398     { \bool_if_p:N \g__enumext_starred_bool }
2399     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2400   }
2401   {
2402     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2403     { \tl_use:N \l__enumext_label_copy_vii_tl }
2404   }
2405   \bool_lazy_all:nT
2406   {
2407     { \bool_not_p:n { \g__enumext_standar_bool } }
2408     { \bool_if_p:N \l__enumext_standar_bool }
2409     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2410   }
2411   {
2412     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2413     {
2414       \tl_use:N \l__enumext_label_copy_vii_tl
2415       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2416     }
2417   }

```

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.

```

2418   \bool_lazy_all:nT
2419   {
2420     { \bool_if_p:N \g__enumext_standar_bool }
2421     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2422     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2423   }
2424   {
2425     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2426     {
2427       \tl_use:N \l__enumext_label_copy_i_tl
2428       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2429     }
2430   }
2431   \cs_set:Npn \__enumext_tmp:n ##1
2432   { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2433   \bool_lazy_all:nT

```

```

2434     {
2435       { \bool_if_p:N \g__enumext_standar_bool }
2436       { \bool_if_p:N \l__enumext_starred_bool }
2437       { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2438     }
2439     {
2440       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2441       {
2442         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2443         \tl_use:N \l__enumext_label_copy_vii_tl
2444       }
2445     }

```

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

```

2446     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2447     {
2448       \l__enumext_store_name_tl \c_colon_str
2449       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2450     }

```

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.

```

2451     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2452     {
2453       \__enumext_newlabel:nn
2454       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2455       { \l__enumext_newlabel_arg_two_tl }
2456     }
2457     \l__enumext_write_aux_file_tl
2458   }

```

(End of definition for `__enumext_store_internal_ref:`)

13.28 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*” $\langle \textit{argument} \rangle$.

```

2459 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2460 {
2461   \int_gincr:N \g__enumext_item_anskey_int
2462   \__enumext_store_addto_prop:n {#1}
2463   \bool_if:NT \l__enumext_store_ref_key_bool
2464   {
2465     \__enumext_store_internal_ref:
2466   }
2467   \__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`.

```

2468   \tl_clear:N \l__enumext_store_anskey_arg_tl
2469   \bool_lazy_and:nnT
2470   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2471   { \bool_not_p:n { \l__enumext_starred_bool } }
2472   {
2473     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2474   }
2475   \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`.

```

2476   \bool_lazy_and:nnT
2477   { \bool_not_p:n { \l__enumext_starred_bool } }
2478   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2479   {
2480     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2481     {
2482       ( \exp_not:V \l__enumext_store_item_join_int )
2483     }
2484   }

```

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*`.

```

2485 \bool_if:NTF \l__enumext_store_item_star_bool
2486 {
2487   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2488   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2489   {
2490     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2491     {
2492       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2493     }
2494   }
2495   \dim_compare:nT
2496   {
2497     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2498   }
2499   {
2500     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2501     {
2502       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2503     }
2504   }
2505   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2506 }
2507 {
2508   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2509 }

```

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*.

```

2510 \bool_lazy_and:nnT
2511 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2512 { \bool_if_p:N \l__enumext_hyperref_bool }
2513 {
2514   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2515   {
2516     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2517     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2518   }
2519 }
2520 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2521 }

```

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

`\l__enumext_anskey_show_wrap_arg:n`

The function `\l__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` key.

```

2522 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_arg:n #1
2523 {
2524   \par
2525   \bool_if:NTF \l__enumext_starred_bool
2526   {
2527     \l__enumext_print_keyans_box:NN
2528     \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2529   }
2530   {
2531     \l__enumext_print_keyans_box:cc
2532     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2533     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2534   }
2535   \l__enumext_anskey_wrapper:n { #1 }
2536 }

```

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

`\l__enumext_anskey_show_wrap_left:n`

The function `\l__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.

```

2537 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_left:n #1
2538 {

```

```

2539 \bool_if:NT \l__enumext_show_answer_bool
2540 {
2541   \__enumext_anskey_show_wrap_arg:n { #1 }
2542 }
2543 \bool_if:NT \l__enumext_show_position_bool
2544 {
2545   \tl_set:Nx \l__enumext_mark_answer_sym_tl
2546   {
2547     \group_begin:
2548     \exp_not:N \normalfont
2549     \exp_not:N \footnotesize [ \int_eval:n
2550       {
2551         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2552       }
2553     ]
2554     \group_end:
2555   }
2556   \__enumext_anskey_show_wrap_arg:n { #1 }
2557 }
2558 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

13.29 The command \anskey

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

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

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

```

\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:nn
2559 \keys_define:nn { enumext / anskey }
2560 {
2561   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2562   break-col .default:n = true,
2563   break-col .value_forbidden:n = true,
2564   item-join .int_set:N = \l__enumext_store_item_join_int,
2565   item-join .value_required:n = true,
2566   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2567   item-star .default:n = true,
2568   item-star .value_forbidden:n = true,
2569   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2570   item-sym* .value_required:n = true,
2571   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2572   item-pos* .value_required:n = true,
2573   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2574 }

```

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

```

2575 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2576 {
2577   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2578 }
2579 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2580 {
2581   \tl_if_blank:nTF {#2}
2582   {
2583     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2584   }
2585   {
2586     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2587   }
2588 }

```

(End of definition for __enumext_anskey_unknown:n and __enumext_anskey_unknown:nn.)

- 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`.

```

2589 \NewDocumentCommand \anskey { o +m }
2590 {
2591   \__enumext_anskey_safe_outer:
2592   \group_begin:
2593     \bool_if:NT \l__enumext_check_answers_bool
2594     {
2595       \tl_if_novalue:nF {#1}
2596       {
2597         \keys_set:nn { enumext / anskey } {#1}
2598       }
2599       \tl_if_blank:nTF {#2}
2600       {
2601         \msg_error:nn { enumext } { anskey-empty-arg }
2602       }
2603       {
2604         \__enumext_anskey_safe_inner:
2605         \__enumext_store_anskey_code:n {#2}
2606       }
2607     }
2608   \group_end:
2609 }

```

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

13.29.1 Internal functions for the command

`__enumext_anskey_safe_outer:`
`__enumext_anskey_safe_inner:`

The `__enumext_store_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the `save-ans` key was activated.

```

2610 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2611 {
2612   \bool_if:NF \l__enumext_store_active_bool
2613   {
2614     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2615   }
2616   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2617   {
2618     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2619   }
2620   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2621   {
2622     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2623   }
2624   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2625   {
2626     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2627   }
2628 }

```

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.

```

2629 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2630 {
2631   \int_incr:N \l__enumext_anskey_level_int
2632   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2633   {
2634     \msg_error:nn { enumext } { anskey-nested }
2635   }
2636   \bool_if:NF \l__enumext_item_number_bool
2637   {
2638     \msg_error:nn { enumext } { anskey-unnumber-item }
2639   }
2640   \mode_if_math:T
2641   {
2642     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2643   }
2644 }

```

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

13.30 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:` (§13.31) which is executed after the environment in which the key `save-ans` is active.

```
2645 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2646 {
2647   \cs_undefine:c { anskey* }
2648   \cs_undefine:c { endanskey* }
2649   \cs_undefine:c { __scontents_anskey*_env_begin: }
2650   \cs_undefine:c { __scontents_anskey*_env_end: }
2651 }
```

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

```
2652 \__enumext_before_env:nn { enumext }
2653 {
2654   \bool_lazy_and:nnT
2655     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2656     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2657     {
2658       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2659       {
2660         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2661       }
2662     }
2663 }
2664 \__enumext_before_env:nn { enumext* }
2665 {
2666   \bool_lazy_and:nnT
2667     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2668     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2669     {
2670       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2671       {
2672         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2673       }
2674     }
2675 }
```

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.

```
2676 \__enumext_before_env:nn { anskey* }
2677 {
2678   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2679   {
2680     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2681   }
2682   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2683   {
2684     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2685   }
2686   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2687   {
2688     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2689   }
2690   \bool_if:NF \__enumext_item_number_bool
2691   {
```

```

2692         \msg_error:nn { enumext } { anskey-unnumber-item }
2693     }
2694     \mode_if_math:T
2695     {
2696         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2697     }
2698 }

```

(End of definition for `__enumext_undefine_anskey_env:`)

anskey*

The function `__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 `__enumext_storing_exec:` (§13.25.1) 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**.

```

2699 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2700 {
2701     \bool_if:NT \l__enumext_anskey_env_bool
2702     {
2703         \newenvsc{anskey*}[store-env=#1,print-env=false]
2704         \__enumext_anskey_env_exec:
2705     }
2706 }
2707 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }

```

The function `__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 `__enumext_before_env:nn`.

```

2708 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2709 {
2710     \keys_define:nn { scontents / scontents }
2711     {
2712         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2713         break-col .default:n   = true,
2714         break-col .value_forbidden:n = true,
2715         item-join .int_gset:N   = \g__enumext_store_item_join_int,
2716         item-join .value_required:n = true,
2717         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2718         item-star .default:n   = true,
2719         item-star .value_forbidden:n = true,
2720         item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2721         item-sym* .value_required:n = true,
2722         item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2723         item-pos* .value_required:n = true,
2724         print-env .undefine:,
2725         store-env .undefine:,
2726         write-out .undefine:,
2727         unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2728     }
2729 }

```

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

```

2730 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2731 {
2732     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2733 }
2734 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2735 {
2736     \tl_if_blank:nTF {#2}
2737     {
2738         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2739     }
2740     {
2741         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2742     }
2743 }

```

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`.

```

2744 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2745 {
2746   \keys_define:nn { scontents / scontents }
2747   {
2748     break-col .undefine:,
2749     item-join .undefine:,
2750     item-star .undefine:,
2751     item-sym* .undefine:,
2752     item-pos* .undefine:,
2753     write-out .code:n = {
2754       \bool_set_false:N \l__scontents_storing_bool
2755       \bool_set_true:N \l__scontents_writing_bool
2756       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2757     },
2758     write-out .value_required:n = true,
2759     print-env .meta:nn = { scontents } { print-env = ##1 },
2760     print-env .default:n = true,
2761     store-env .meta:nn = { scontents } { store-env = ##1 },
2762     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2763   }
2764 }

```

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*.

```

2765 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2766 {
2767   \group_begin:
2768   \int_set:Nn \tex_newlinechar:D { `^^J }
2769   \__scontents_rescan_tokens:x
2770   {
2771     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2772     #1
2773   }
2774 }

```

(End of definition for *anskey** and others. This function is documented on page 13.)

`__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)*.

```

2775 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2776 {
2777   \__enumext_before_env:nn { anskey* }
2778   {
2779     \__enumext_anskey_env_define_keys:
2780   }

```

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.

```

2781   \hook_if_empty:nF {env/anskey*/after}
2782   {
2783     \hook_gremove_code:nn {env/anskey*/after} { * }
2784   }
2785   \__enumext_after_env:nn { anskey* }
2786   {
2787     \__enumext_anskey_env_save_keys:
2788     \tl_clear:N \l__enumext_store_anskey_env_tl
2789     \tl_clear:N \l__enumext_store_anskey_opt_tl
2790     \bool_if:NT \l__enumext_check_answers_bool
2791     {
2792       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2793       {
2794         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2795       }
2796       \regex_match:nVTF
2797       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2798       \l__enumext_store_anskey_env_tl

```

```

2799         {
2800             \msg_error:nn { enumext } { anskey-empty-arg }
2801         }
2802         {
2803             \__enumext_anskey_env_store:
2804         }
2805     }
2806     \__enumext_anskey_env_clean_vars:
2807     \__enumext_anskey_env_reset_keys:
2808 }
2809 }

```

• 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 `\l__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`.

```

2810 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2811 {
2812     \bool_lazy_and:nnT
2813     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2814     { \bool_not_p:n { \l__enumext_starred_bool } }
2815     {
2816         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2817     }

```

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

```

2818     \bool_lazy_and:nnT
2819     { \bool_not_p:n { \l__enumext_starred_bool } }
2820     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2821     {
2822         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2823         {
2824             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2825         }
2826     }

```

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

```

2827     \bool_if:NT \g__enumext_store_item_star_bool
2828     {
2829         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2830         {
2831             ,item-star,
2832         }
2833         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2834         {
2835             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2836             {
2837                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2838             }
2839         }
2840         \dim_compare:nT
2841         {
2842             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2843         }
2844         {
2845             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2846             {
2847                 ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2848             }
2849         }
2850     }
2851 }

```

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`.

```

2852 \cs_new_protected:Nn \__enumext_anskey_env_store:

```

```

2853 {
2854   \group_begin:
2855   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2856   {
2857     \exp_args:Ne
2858     \__enumext_store_anskey_code:n
2859     {
2860       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2861     }
2862   }
2863   {
2864     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2865     \exp_args:Ne
2866     \__enumext_store_anskey_code:n
2867     {
2868       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2869     }
2870   }
2871   \group_end:
2872 }

```

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

```

2873 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2874 {
2875   \bool_gset_false:N \g__enumext_store_columns_break_bool
2876   \int_gzero:N \g__enumext_store_item_join_int
2877   \bool_gset_false:N \g__enumext_store_item_star_bool
2878   \tl_gclear:N \g__enumext_store_item_symbol_tl
2879   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2880 }

```

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

13.31 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*` (§13.30) 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.

```

2881 \cs_new_protected:Nn \__enumext_execute_after_env:
2882 {
2883   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
2884   {
2885     \tl_if_empty:NF \g__enumext_store_name_tl
2886     {
2887       \__enumext_stop_save_ans_msg:
2888       \__enumext_item_answer_diff:
2889       \__enumext_log_global_vars:
2890       \__enumext_log_answer_vars:
2891       \bool_if:NTF \g__enumext_check_ans_key_bool
2892       {
2893         \__enumext_check_ans_show:
2894       }
2895       { \__enumext_check_ans_log: }
2896       \__enumext_undefine_anskey_env:
2897     }
2898     \__enumext_reset_global_vars:
2899   }
2900 }

```

(End of definition for `__enumext_execute_after_env:`.)

- This function is passed to the function `__enumext_after_env:nn` for the environments `enumext` (§13.38) and `enumext*` (§13.43) and it is executed only when the environments are not nested or at some level of these..

13.32 Common functions for keyans, keyans* and keyanspic

13.32.1 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 `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `__enumext_store_addto_prop:V`.

```

2901 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2902 {
2903   \tl_clear:N \l__enumext_store_current_label_tl
2904   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2905   {
2906     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2907   }
2908   {
2909     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2910   }

```

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

```

2911   \tl_if_novalue:nF { #1 }
2912   {
2913     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2914     {
2915       \tl_put_right:Ne \l__enumext_store_current_label_tl
2916       {
2917         \l__enumext_store_keyans_item_opt_sep_tl
2918       }
2919     }
2920     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2921   }
2922   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2923 }

```

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

13.32.2 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 `\ref{⟨store name : position⟩}` and will return `1.` (A).

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

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

```

2924 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2925 {
2926   \bool_if:NT \l__enumext_store_ref_key_bool
2927   {
2928     \cs_set_protected:Npn \__enumext_tmp:n ##1
2929     {
2930       \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2931       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2932       \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2933       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2934     }
2935     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2936     \__enumext_keyans_store_ref_aux_i:
2937   }
2938 }

```

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`.

```

2939 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2940 {
2941   \bool_if:NT \g__enumext_starred_bool
2942   {
2943     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2944   }
2945   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

2946     {
2947       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2948       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2949     }
2950     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2951     {
2952       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2953       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2954     }
2955     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2956     {
2957       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2958       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2959     }
2960     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2961     {
2962       \l__enumext_store_name_tl \c_colon_str
2963       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2964     }
2965     \__enumext_keyans_store_ref_aux_ii:
2966   }

```

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.

```

2967 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2968 {
2969   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2970   {
2971     \__enumext_newlabel:nn
2972     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2973     { \l__enumext_newlabel_arg_two_tl }
2974   }
2975   \l__enumext_write_aux_file_tl
2976 }

```

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

13.32.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`
`__enumext_keyans_addto_seq_link:`

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

```

2977 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2978 {
2979   \tl_clear:N \l__enumext_store_current_label_tl
2980   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2981   {
2982     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2983   }
2984   {
2985     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2986   }
2987   \tl_if_no_value:nF { #1 }
2988   {
2989     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2990     {
2991       \tl_put_right:Ne \l__enumext_store_current_label_tl
2992       {
2993         \l__enumext_store_keyans_item_opt_sep_tl
2994       }
2995     }
2996     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2997   }
2998   \__enumext_keyans_addto_seq_link:
2999 }

```

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.

```

3000 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3001 {
3002   \bool_lazy_and:nnT
3003     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3004     { \bool_if_p:N \l__enumext_hyperref_bool }
3005   {
3006     \tl_put_right:Ne \l__enumext_store_current_label_tl
3007     {
3008       \hfill \exp_not:N \hyperlink
3009       {
3010         \exp_not:V \l__enumext_newlabel_arg_one_tl
3011       }
3012       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3013     }
3014   }
3015   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3016   \bool_if:NT \l__enumext_check_answers_bool
3017   {
3018     \int_gincr:N \g__enumext_item_anskey_int
3019   }
3020 }

```

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

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

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `(label)` are incorrect.

```

\__enumext_keyans_show_left:n
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:
\__enumext_keyans_show_item_opt:

```

Common function to show *starred commands* `\item*` and `(position)` of stored content in *prop list* for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_⟨store name⟩_prop` for show-pos key.

```

3021 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3022 {
3023   \tl_if_novalue:nF { #1 }
3024   {
3025     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3026   }
3027   \bool_if:NT \l__enumext_show_answer_bool
3028   {
3029     \__enumext_keyans_show_ans:
3030   }
3031   \bool_if:NT \l__enumext_show_position_bool
3032   {
3033     \__enumext_keyans_show_pos:
3034   }
3035 }
3036 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3037 {
3038   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3039   {
3040     \bool_lazy_or:nnT
3041       { \bool_if_p:N \l__enumext_show_answer_bool }
3042       { \bool_if_p:N \l__enumext_show_position_bool }
3043     {
3044       \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3045     }
3046   }
3047 }
3048 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3049 {
3050   \bool_if:NT \l__enumext_starred_bool
3051   {
3052     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3053     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3054   }
3055   \tl_put_left:Nn \l__enumext_label_v_tl
3056   {
3057     \__enumext_print_keyans_box:NN
3058     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3059   }

```

```

3060 }
3061 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3062 {
3063   \bool_if:NT \l__enumext_starred_bool
3064   {
3065     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3066     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3067   }
3068   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3069   {
3070     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3071     {
3072       \group_begin:
3073       \exp_not:N \normalfont
3074       \exp_not:N \footnotesize [ \int_eval:n
3075         {
3076           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3077         }
3078       ]
3079       \group_end:
3080     }
3081   }
3082   {
3083     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3084     {
3085       \group_begin:
3086       \exp_not:N \normalfont
3087       \exp_not:N \footnotesize [ \int_eval:n
3088         {
3089           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3090         }
3091       ]
3092       \group_end:
3093     }
3094   }
3095   \tl_put_left:Nn \l__enumext_label_v_tl
3096   {
3097     \__enumext_print_keyans_box:NN
3098     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3099   }
3100 }

```

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

13.33 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.

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`.

```

3101 \cs_new_protected:Npn \__enumext_default_item:n #1
3102 {
3103   \tl_if_novalue:nTF {#1}
3104   {
3105     \bool_if:NT \l__enumext_check_answers_bool
3106     {
3107       \int_gincr:N \g__enumext_item_number_int
3108       \bool_set_true:N \l__enumext_item_number_bool
3109     }
3110     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3111     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3112   }
3113   {

```

```

3114         \bool_set_eq:cc
3115         { \l__enumext_wrap_label_ \l__enumext_level: _bool }
3116         { \l__enumext_wrap_label_opt_ \l__enumext_level: _bool }
3117         \l__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \l__enumext_level: _tl
3118     }
3119 }

```

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

`\l__enumext_starred_item:nn`
`\l__enumext_item_star_exec:`

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

#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 `\l__enumext_item_std:w` and the key `itemindent`.

```

3120 \cs_new_protected:Npn \l__enumext_starred_item:nn #1 #2
3121 {
3122     \tl_if_novalue:nTF {#1}
3123     {
3124         \tl_gset_eq:Nc
3125         \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \l__enumext_level: _tl }
3126     }
3127     {
3128         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3129     }
3130     \tl_if_novalue:nTF {#2}
3131     {
3132         \dim_set_eq:cc
3133         { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim }
3134         { \l__enumext_labelsep_ \l__enumext_level: _dim }
3135     }
3136     {
3137         \dim_set:cn { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } {#2}
3138     }
3139     \bool_if:NT \l__enumext_check_answers_bool
3140     {
3141         \int_gincr:N \g__enumext_item_number_int
3142         \bool_set_true:N \l__enumext_item_number_bool
3143     }
3144     \bool_set_true:c { \l__enumext_wrap_label_ \l__enumext_level: _bool }
3145     \l__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \l__enumext_level: _tl }
3146 }

```

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

```

3147 \cs_new_protected:Nn \l__enumext_item_star_exec:
3148 {
3149     \tl_if_empty:cF { \l__enumext_item_symbol_ \l__enumext_level: _tl }
3150     {
3151         \mode_leave_vertical:
3152         \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
3153         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3154         \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \l__enumext_level: _dim } }
3155     }
3156 }

```

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

`\l__enumext_redefine_item:`

The function `\l__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `\l__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.38).

```

3157 \cs_new_protected:Nn \l__enumext_redefine_item:
3158 {
3159     \RenewDocumentCommand \item { s o o }
3160     {

```

```

3161         \bool_if:nTF {##1}
3162         {
3163             \__enumext_starred_item:nn {##2} {##3}
3164         }
3165         { \__enumext_default_item:n {##2} }
3166     }
3167 }

```

(End of definition for `__enumext_redefine_item:`)

- When *tagged* PDF is active `\makelabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is by using `\makebox`. The solution here is to redefine `\makelabel` conditionally using `\IfDocumentMetadataTF`.

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makelabel` for the keys `align`, `font`, `wrap-label`, `__enumext_make_label_std:` `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `__enumext_list_arg-` `__enumext_make_label_box:` `two_X:` used in the definition of the `enumext` environment (§13.38).

```

3168 \cs_new_protected:Nn \__enumext_make_label:
3169 {
3170     \IfDocumentMetadataTF
3171     {
3172         \__enumext_make_label_box:
3173     }
3174     { \__enumext_make_label_std: }
3175 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3176 \cs_new_protected:Nn \__enumext_make_label_std:
3177 {
3178     \RenewDocumentCommand \makelabel { m }
3179     {
3180         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3181         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3182         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3183         {
3184             \__enumext_item_star_exec:
3185             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3186         }
3187         { ##1 }
3188         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3189         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3190     }
3191 }

```

Definition using `\makebox` when `\DocumentMetadata` is active.

```

3192 \cs_new_protected:Nn \__enumext_make_label_box:
3193 {
3194     \RenewDocumentCommand \makelabel { m }
3195     {
3196         \makebox
3197         [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3198         [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3199         {
3200             \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3201             \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3202             {
3203                 \__enumext_item_star_exec:
3204                 \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3205             }
3206             { ##1 }
3207             \tl_gclear:N \g__enumext_item_symbol_aux_tl
3208         }
3209     }
3210 }

```

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

13.34 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*
3211 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3212 {
3213   \keys_define:nn { enumext / #1 }
3214   {
3215     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3216     item-sym* .value_required:n = true,
3217     item-sym* .initial:n = {$\star$},
3218     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3219     item-pos* .value_required:n = true,
3220   }
3221 }
3222 \clist_map_inline:nn
3223 {
3224   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3225 }
3226 { \__enumext_tmp:nn #1 }

```

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

13.35 Handling unknown keys

At this point in the code I already know that I will not add more *⟨keys⟩* 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).

13.35.1 Handling unknown keys for keyans and keyans*

Define and set unknown key for keyans and keyans* environments.

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3227 \cs_set_protected:Npn \__enumext_tmp:n #1
3228 {
3229   \keys_define:nn { enumext / #1 }
3230   {
3231     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3232   }
3233 }
3234 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3235 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3236 {
3237   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3238 }
3239 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3240 {
3241   \tl_if_blank:nTF {#2}
3242   {
3243     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3244   }
3245   {
3246     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3247   }
3248 }

```

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

13.35.2 Handling unknown keys for enumext*

Define and set unknown key for enumext* environment.

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3249 \keys_define:nn { enumext / enumext* }
3250 {
3251   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3252 }

```

Internal functions for handling unknown key.

```

3253 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3254 {
3255   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3256 }
3257 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3258 {
3259   \tl_if_blank:nTF {#2}
3260   {
3261     \msg_error:nnn { enumext } { starred-unknown-key } {#1}

```

```

3262     }
3263     {
3264         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3265     }
3266 }

```

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

13.35.3 Handling unknown keys for enumext

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

```

unknown
\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
3267 \cs_set_protected:Npn \__enumext_tmp:n #1
3268 {
3269     \keys_define:nn { enumext / #1 }
3270     {
3271         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3272     }
3273 }
3274 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3275 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3276 {
3277     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3278 }
3279 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3280 {
3281     \tl_if_blank:nTF {#2}
3282     {
3283         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3284     }
3285     {
3286         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3287     }
3288 }

```

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

13.36 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.

The function `__enumext_keyans_default_item:n` executes the original behavior of the `\item` along with the keys `wrap-label`, `wrap-label*` and `itemindent`.

```

3289 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3290 {
3291     \tl_if_novalue:nTF { #1 }
3292     {
3293         \bool_set_true:N \__enumext_wrap_label_v_bool
3294         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3295     }
3296     {
3297         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3298         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3299     }
3300 }

```

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

The function `__enumext_keyans_starred_item:n` which will make a temporary copy of the current `⟨label⟩`, execute the `show-ans` or `show-pos` keys using the function `__enumext_keyans_show_left:n` and will display the `⟨contents⟩` of that item using the internal copy `__enumext_item_std:w`, this is necessary to prevent incrementing the current “*counter*” of the original `⟨label⟩`, followed by this it will execute function `__enumext_keyans_show_item_opt:` handled by `wrap-opt` key.

```

3301 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3302 {
3303     \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3304     \__enumext_keyans_show_left:n { #1 }
3305     \bool_set_true:N \__enumext_wrap_label_v_bool
3306     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3307     \__enumext_keyans_show_item_opt:

```

Recover the original value of the current (*label*) and store it first in the *prop list* (including the *optional argument*), run the internal “*label and ref*” system if the *save-ref* key is active, store it in the *sequence* and finally increments `\g__enumext_check_starred_cmd_int` for internal check system.

```

3308      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3309      \__enumext_keyans_addto_prop:n { #1 }
3310      \__enumext_keyans_store_ref:
3311      \__enumext_keyans_addto_seq:n { #1 }
3312      \int_gincr:N \g__enumext_check_starred_cmd_int
3313  }

```

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

`\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 need to use `\peek_remove_spaces:n` to prevent an unwanted space when using `\item*` in conjunction with the *itemindent* key. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the *keyans* environment (§13.37.2).

```

3314 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3315 {
3316   \RenewDocumentCommand \item { s o }
3317   {
3318     \bool_if:nTF {##1}
3319     {
3320       \peek_remove_spaces:n
3321       {
3322         \__enumext_keyans_starred_item:n {##2}
3323       }
3324     }
3325     {
3326       \__enumext_keyans_default_item:n {##2}
3327     }
3328   }
3329 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 14.)

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for the keys *align*, *font*, *wrap-label*, *wrap-label** and `\item*` for *keyans* environment. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the *keyans* environment (§13.37.2).

```

3330 \cs_new_protected:Nn \__enumext_keyans_make_label:
3331 {
3332   \IfDocumentMetadataTF
3333   {
3334     \__enumext_keyans_make_label_box:
3335   }
3336   { \__enumext_keyans_make_label_std: }
3337 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3338 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3339 {
3340   \RenewDocumentCommand \makeLabel { m }
3341   {
3342     \tl_use:N \l__enumext_label_fill_left_v_tl
3343     \tl_use:N \l__enumext_label_font_style_v_tl
3344     \bool_if:NTF \l__enumext_wrap_label_v_bool
3345     {
3346       \__enumext_wrapper_label_v:n { ##1 }
3347     }
3348     { ##1 }
3349     \tl_use:N \l__enumext_label_fill_right_v_tl
3350   }
3351 }

```

Definition using `\makebox` when `\DocumentMetadata` is active.

```

3352 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3353 {
3354   \RenewDocumentCommand \makeLabel { m }
3355   {
3356     \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3357     {

```



```

3358         \tl_use:N \l__enumext_label_font_style_v_tl
3359         \bool_if:NTF \l__enumext_wrap_label_v_bool
3360         {
3361             \__enumext_wrapper_label_v:n { ##1 }
3362         }
3363         { ##1 }
3364     }
3365 }
3366 }

```

(End of definition for `__enumext_keyans_make_label:`, `__enumext_keyans_make_label_std:`, and `__enumext_keyans_make_label_box:`.)

13.37 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`.

13.37.1 Calculation of `\leftmargin` and `\itemindent`

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

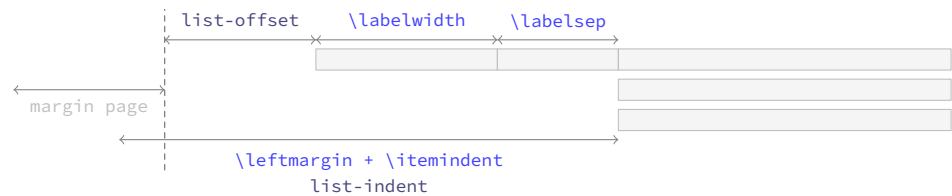


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

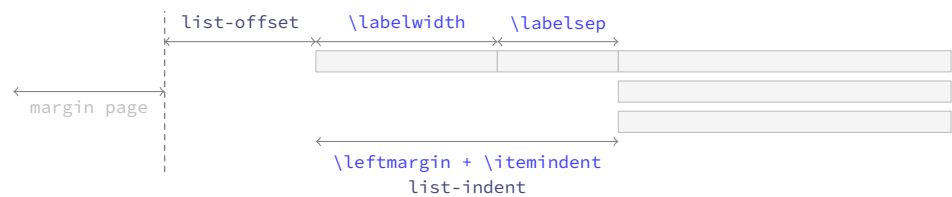


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

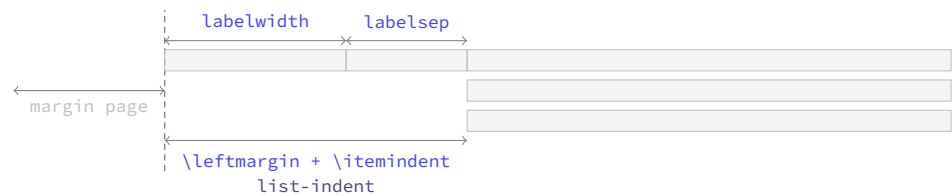


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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

```

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

```

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§13.37.2).

```

3367 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3368 {
3369     \dim_compare:nNt { #1 } < { \c_zero_dim }
3370     {
3371         \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3372         \dim_set:Nn #1 { \dim_abs:n { #1 } }

```

```

3373     }
3374     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3375     {
3376         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3377         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3378     }

```

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

```

3379     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3380     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3381     {
3382         \dim_set:Nn #6 { #1 + #2 - #4 }
3383         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3384     }
3385     {
3386         \dim_compare:nNnT { #4 } = { #1 + #2 }
3387         { \dim_set:Nn #6 { \c_zero_dim } }
3388         \dim_compare:nNnT { #4 } < { #1 + #2 }
3389         { \dim_set:Nn #6 { #1 + #2 - #4 } }
3390         \dim_compare:nNnT { #4 } > { #1 + #2 }
3391         {
3392             \dim_set:Nn #6 { -#1 - #2 + #4 }
3393             \dim_set:Nn #6 { #6*-1 }
3394         }
3395         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3396     }
3397 }
3398 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

13.37.2 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.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3399 \cs_set_protected:Npn \__enumext_tmp:n #1
3400 {
3401     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3402     {
3403         \__enumext_calc_hspace:ccccc
3404         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3405         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3406         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3407         { \__enumext_leftmargin_tmp_#1_bool }
3408         \clist_map_inline:nn
3409         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3410         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3411         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3412         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3413         \usecounter { enumX#1 }
3414         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3415         \str_if_eq:nnTF {#1} { v }
3416         {
3417             \__enumext_keyans_redefine_item:
3418             \__enumext_keyans_make_label:
3419             \__enumext_keyans_ref:
3420             \__enumext_keyans_fake_item_indent:
3421             \bool_if:cT { \__enumext_show_length_#1_bool }
3422             {
3423                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3424             }
3425         }
3426         {
3427             \__enumext_redefine_item:
3428             \__enumext_make_label:
3429             \__enumext_standar_ref:
3430             \__enumext_fake_item_indent:
3431             \bool_if:cT { \__enumext_show_length_#1_bool }
3432             {

```

```

3433         \msg_term:nne { enumext } { list-lengths } {#1}
3434         { \int_use:N \l__enumext_level_int }
3435     }
3436 }
3437 }
3438 }
3439 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `0pt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `listparindent` and `parsep` to set the value of `\parskip` locally.

```

3440 \cs_set_protected:Npn \__enumext_tmp:n #1
3441 {
3442     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3443     {
3444         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3445         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3446         \__enumext_calc_hspace:ccccc
3447         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3448         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3449         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3450         { l__enumext_leftmargin_tmp_#1_bool }
3451         \clist_map_inline:nn
3452         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3453         { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3454         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3455         { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3456         \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3457         \skip_zero:N \partopsep
3458         \usecounter { enumX#1 }
3459         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3460         \__enumext_starred_ref:
3461         \str_if_eq:nnTF {#1} { vii }
3462         {
3463             \__enumext_fake_item_indent_vii:
3464             \bool_if:cT { l__enumext_show_length_vii_bool }
3465             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3466         }
3467         {
3468             \__enumext_fake_item_indent_viii:
3469             \bool_if:cT { l__enumext_show_length_#1_bool }
3470             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3471         }
3472     }
3473 }
3474 \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:`.)

13.38 The environment `enumext`

```
\__enumext_safe_exec:
```

The `__enumext_safe_exec:` function first call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_standar_bool` to “true” if we are not nested within `enumext*`, 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”.

```

3475 \cs_new_protected:Nn \__enumext_safe_exec:
3476 {
3477     \__enumext_internal_mini_page:
3478     \__enumext_is_not_nested:
3479     \int_incr:N \l__enumext_level_int
3480     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3481     { \msg_fatal:nn { enumext } { list-too-deep } }
3482     \bool_set_true:N \l__enumext_standar_bool
3483     \bool_set_false:N \l__enumext_starred_bool
3484     \__enumext_is_on_first_level:
3485 }

```

(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.

```

3486 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3487 {
3488   \tl_if_novalue:nF {#1}
3489   {
3490     \str_clear:N \l__enumext_series_str
3491     \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3492     {
3493       \keys_set:nn { enumext / level-1 } {#1}
3494       \__enumext_parse_series:n {#1}
3495       \__enumext_nested_base_line_fix:
3496     }
3497     {
3498       \exp_args:Ne \keys_set:nn
3499         { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3500     }
3501     \__enumext_store_active_keys:n {#1}
3502   }
3503 }

```

(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*`.

```

3504 \cs_new_protected:Nn \__enumext_start_store_level:
3505 {
3506   \bool_lazy_all:nT
3507   {
3508     { \bool_if_p:N \l__enumext_store_active_bool }
3509     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3510     { \bool_if_p:N \g__enumext_standar_bool }
3511   }
3512   {
3513     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3514     {
3515       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3516       \__enumext_store_level_open:
3517     }
3518   }

```

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

```

3519   \bool_lazy_all:nT
3520   {
3521     { \bool_if_p:N \l__enumext_store_active_bool }
3522     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3523     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3524   }
3525   {
3526     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3527     {
3528       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3529       \__enumext_store_level_open:
3530     }
3531   }
3532 }

```

(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*`.

```

3533 \cs_new_protected:Nn \__enumext_stop_store_level:
3534 {
3535   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3536   {

```

```

3537     \__enumext_store_level_close:
3538   }
3539 }

```

(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.

```

3540 \cs_new_protected:Nn \__enumext_multicols_start:
3541 {
3542   \int_compare:nNtT
3543     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3544   {
3545     \dim_compare:nNtT
3546       { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3547     {
3548       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
3549         {
3550           ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
3551             + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3552             ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3553             - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3554           }
3555         }
3556       \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
3557       \int_compare:nNtT { \__enumext_level_int } > { 1 }
3558       {
3559         \dim_zero:N \columnseprule
3560       }
3561     }
3562   }
3563   \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
3564   \int_compare:nNtT { \__enumext_level_int } > { 1 }
3565   {
3566     \dim_zero:N \columnseprule
3567   }
3568 }

```

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.

```

3561   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3562   {
3563     \skip_zero:N \multicolsep
3564     \__enumext_multi_addvspace:
3565   }
3566   \raggedcolumns
3567   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
3568 }
3569 }

```

(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:`.

```

3570 \cs_new_protected:Nn \__enumext_multicols_stop:
3571 {
3572   \int_compare:nNtTF
3573     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3574     {
3575       \__enumext_stop_list:
3576       \__enumext_stop_store_level:
3577       \end{multicols}
3578       \__enumext_unskip_unkern:
3579       \__enumext_unskip_unkern:
3580       \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3581     }
3582     {
3583       \__enumext_stop_list:
3584       \__enumext_stop_store_level:
3585     }
3586 }

```

(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.

```

3587 \cs_new_protected:Nn \__enumext_before_list:
3588 {
3589   \__enumext_vspace_above:
3590   \__enumext_before_args_exec:
3591   \__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.

```

3592   \dim_compare:nNt
3593   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3594   {
3595     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3596     {
3597       \linewidth
3598       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3599       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3600     }

```

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.

```

3601     \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3602     \int_gincr:N \g__enumext_minipage_stat_int
3603     \__enumext_minipage_add_space:
3604     \noindent
3605     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3606   }
3607   \__enumext_multicols_start:
3608 }

```

(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.

```

3609 \cs_new_protected:Nn \__enumext_second_part:
3610 {
3611   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3612   {
3613     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3614     {
3615       \msg_warning:nn { enumext } { missing-miniright }
3616       \miniright
3617     }
3618     \int_gzero:N \g__enumext_minipage_stat_int
3619     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3620     \end__enumext_mini_page
3621   }
3622   {
3623     \__enumext_multicols_stop:
3624   }

```

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.

```

3625   \__enumext_after_stop_list:
3626   \__enumext_check_ans_key_hook:
3627   \__enumext_vspace_below:
3628   \bool_set_false:N \l__enumext_standar_bool
3629   \__enumext_resume_save_counter:
3630 }

```

(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.

```

3631 \cs_new_protected:Nn \__enumext_set_item_width:
3632 {
3633   \dim_set:Nn \itemwidth { \linewidth }
3634   \dim_compare:nT
3635     {
3636       \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3637     }
3638     {
3639       \dim_sub:Nn \itemwidth
3640         {
3641           \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3642         }
3643     }
3644 }

```

(End of definition for `__enumext_set_item_width:`.)

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

```

3645 \NewDocumentEnvironment{enumext}{0}{ }
3646 {
3647   \__enumext_safe_exec:
3648   \__enumext_parse_keys:n {#1}
3649   \__enumext_before_list:
3650   \__enumext_start_store_level:
3651   \__enumext_start_list:nn
3652     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3653     {
3654       \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3655       \__enumext_before_keys_exec:
3656     }
3657   \__enumext_set_item_width:
3658   \__enumext_after_args_exec:
3659 }
3660 {
3661   \__enumext_second_part:
3662 }

```

(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`.

```

3663 \__enumext_after_env:nn {enumext}
3664 {
3665   \__enumext_execute_after_env:
3666 }

```

13.39 The environment keyans

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

`__enumext_keyans_safe_exec:` The `keyans` environment will only be available if the `save-ans` key is active and can only be used at the “first level” within the `enumext` environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```

3667 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3668 {
3669   \bool_if:NF \l__enumext_store_active_bool
3670     {
3671       \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3672     }
3673   \int_incr:N \l__enumext_keyans_level_int
3674   \bool_set_true:N \l__enumext_keyans_env_bool
3675   \__enumext_keyans_name_and_start:
3676   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3677   \bool_set_false:N \l__enumext_store_active_bool
3678   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3679   {

```



```

3680         \msg_error:nn { enumext } { keyans-nested }
3681     }
3682     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3683     {
3684         \msg_error:nn { enumext } { keyans-wrong-level }
3685     }
3686 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3687 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3688 {
3689     \keys_set:nn { enumext / keyans } {#1}
3690 }

```

(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:
3691 \cs_new_protected:Nn \__enumext_before_list_v:
3692 {
3693     \__enumext_vspace_above_v:
3694     \__enumext_before_args_exec_v:
3695     \dim_compare:nNt { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3696     {
3697         \dim_set:Nn \l__enumext_minipage_left_v_dim
3698         {
3699             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3700         }
3701         \bool_set_true:N \l__enumext_minipage_active_v_bool
3702         \int_gincr:N \g__enumext_minipage_stat_int
3703         \__enumext_keyans_minipage_add_space:
3704         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3705     }
3706     \__enumext_keyans_multicols_start:
3707 }
3708 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3709 {
3710     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3711     {
3712         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3713         {
3714             \dim_set:Nn \l__enumext_columns_sep_v_dim
3715             {
3716                 (
3717                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3718                 ) / \l__enumext_columns_v_int
3719                 - \l__enumext_listoffset_v_dim
3720             }
3721         }
3722         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3723         \dim_zero:N \columnseprule % no rule here
3724         \bool_if:NF \l__enumext_minipage_active_v_bool
3725         {
3726             \skip_zero:N \multicolsep
3727             \__enumext_keyans_multi_addvspace:
3728         }
3729         \raggedcolumns
3730         \begin{multicols}{ \l__enumext_columns_v_int }
3731     }
3732 }
3733 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3734 {
3735     \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }
3736     {
3737         \__enumext_stop_list:
3738         \end{multicols}
3739         \__enumext_unskip_unkern:
3740         \__enumext_unskip_unkern:
3741         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3742     }

```

```

3743     {
3744       \__enumext_stop_list:
3745     }
3746   }
3747   \cs_new_protected:Nn \__enumext_second_part_v:
3748   {
3749     \bool_if:NTF \l__enumext_minipage_active_v_bool
3750     {
3751       \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3752       {
3753         \msg_warning:nn { enumext } { missing-miniright }
3754         \miniright
3755       }
3756       \int_gzero:N \g__enumext_minipage_stat_int
3757       \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3758       \end__enumext_mini_page
3759       \par\addvspace{ \l__enumext_minipage_after_skip }
3760     }
3761     {
3762       \__enumext_keyans_multicols_stop:
3763     }
3764     \bool_set_false:N \l__enumext_keyans_env_bool
3765     \__enumext_after_stop_list_v:
3766     \__enumext_vspace_below_v:
3767   }

```

(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.

```

3768   \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3769   {
3770     \dim_set:Nn \itemwidth { \linewidth }
3771     \dim_compare:nT
3772     {
3773       \l__enumext_listoffset_v_dim != \c_zero_dim
3774     }
3775     {
3776       \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3777     }
3778   }

```

(End of definition for __enumext_keyans_set_item_width:.)

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

```

3779   \NewDocumentEnvironment{keyans}{0}{ }
3780   {
3781     \__enumext_keyans_safe_exec:
3782     \__enumext_keyans_parse_keys:n {#1}
3783     \__enumext_before_list_v:
3784     \__enumext_start_list:nn
3785     { \tl_use:N \l__enumext_label_v_tl }
3786     {
3787       \__enumext_list_arg_two_v:
3788       \__enumext_before_keys_exec_v:
3789     }
3790     \__enumext_keyans_set_item_width:
3791     \__enumext_after_args_exec_v:
3792   }
3793   {
3794     \__enumext_check_starred_cmd:n { item }
3795     \__enumext_second_part_v:
3796   }

```

(End of definition for keyans. This function is documented on page 14.)

13.40 Tagging PDF support for non-standart list environments

The L^AT_EX 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. 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](#).

13.40.1 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
3797 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3798 \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3799 {
3800   \tag_resume:n {#1}
3801   \tag_struct_begin:n {tag=LI}
3802   \tag_struct_begin:n {tag=Lbl}
3803   \tag_mc_begin:n {tag=Lbl}
3804 }
3805 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3806 {
3807   \tag_mc_end:
3808   \tag_struct_end:n {tag=Lbl}
3809   \tag_struct_begin:n {tag=LBody}
3810   \tag_struct_begin:n {tag=text-unit}
3811   \tag_struct_begin:n {tag=text}
3812 }
3813 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3814 {
3815   \tag_struct_end:n {tag=text}
3816   \tag_struct_end:n {tag=text-unit}
3817   \tag_struct_end:n {tag=LBody}
3818   \tag_struct_end:n {tag=LI}
3819   \tag_suspend:n {#1}
3820 }
```

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

```

3821 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
3822 {
3823   \IfDocumentMetadataTF
3824   {
3825     \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3826     \socket_use:n {tagsupport/enumext/starred} {#1}
3827   } {}
3828 }
3829 \cs_new_protected_nopar:Nn __enumext_stop_start_list_tag:
3830 {
3831   \IfDocumentMetadataTF
3832   {
3833     \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3834     \socket_use:nn {tagsupport/enumext/starred} { }
3835   } {}
3836 }
3837 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
3838 {
3839   \IfDocumentMetadataTF
3840   {
3841     \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3842     \socket_use:nn {tagsupport/enumext/starred} {#1}
3843   } {}
3844 }
```

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

13.40.2 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:
3845 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3846 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3847 {
3848   \tag_resume:n {keyanspic}
3849   \tag_struct_begin:n {tag=LI}
3850   \tag_struct_begin:n {tag=Lbl}
3851   \tag_mc_begin:n {tag=Lbl}
3852 }
```

```
3853 \socket_new_plugin:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3854 {
3855   \tag_mc_end:
3856   \tag_struct_end:n {tag=Lbl}
3857   \tag_struct_begin:n {tag=LBody}
3858   \tag_struct_begin:n {tag=text-unit}
3859   \tag_struct_begin:n {tag=text}
3860   \tag_mc_begin:n {tag=text}
3861 }
3862 \socket_new_plugin:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3863 {
3864   \tag_mc_end:
3865   \tag_struct_end:n {tag=text-unit}
3866   \tag_struct_end:n {tag=text}
3867   \tag_struct_end:n {tag=LBody}
3868   \tag_struct_end:n {tag=LI}
3869   \tag_suspend:n {keyanspic}
3870 }
```

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

```
3871 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3872 {
3873   \IfDocumentMetadataTF
3874   {
3875     \socket_assign_plugin:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3876     \socket_use:n {tagsupport/enumext/keyanspic}
3877   } {}
3878 }
3879 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3880 {
3881   \IfDocumentMetadataTF
3882   {
3883     \socket_assign_plugin:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3884     \socket_use:nn {tagsupport/enumext/keyanspic}
3885   } {}
3886 }
3887 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3888 {
3889   \IfDocumentMetadataTF
3890   {
3891     \socket_assign_plugin:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3892     \socket_use:nn {tagsupport/enumext/keyanspic}
3893   } {}
3894 }
```

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

13.41 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.

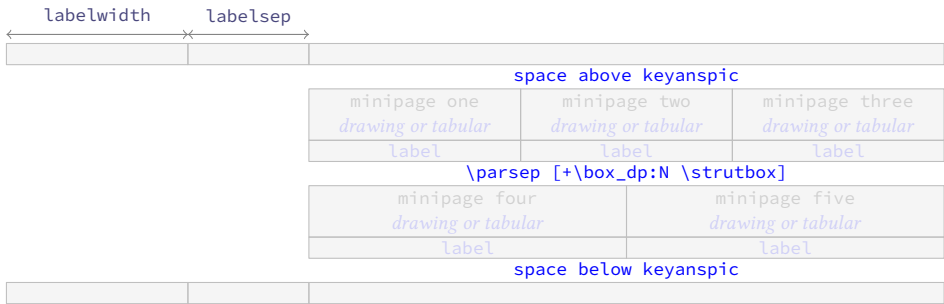


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

The environment `keyanspic` will take two arguments, the first *starred argument* “`*`” will set the position of the `<label>` processed by the command `\anspic` which will be “above” if present and “below” otherwise, the second *optional argument* 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 environment, if not present these will be printed on a *single line*.

- ◆ One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of $\langle label \rangle$, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to $\langle label \rangle$ and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is $\langle label \rangle$, 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.

13.41.1 The environment `keyanspic`

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?](#).

`__enumext_keyans_pic_safe_exec:n`

The function `__enumext_keyans_pic_safe_exec:n` check the *starred argument* ‘*’ and nested level position inside the `enumext` environment. We will set the state of the variable `\l__enumext_keyans_pic_star_bool` along with the value of the variable `\l__enumext_anspic_mini_pos_str` using by `\anspic` according to the presence of the *starred argument* ‘*’.

```

3895 \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3896 {
3897   \int_incr:N \l__enumext_keyans_pic_level_int
3898   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3899   {
3900     \msg_error:nn { enumext } { keyanspic-nested }
3901   }
3902   \__enumext_keyans_name_and_start:
3903   \bool_if:nTF { #1 }
3904   {
3905     \bool_set_true:N \l__enumext_keyans_pic_star_bool
3906     \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
3907   }
3908   {
3909     \str_set:Nn \l__enumext_anspic_mini_pos_str { b }
3910   }
3911 }

```

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

`__enumext_keyans_pic_skip_abs:N`

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

```

3912 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3913 {
3914   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3915   {
3916     \skip_set:Nn #1 { -#1 }
3917   }
3918 }

```

(End of definition for `__enumext_keyans_pic_skip_abs:N`.)

`__enumext_keyans_pic_arg_two:`

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 $\langle keys \rangle$ label and `wrap-label` 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.

```

3919 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3920 {
3921   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3922   \__enumext_list_arg_two_v:
3923   \__enumext_keyans_pic_skip_abs:N \parsep

```

Now we increment the `enumXv` counter of the `keyans` environment and save the *total height* of the $\langle label \rangle$ in `\l__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the *starred argument* ‘*’ is NOT present.

```

3924   \bool_if:NF \l__enumext_keyans_pic_star_bool
3925   {
3926     \stepcounter { enumXv }
3927     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3928     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3929     {
3930       \box_ht_plus_dp:N \l__enumext_anspic_label_box
3931     }
3932     \skip_add:Nn \parsep

```

```

3933     {
3934         \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3935     }
3936     \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3937 }

```

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.

```

3938     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3939     \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3940     \dim_zero:N \listparindent
3941     \skip_zero:N \partopsep
3942     \skip_zero:N \itemsep
3943 }

```

(End of definition for `__enumext_keyans_pic_arg_two:`)

keyanspic Now we define the environment `keyanspic`. For compatibility with *tagged* PDF we must use the `\beginlist` form and a lot of conditional code using `\IfDocumentMetadataTF`.

```

3944 \NewDocumentEnvironment{keyanspic}{ s o }
3945 {
3946     \__enumext_keyans_pic_safe_exec:n { #1 }
3947     \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3948     \IfDocumentMetadataTF
3949     {
3950         \tag_suspend:n {list}
3951     }{}
3952     \item[] \scan_stop:
3953     % paranoia
3954     \RenewDocumentCommand \item {}
3955     {
3956         \msg_error:nn { enumext } { keyanspic-item-cmd }
3957     }
3958     \IfDocumentMetadataTF
3959     {
3960         \tag_resume:n {keyanspic}
3961         \tag_tool:n {para/tagging=false}
3962         \tag_suspend:n {keyanspic}
3963     } { }
3964 }
3965 {
3966     \IfDocumentMetadataTF
3967     {
3968         \tag_resume:n {keyanspic}
3969         \tag_struct_begin:n {tag=L,attribute=enumerate}
3970     } { }

```

Now we process the command `\anspic`, if the *optional argument* is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_anspic_args_seq` and placed a single line.

```

3971     \tl_if_novalue:nTF { #2 }
3972     {
3973         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
3974     }
3975     { \__enumext_anspic_print:n { #2 } }
3976     \IfDocumentMetadataTF
3977     {
3978         \tag_suspend:n {keyanspic}
3979     } { }
3980     \end{list}
3981     \IfDocumentMetadataTF
3982     {
3983         \tag_struct_end:
3984         \tag_struct_end:
3985     } { }

```

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

```

3986     \__enumext_check_starred_cmd:n { anspic }
3987     \setcounter { enumXvi } { 0 }
3988     \bool_if:NTF \l__enumext_keyans_pic_star_bool
3989     {
3990         \par\addvspace{ 0.5\box_dp:N \strutbox }

```

```

3991     }
3992     {
3993         \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
3994     }
3995     %\bool_set_false:N \l__enumext_store_active_bool
3996 }

```

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

13.41.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*` [`<content>`] store the current `<label>` next to the *optional argument* [`<content>`] in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* [`<drawing or tabular>`] is NOT stored in the *sequence* or *prop list*.

`\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 `__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

3997 \NewDocumentCommand \anspic { s o +m }
3998 {
3999     \bool_if:NF \l__enumext_store_active_bool
4000     {
4001         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4002     }
4003     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4004     {
4005         \msg_error:nn { enumext } { keyanspic-wrong-level }
4006     }
4007     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4008     {
4009         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4010     }
4011     \seq_put_right:Nn \l__enumext_anspic_args_seq
4012     {
4013         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4014     }
4015 }

```

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

`__enumext_anspic_body_dim:n` The `__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the height and depth of the mandatory argument if the `keyanspic*` environment is used with the *starred argument* ‘*’.

```

4016 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4017 {
4018     \bool_if:NF \l__enumext_keyans_pic_star_bool
4019     {
4020         \IfDocumentMetadataTF
4021         {
4022             \tag_suspend:n {keyanspic}
4023         } { }
4024         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4025         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4026         {
4027             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4028         }
4029         \IfDocumentMetadataTF
4030         {
4031             \tag_resume:n {keyanspic}
4032         } { }
4033     }
4034 }

```

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

`__enumext_anspic_label:nn` 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` and `wrap-opt` keys.

```

4035 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4036 {

```



```

4037 \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4038 {
4039   \bool_if:nT { #1 }
4040   {
4041     \__enumext_keyans_addto_prop:n { #2 }
4042     \__enumext_keyans_store_ref:
4043     \__enumext_keyans_addto_seq:n { #2 }
4044     \int_gincr:N \g__enumext_check_starred_cmd_int
4045     \bool_lazy_or:nnT
4046     { \bool_if_p:N \l__enumext_show_answer_bool }
4047     { \bool_if_p:N \l__enumext_show_position_bool }
4048     {
4049       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4050       \__enumext_keyans_show_left:n { #2 }
4051       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4052     }
4053   }
4054   \tl_use:N \l__enumext_label_font_style_v_tl
4055   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4056   \__enumext_keyans_show_item_opt:
4057 }
4058 }

```

(End of definition for __enumext_anspic_label:nn.)

__enumext_anspic_label_pos:nnn

The function __enumext_anspic_label_pos:nnn will be in charge of handling the “counter” and the position of the $\langle label \rangle$, which will have the same configuration as the **keyans** environment.

```

4059 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4060 {
4061   \stepcounter { enumXvi }
4062   \__enumext_anspic_body_dim:n { #3 }
4063   \bool_if:NTF \l__enumext_keyans_pic_star_bool
4064   {
4065     \__enumext_anspic_label:nn { #1 } { #2 }
4066   }
4067   {
4068     \raisebox
4069     {
4070       -\dim_eval:n
4071       {
4072         \l__enumext_anspic_label_htdp_dim
4073         + \l__enumext_anspic_body_htdp_dim
4074         + \box_dp:N \strutbox
4075       }
4076     }
4077     [ opt ] [ opt ]
4078     {
4079       \__enumext_anspic_label:nn { #1 } { #2 }
4080     }
4081   }
4082 }
4083 %

```

(End of definition for __enumext_anspic_label_pos:nnn.)

__enumext_anspic_args:nnn

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.

```

4084 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4085 {
4086   \__enumext_anspic_start_list_tag:
4087   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4088   \__enumext_anspic_stop_start_list_tag:
4089   \\\ #3
4090   \__enumext_anspic_stop_list_tag:
4091 }

```

(End of definition for __enumext_anspic_args:nnn.)

```

\__enumext_anspic_print:n
\__enumext_anspic_print:e
\__enumext_anspic_row:n

```

The *optional argument* [$\langle n^{\circ} upper, n^{\circ} lower \rangle$] passed to the `keyanspic` environment is split by comma and is handled directly by the function `__enumext_anspic_print:n` and passed to the function `__enumext_anspic_row:n`.

```

4092 \cs_new_protected:Nn \__enumext_anspic_print:n
4093 {
4094   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4095 }
4096 \cs_generate_variant:Nn \__enumext_anspic_print:n { e }

```

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.

```

4097 \cs_new_protected:Nn \__enumext_anspic_row:n
4098 {
4099   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4100   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4101   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4102   \int_step_inline:nnn
4103     { \l__enumext_anspic_above_int + 1 }
4104     { \l__enumext_anspic_below_int }
4105     {
4106       \IfDocumentMetadataTF
4107       {
4108         \tag_suspend:n {minipage}
4109       } { }
4110       \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4111         \centering
4112         \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4113       \end{minipage}
4114       \IfDocumentMetadataTF
4115       {
4116         \tag_resume:n {minipage}
4117       } { }
4118     }
4119   \par
4120 }

```

(End of definition for `__enumext_anspic_print:n` and `__enumext_anspic_row:n`.)

13.42 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard \LaTeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. 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.

13.42.1 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:

```

```

4121 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4122 {
4123   \dim_compare:nNNT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4124   {
4125     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4126     {

```

```

4127         ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4128         / \l__enumext_columns_vii_int
4129     }
4130 }
4131 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4132 \dim_set:Nn \l__enumext_item_width_vii_dim
4133 {
4134     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4135     / \l__enumext_columns_vii_int
4136     - \l__enumext_labelwidth_vii_dim
4137     - \l__enumext_labelsep_vii_dim
4138 }

```

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

```

4139     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4140     {
4141         \dim_sub:Nn \l__enumext_item_width_vii_dim
4142         {
4143             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4144             / \l__enumext_columns_vii_int
4145         }
4146         \dim_add:Nn \l__enumext_columns_sep_vii_dim
4147         {
4148             \l__enumext_rightmargin_vii_dim
4149         }
4150     }
4151 }

```

Same implementation for the `keyans*` environment.

```

4152 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4153 {
4154     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4155     {
4156         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4157         {
4158             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4159             / \l__enumext_columns_viii_int
4160         }
4161     }
4162     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4163     \dim_set:Nn \l__enumext_item_width_viii_dim
4164     {
4165         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4166         / \l__enumext_columns_viii_int
4167         - \l__enumext_labelwidth_viii_dim
4168         - \l__enumext_labelsep_viii_dim
4169     }
4170     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4171     {
4172         \dim_sub:Nn \l__enumext_item_width_viii_dim
4173         {
4174             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
4175             / \l__enumext_columns_viii_int
4176         }
4177         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4178         {
4179             \l__enumext_rightmargin_viii_dim
4180         }
4181     }
4182 }

```

(End of definition for `__enumext_starred_columns_set_vii:` and `__enumext_starred_columns_set_viii:`)

13.42.2 Functions for join item columns

```

\__enumext_starred_joined_item_vii:n
\__enumext_starred_joined_item_viii:n

```

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the `⟨content⟩` passed to `\item(⟨columns⟩)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4183 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4184 {
4185     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4186     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4187     {

```

```

4188         \msg_warning:nnee { enumext } { item-joined }
4189         { \int_use:N \l__enumext_joined_item_vii_int }
4190         { \int_use:N \l__enumext_columns_vii_int }
4191         \int_set:Nn \l__enumext_joined_item_vii_int
4192         {
4193             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4194         }
4195     }
4196     \int_compare:nNnT
4197     { \l__enumext_joined_item_vii_int }
4198     >
4199     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4200     {
4201         \msg_warning:nnee { enumext } { item-joined-columns }
4202         { \int_use:N \l__enumext_joined_item_vii_int }
4203         {
4204             \int_eval:n
4205             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4206         }
4207         \int_set:Nn \l__enumext_joined_item_vii_int
4208         {
4209             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4210         }
4211     }
4212     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4213     {
4214         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4215         \int_decr:N \l__enumext_joined_item_aux_vii_int
4216         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4217         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4218         \dim_set:Nn \l__enumext_joined_width_vii_dim
4219         {
4220             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4221             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4222               + \l__enumext_columns_sep_vii_dim
4223               ) * \l__enumext_joined_item_aux_vii_int
4224         }
4225         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4226     }
4227     {
4228         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4229         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4230     }
4231 }

```

Same implementation for the [keyans*](#) environment.

```

4232 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4233 {
4234     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4235     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4236     {
4237         \msg_warning:nnee { enumext } { item-joined }
4238         { \int_use:N \l__enumext_joined_item_viii_int }
4239         { \int_use:N \l__enumext_columns_viii_int }
4240         \int_set:Nn \l__enumext_joined_item_viii_int
4241         {
4242             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4243         }
4244     }
4245     \int_compare:nNnT
4246     { \l__enumext_joined_item_viii_int }
4247     >
4248     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4249     {
4250         \msg_warning:nnee { enumext } { item-joined-columns }
4251         { \int_use:N \l__enumext_joined_item_viii_int }
4252         {
4253             \int_eval:n
4254             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4255         }
4256         \int_set:Nn \l__enumext_joined_item_viii_int
4257         {

```

```

4258         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4259     }
4260 }
4261 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4262 {
4263     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4264     \int_decr:N \l__enumext_joined_item_aux_viii_int
4265     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4266     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4267     \dim_set:Nn \l__enumext_joined_width_viii_dim
4268     {
4269         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4270         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4271             + \l__enumext_columns_sep_viii_dim
4272             ) * \l__enumext_joined_item_aux_viii_int
4273     }
4274     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4275 }
4276 {
4277     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4278     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4279 }
4280 }

```

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

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

```

\__enumext_start_mini_vii:
\__enumext_stop_mini_vii:

```

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

```

4281 \cs_new_protected:Nn \__enumext_start_mini_vii:
4282 {
4283     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4284     {
4285         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4286         {
4287             \linewidth
4288             - \l__enumext_minipage_right_vii_dim
4289             - \l__enumext_minipage_hsep_vii_dim
4290         }
4291         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4292         \dim_gset_eq:NN
4293             \g__enumext_minipage_right_vii_dim
4294             \l__enumext_minipage_right_vii_dim
4295         \__enumext_mini_addvspace_vii:
4296         \nointerlineskip\noindent
4297         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4298     }
4299 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:n` to execute the `minipage` on the “right side”. At this point we will execute the `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.43).

```

4300 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4301 {
4302     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4303     {
4304         \__enumext_stop_list:
4305         \__enumext_stop_store_level_vii:
4306         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { } { }
4307         \end__enumext_mini_page
4308         \hfill
4309         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4310     }
4311     {
4312         \__enumext_stop_list:

```

```

4313     \__enumext_stop_store_level_vii:
4314 }
4315 }

```

(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.

```

4316 \__enumext_after_env:nn {enumext*}
4317 {
4318   \bool_if:NT \g__enumext_minipage_active_vii_bool
4319   {
4320     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4321     \legacy_if_gset_false:n { @minipage }
4322     \skip_vertical:N \c_zero_skip
4323     \par\addvspace { \g__enumext_minipage_right_skip }
4324     \bool_if:NF \g__enumext_minipage_center_vii_bool
4325     {
4326       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4327       {
4328         \centering
4329       }
4330     }
4331     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4332     {
4333       \tl_use:N \g__enumext_miniright_code_vii_tl
4334     }
4335     \box_use_drop:N \l__enumext_miniright_code_vii_box
4336     \skip_vertical:N \c_zero_skip
4337     \__enumext_endminipage:
4338     \par\addvspace{ \g__enumext_minipage_after_skip }
4339   }
4340   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4341   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4342   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4343   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4344   \bool_gset_false:N \g__enumext_starred_bool
4345 }

```

`__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.

```

4346 \cs_new_protected:Nn \__enumext_start_mini_viii:
4347 {
4348   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4349   {
4350     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4351     {
4352       \linewidth
4353       - \l__enumext_minipage_right_viii_dim
4354       - \l__enumext_minipage_hsep_viii_dim
4355     }
4356     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4357     \dim_gset_eq:NN
4358       \g__enumext_minipage_right_viii_dim
4359       \l__enumext_minipage_right_viii_dim
4360     \__enumext_mini_addvspace_viii:
4361     \nointerlineskip\noindent
4362     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4363   }
4364 }
4365 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4366 {
4367   \bool_if:NTF \l__enumext_minipage_active_viii_bool
4368   {
4369     \__enumext_stop_list:
4370     \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4371     \end__enumext_mini_page
4372     \hfill
4373     \bool_gset_true:N \g__enumext_minipage_active_viii_bool

```

```

4374     }
4375     {
4376         \__enumext_stop_list:
4377     }
4378 }
4379 \__enumext_after_env:nn {keyans*}
4380 {
4381     \bool_if:NT \g__enumext_minipage_active_viii_bool
4382     {
4383         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4384         \par\addvspace { \g__enumext_minipage_right_skip }
4385         \bool_if:NF \g__enumext_minipage_center_viii_bool
4386         {
4387             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4388             {
4389                 \centering
4390             }
4391         }
4392         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4393         {
4394             \tl_use:N \g__enumext_miniright_code_viii_tl
4395         }
4396         \box_use_drop:N \l__enumext_miniright_code_viii_box
4397         \end__enumext_mini_page
4398         \par\addvspace{ \g__enumext_minipage_after_skip }
4399     }
4400     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4401     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4402     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4403     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4404 }

```

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

13.42.4 Redefining \footnote command

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

```

4405 \cs_new_protected:Nn \__enumext_footnotetext:nn
4406 {
4407     \footnotetext[#1]{#2}
4408 }
4409 \cs_new_protected:Nn \__enumext_renew_footnote:
4410 {
4411     \seq_gclear:N \g__enumext_footnote_arg_seq
4412     \seq_gclear:N \g__enumext_footnote_int_seq
4413     \RenewDocumentCommand \footnote { o +m }
4414     {
4415         \tl_if_novalue:nTF {##1}
4416         {
4417             \stepcounter{footnote}
4418             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4419         }
4420         {
4421             \int_gset:Nn \g__enumext_footnote_int { ##1 }
4422         }
4423         \footnotemark [ \g__enumext_footnote_int ]
4424         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4425         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4426     }
4427 }
4428 \cs_new_protected:Nn \__enumext_print_footnote:
4429 {
4430     \seq_if_empty:NF \g__enumext_footnote_int_seq
4431     {
4432         \seq_map_pairwise_function:NNN
4433         \g__enumext_footnote_int_seq
4434         \g__enumext_footnote_arg_seq
4435         \__enumext_footnotetext:nn
4436     }
4437 }

```


(End of definition for `__enumext_footnotetext:nn`, `__enumext_renew_footnote:`, and `__enumext_print_footnote:.`)

13.43 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 .

```

4438 \NewDocumentEnvironment{enumext*}{o }
4439 {
4440   \__enumext_safe_exec_vii:
4441   \__enumext_parse_keys_vii:n {#1}
4442   \__enumext_before_list_vii:
4443   \__enumext_start_store_level_vii:
4444   \__enumext_start_list:nn { }
4445   {
4446     \__enumext_list_arg_two_vii:
4447     \__enumext_before_keys_exec_vii:
4448   }
4449   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4450   \__enumext_starred_columns_set_vii:
4451   \item[] \scan_stop:
4452   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4453   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4454   \ignorespaces
4455 }
4456 {
4457   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4458   \__enumext_stop_item_tmp_vii:
4459   \__enumext_remove_extra_parsep_vii:
4460   \__enumext_after_list_vii:
4461 }

```

(End of definition for `enumext*`. This function is documented on page 5.)

`__enumext_safe_exec_vii:`

We will first call the function `__enumext_internal_mini_page:` to create the environment `__enumext-mini_page`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are not nested within `enumext`, 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.

```

4462 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4463 {
4464   \__enumext_internal_mini_page:
4465   \__enumext_is_not_nested:
4466   \int_incr:N \l__enumext_level_h_int
4467   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4468   {
4469     \msg_error:nn { enumext } { nested }
4470   }
4471   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4472   {
4473     \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4474   }
4475   \bool_set_true:N \l__enumext_starred_bool
4476   \bool_set_false:N \l__enumext_standar_bool
4477   \__enumext_is_on_first_level:
4478 }

```

(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.

```

4479 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4480 {
4481   \tl_if_novalue:nF {#1}

```

```

4482     {
4483         \str_clear:N \__enumext_series_str
4484         \keys_set:nn { enumext / enumext* } {#1}
4485         \__enumext_parse_series:n {#1}
4486         \__enumext_store_active_keys_vii:n {#1}
4487     }
4488 }

```

(End of definition for __enumext_parse_keys_vii:n.)

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

```

4489 \cs_new_protected:Nn \__enumext_before_list_vii:
4490 {
4491     \__enumext_vspace_above_vii:
4492     \__enumext_check_ans_active:
4493     \__enumext_before_args_exec_vii:
4494     \__enumext_start_mini_vii:
4495 }

```

(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: (§13.42.3) 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.

```

4496 \cs_new_protected:Nn \__enumext_after_list_vii:
4497 {
4498     \__enumext_stop_mini_vii:
4499     \__enumext_after_stop_list_vii:
4500     \__enumext_check_ans_key_hook:
4501     \__enumext_vspace_below_vii:
4502     \bool_set_false:N \__enumext_starred_bool
4503     \__enumext_resume_save_counter:
4504 }

```

(End of definition for __enumext_after_list_vii:.)

__enumext_start_store_level_vii: __enumext_stop_store_level_vii: The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions activate the “*storing structure*” mechanism in *sequence* for \anskey command and anskey* environment if enumext* are nested in enumext.

```

4505 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4506 {
4507     \bool_if:NT \__enumext_store_active_bool
4508     {
4509         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4510         {
4511             \__enumext_store_level_open_vii:
4512         }
4513     }
4514 }
4515 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4516 {
4517     \bool_if:NT \__enumext_store_active_bool
4518     {
4519         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4520         {
4521             \__enumext_store_level_close_vii:
4522         }
4523     }
4524 }

```

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

13.43.1 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.

```
4525 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4526 {
4527   \skip_horizontal:n { -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim }
4528 }
```

(End of definition for `__enumext_first_item_tmp_vii:`.)

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

```
4529 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4530 {
4531   \__enumext_stop_item_tmp_vii:
4532   \int_incr:N \__enumext_item_column_pos_vii_int
4533   \int_gincr:N \g__enumext_item_count_all_vii_int
4534   \__enumext_item_peek_args_vii:
4535 }
```

(End of definition for `__enumext_start_item_tmp_vii:`.)

`__enumext_item_peek_args_vii:` 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).

```
4536 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4537 {
4538   \peek_meaning:NTF (
4539     { \__enumext_joined_item_vii:w }
4540     { \__enumext_joined_item_vii:w (1) }
4541 }
```

(End of definition for `__enumext_item_peek_args_vii:`.)

`__enumext_joined_item_vii:w` 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`.

```
4542 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4543 {
4544   \__enumext_starred_joined_item_vii:n {#1}
4545   \peek_meaning_remove:NTF *
4546     { \__enumext_starred_item_vii:w }
4547     { \__enumext_standar_item_vii:w }
4548 }
```

(End of definition for `__enumext_joined_item_vii:w`.)

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

```
4549 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4550 {
4551   \bool_set_false:N \__enumext_item_starred_vii_bool
4552   \peek_meaning:NTF [
4553     {
4554       \bool_set_eq:NN \__enumext_wrap_label_vii_bool \__enumext_wrap_label_opt_vii_bool
4555       \__enumext_start_item_vii:w
4556     }
4557     {
4558       \bool_set_true:N \__enumext_wrap_label_vii_bool
```

```

4559         \legacy_if_set_true:n { @noitemarg }
4560         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4561     }
4562 }

```

(End of definition for __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

```

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

```

4563 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4564 {
4565     \bool_set_true:N \__enumext_item_starred_vii_bool
4566     \bool_set_true:N \__enumext_wrap_label_vii_bool
4567     \peek_meaning:NTF [
4568         { \__enumext_starred_item_vii_aux_i:w }
4569         { \__enumext_starred_item_vii_aux_ii:w }
4570     }
4571 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4572 {
4573     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4574     \__enumext_starred_item_vii_aux_ii:w
4575 }
4576 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4577 {
4578     \peek_meaning:NTF [
4579         { \__enumext_starred_item_vii_aux_iii:w }
4580         {
4581             \dim_set_eq:NN \__enumext_item_symbol_sep_vii_dim \__enumext_labelsep_vii_dim
4582             \legacy_if_set_true:n { @noitemarg }
4583             \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4584         }
4585     }
4586 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4587 {
4588     \dim_set:Nn \__enumext_item_symbol_sep_vii_dim {#1}
4589     \legacy_if_set_true:n { @noitemarg }
4590     \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4591 }

```

(End of definition for __enumext_starred_item_vii:w and others.)

```
\__enumext_fake_make_label_vii:n
```

The __enumext_fake_make_label_vii:n function will be in charge of handling our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[\langle symbol \rangle][\langle offset \rangle]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a `\skip_horizontal:N` and `\ignorespaces`.

- For compatibility with *tagged* PDF and `hyperref` when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lualatex`

```

4592 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4593 {
4594     \legacy_if:nT { @noitemarg }
4595     {
4596         \legacy_if_set_false:n { @noitemarg }
4597         \legacy_if:nT { @nmbrrlist }
4598         {
4599             \IfDocumentMetadataTF
4600             {
4601                 \bool_if:NT \__enumext_hyperref_bool
4602                 {
4603                     \legacy_if_set_true:n { @hyper@item }
4604                 }
4605             } { }
4606             \refstepcounter{enumXvii}
4607             \bool_if:NT \__enumext_check_answers_bool
4608             {
4609                 \int_gincr:N \g__enumext_item_number_int

```

```

4610         \bool_set_true:N \l__enumext_item_number_bool
4611     }
4612 }
4613 }
4614 \bool_if:NT \l__enumext_item_starred_vii_bool
4615 {
4616     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4617     {
4618         \tl_gset_eq:NN
4619         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4620     }
4621     \mode_leave_vertical:
4622     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4623     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4624     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4625     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4626 }
4627 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4628 {
4629     \tl_use:N \l__enumext_label_font_style_vii_tl
4630     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4631     {
4632         \__enumext_wrapper_label_vii:n {#1}
4633     }
4634     { #1 }
4635 }
4636 \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4637 }

```

(End of definition for __enumext_fake_make_label_vii:n.)

13.43.2 Real definition of \item in enumext*

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

__enumext_start_item_vii:w
 __enumext_stop_item_vii:

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

```

4638 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4639 {
4640     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4641     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4642     {
4643         \l__enumext_joined_width_vii_dim
4644         + \l__enumext_labelwidth_vii_dim
4645         + \l__enumext_labelsep_vii_dim
4646     }

```

If \DocumentMetadata is not active and the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the \footnote command.

```

4647     \IfDocumentMetadataTF { }
4648     {
4649         \bool_if:NF \l__enumext_footnotes_key_bool
4650         {
4651             \__enumext_renew_footnote:
4652         }
4653     }

```

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

```

4654     \__enumext_start_list_tag:n {enumext*}
4655     \__enumext_fake_make_label_vii:n {#1}
4656     \__enumext_stop_start_list_tag:

```

Finally we open the minipage environment, capture the “item content” and execute first and itemindent keys, then listparindent key which will be equal to \parindent, then parsep key which will be equal to \parskip.

```

4657     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4658     \tl_use:N \l__enumext_after_list_args_vii_tl
4659     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4660     \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4661     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip

```

```
4662 }
```

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*.

```
4663 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4664 {
4665     \__enumext_endminipage:
4666     \__enumext_stop_list_tag:n {enumext*}
4667     \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* and *footnotes* if it was redefined.

```
4668     \int_set:Nn \hbadness { 10000 }
4669     \box_use_drop:N \__enumext_item_text_vii_box
4670     \IfDocumentMetadataTF { }
4671     {
4672         \bool_if:NF \__enumext_footnotes_key_bool
4673         {
4674             \__enumext_print_footnote:
4675         }
4676     }
```

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`.

```
4677     \int_compare:nNnTF
4678     { \__enumext_item_column_pos_vii_int } = { \__enumext_columns_vii_int }
4679     {
4680         \par\noindent
4681         \int_zero:N \__enumext_item_column_pos_vii_int
4682     }
4683     {
4684         \skip_horizontal:N \__enumext_columns_sep_vii_dim
4685     }
4686 }
```

(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 `\removelastskip` 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)*.

```
4687 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4688 {
4689     \int_compare:nNnT
4690     {
4691         \int_mod:nn
4692         { \__enumext_item_count_all_vii_int } { \__enumext_columns_vii_int }
4693     }
4694     =
4695     { 0 }
4696     {
4697         \para_end:
4698         \skip_vertical:n { -\__enumext_itemsep_vii_skip }
4699         \skip_vertical:N \c_zero_skip
4700         \int_gzero:N \__enumext_item_count_all_vii_int
4701     }
4702 }
```

(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`.

```
4703 \__enumext_after_env:nn {enumext*}
4704 {
4705     \__enumext_execute_after_env:
4706 }
```

13.44 The environment keyans*

keyans*

First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii`: equal to `__enumext_first_item_tmp_viii`: and next to `\item` equal to `__enumext_start_item_tmp_viii`: which we will redefine later. The implementation of this environment is the same as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```

4707 \NewDocumentEnvironment{keyans*}{ o }
4708 {
4709   \__enumext_safe_exec_viii:
4710   \__enumext_parse_keys_viii:n {#1}
4711   \__enumext_before_list_viii:
4712   \__enumext_start_list:nn { }
4713   {
4714     \__enumext_list_arg_two_viii:
4715     \__enumext_before_keys_exec_viii:
4716   }
4717   \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4718   \__enumext_starred_columns_set_viii:
4719   \item[] \scan_stop:
4720   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4721   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4722   \ignorespaces
4723 }
4724 {
4725   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4726   \__enumext_stop_item_tmp_viii:
4727   \__enumext_remove_extra_parsep_viii:
4728   \__enumext_check_starred_cmd:n { item }
4729   \__enumext_after_list_viii:
4730 }

```

(End of definition for keyans*. This function is documented on page 14.)

__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.

```

4731 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4732 {
4733   \bool_if:NF \l__enumext_store_active_bool
4734   {
4735     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4736   }
4737   \int_incr:N \l__enumext_keyans_level_h_int
4738   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4739   {
4740     \msg_error:nn { enumext } { nested }
4741   }
4742   \__enumext_keyans_name_and_start:
4743   \bool_if:NT \l__enumext_starred_bool
4744   {
4745     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4746   }
4747   \bool_set_true:N \l__enumext_starred_bool
4748   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4749   \bool_set_false:N \l__enumext_store_active_bool
4750   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4751   {
4752     \msg_error:nn { enumext } { keyans-wrong-level }
4753   }
4754 }

```

(End of definition for __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

Parse [`<key = val>`] for `keyans*`.

```

4755 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1

```



```

4756 {
4757   \tl_if_novalue:nF {#1}
4758   {
4759     \keys_set:nn { enumext / keyans* } {#1}
4760   }
4761 }

```

(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, then call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

4762 \cs_new_protected:Nn \__enumext_before_list_viii:
4763 {
4764   \__enumext_vspace_above_viii:
4765   \__enumext_before_args_exec_viii:
4766   \__enumext_start_mini_viii:
4767 }

```

(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.

```

4768 \cs_new_protected:Nn \__enumext_after_list_viii:
4769 {
4770   \__enumext_stop_mini_viii:
4771   \__enumext_after_stop_list_viii:
4772   \__enumext_vspace_below_viii:
4773 }

```

(End of definition for `__enumext_after_list_viii:`.)

13.44.1 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.

```

4774 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4775 {
4776   \skip_horizontal:n { -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim }
4777 }

```

(End of definition for `__enumext_first_item_tmp_viii:`.)

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and 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`.

```

4778 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4779 {
4780   \__enumext_stop_item_tmp_viii:
4781   \int_incr:N \l__enumext_item_column_pos_viii_int
4782   \int_gincr:N \g__enumext_item_count_all_viii_int
4783   \__enumext_item_peek_args_viii:
4784 }

```

(End of definition for `__enumext_start_item_tmp_viii:`.)

`__enumext_item_peek_args_viii:` 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).

```

4785 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4786 {
4787   \peek_meaning:NTF (
4788     { \__enumext_joined_item_viii:w }
4789     { \__enumext_joined_item_viii:w (1) }
4790   }

```

(End of definition for `__enumext_item_peek_args_viii:.`)

`__enumext_joined_item_viii:w` 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`.

```

4791 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4792 {
4793   \__enumext_starred_joined_item_viii:n {#1}
4794   \peek_meaning_remove:NTF *
4795   { \__enumext_starred_item_viii:w }
4796   { \__enumext_standar_item_viii:w }
4797 }

```

(End of definition for `__enumext_joined_item_viii:w`.)

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

```

4798 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4799 {
4800   \bool_set_false:N \l__enumext_item_starred_viii_bool
4801   \peek_meaning:NTF [
4802   {
4803     \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4804     \__enumext_start_item_viii:w
4805   }
4806   {
4807     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4808     \legacy_if_set_true:n { @noitemarg }
4809     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4810   }
4811 }

```

(End of definition for `__enumext_standar_item_viii:w`.)

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

```

4812 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4813 {
4814   \bool_set_true:N \l__enumext_item_starred_viii_bool
4815   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4816   \peek_meaning:NTF [
4817   { \__enumext_starred_item_viii_aux_i:w }
4818   { \__enumext_starred_item_viii_aux_ii:w }
4819 }

```

The function `__enumext_starred_item_viii_aux_i:w` will save the *optional argument* to `\item*` in `\l__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_current_label_tl` if present, then call the function `__enumext_starred_item_viii_aux_ii:w`.

```

4820 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4821 {
4822   \tl_clear:N \l__enumext_store_current_label_tl
4823   \tl_if_no_value:nF { #1 }
4824   {

```

```

4825         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4826         {
4827             \tl_put_right:Ne \l__enumext_store_current_label_tl
4828             {
4829                 \l__enumext_store_keyans_item_opt_sep_tl
4830             }
4831             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4832         }
4833         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4834     }
4835     \__enumext_starred_item_viii_aux_ii:w
4836 }
4837 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4838 {
4839     \legacy_if_set_true:n { @noitemarg }
4840     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4841 }

```

(End of definition for `__enumext_starred_item_viii:w`, `__enumext_starred_item_viii_aux_i:w`, and `__enumext_starred_item_viii_aux_ii:w`.)

`__enumext_starred_item_exec:`

The function `__enumext_starred_item_exec:` will be in charge of storing the current *⟨label⟩* for *⟨item⟩** followed by the *⟨content⟩* for *⟨item⟩**[*⟨content⟩*] if present in the *sequence* and *prop list* set by the *save-ans* key. In this same function the keys *show-ans*, *show-pos* and *save-ref* are implemented.

```

4842 \cs_new_protected:Nn \__enumext_starred_item_exec:
4843 {
4844     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4845     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4846     \__enumext_keyans_store_ref:
4847     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4848     \__enumext_keyans_addto_seq_link:
4849     \int_gincr:N \g__enumext_check_starred_cmd_int
4850     \bool_if:NT \l__enumext_show_answer_bool
4851     {
4852         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4853     }
4854     \bool_if:NT \l__enumext_show_position_bool
4855     {
4856         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4857         {
4858             \group_begin:
4859             \exp_not:N \normalfont
4860             \exp_not:N \footnotesize [ \int_eval:n
4861                 {
4862                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4863                 }
4864             ]
4865             \group_end:
4866         }
4867         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4868     }
4869 }

```

(End of definition for `__enumext_starred_item_exec:`.)

`__enumext_fake_make_label_viii:n`

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

```

4870 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4871 {
4872     \legacy_if:nT { @noitemarg }
4873     {
4874         \legacy_if_set_false:n { @noitemarg }
4875         \legacy_if:nT { @nmbrlist }
4876         {
4877             \refstepcounter{enumXviii}
4878         }
4879     }
4880     \bool_if:NT \l__enumext_item_starred_viii_bool
4881     {
4882         \__enumext_starred_item_exec:
4883     }
4884     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]

```

```

4885     {
4886         \tl_use:N \l__enumext_label_font_style_viii_tl
4887         \bool_if:NTF \l__enumext_wrap_label_viii_bool
4888         {
4889             \__enumext_wrapper_label_viii:n {#1}
4890         }
4891         { #1 }
4892     }
4893     \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
4894 }

```

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

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

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

```

\__enumext_start_item_viii:w
\__enumext_stop_item_viii:
4895 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4896 {
4897     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4898     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4899     {
4900         \l__enumext_joined_width_viii_dim
4901         + \l__enumext_labelwidth_viii_dim
4902         + \l__enumext_labelsep_viii_dim
4903     }
4904     \IfDocumentMetadataTF { }
4905     {
4906         \bool_if:NF \l__enumext_footnotes_key_bool
4907         {
4908             \__enumext_renew_footnote:
4909         }
4910     }
4911     \__enumext_start_list_tag:n {keyans*}
4912     \__enumext_fake_make_label_viii:n {#1}
4913     \__enumext_stop_start_list_tag:
4914     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4915     \tl_use:N \l__enumext_after_list_args_viii_tl

```

The implementation of `\item*[\langle content \rangle]` with the `itemindent` and `wrap-opt` keys calls the function `__enumext_keyans_show_item_opt:` which shifts `itemindent` key to the right.

```

4916     \bool_if:NT \l__enumext_item_starred_viii_bool
4917     {
4918         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4919         \__enumext_keyans_show_item_opt:
4920         \skip_horizontal:n
4921         {
4922             -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim
4923         }
4924     }
4925     {
4926         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4927     }
4928     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4929     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4930 }
4931 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4932 {
4933     \__enumext_endminipage:
4934     \__enumext_stop_list_tag:n {keyans*}
4935     \hbox_set_end:
4936     \int_set:Nn \hbadness { 10000 }
4937     \box_use_drop:N \l__enumext_item_text_viii_box
4938     \IfDocumentMetadataTF { }
4939     {
4940         \bool_if:NF \l__enumext_footnotes_key_bool
4941         {
4942             \__enumext_print_footnote:
4943         }
4944     }
4945     \int_compare:nNnTF
4946     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4947     {

```

```

4948     \par\noindent
4949     \int_zero:N \l__enumext_item_column_pos_viii_int
4950   }
4951   {
4952     \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4953   }
4954 }

```

(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.

```

4955 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4956 {
4957   \int_compare:nNnT
4958   {
4959     \int_mod:nn
4960     { \g__enumext_item_count_all_viii_int }
4961     { \l__enumext_columns_viii_int }
4962   }
4963   =
4964   { 0 }
4965   {
4966     \para_end:
4967     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4968     \skip_vertical:N \c_zero_skip
4969     \int_gzero:N \g__enumext_item_count_all_viii_int
4970   }
4971 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:.`)

13.45 The command `\getkeyans`

`\getkeyans`
`__enumext_getkeyans_aux:n`
`__enumext_getkeyans:nn`

The `\getkeyans` command takes a *mandatory argument* of the form $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-ans` key.

```

4972 \NewDocumentCommand \getkeyans { m }
4973 {
4974   \exp_args:Ne \__enumext_getkeyans_aux:n
4975   { \tl_to_str:e { \text_expand:n {#1} } }
4976 }

```

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

```

4977 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4978 {
4979   \str_if_in:nnTF {#1} { : }
4980   {
4981     \use:e
4982     {
4983       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4984       { {##1} {##2} }
4985     }
4986     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4987   }
4988   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4989 }

```

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*.

```

4990 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4991 {
4992   \prop_if_exist:cTF { g__enumext_#1_prop }
4993   {
4994     \prop_item:cn { g__enumext_#1_prop } {#2}
4995   }
4996   {
4997     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4998   }
4999 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 17.)

13.46 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⟩]`.

```

5000 \keys_define:nn { enumext / print }
5001 {
5002   print* .code:n = \keys_precompile:neN { enumext / enumext* }
5003               { \__enumext_filter_save_key:n {#1} }
5004               \l__enumext_print_keyans_starred_tl, % starred cmd
5005   print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5006   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5007               { \__enumext_filter_save_key:n {#1} }
5008               \l__enumext_print_keyans_i_tl,
5009   print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5010   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5011               { \__enumext_filter_save_key:n {#1} }
5012               \l__enumext_print_keyans_ii_tl,
5013   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5014   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5015               { \__enumext_filter_save_key:n {#1} }
5016               \l__enumext_print_keyans_iii_tl,
5017   print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5018   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
5019               { \__enumext_filter_save_key:n {#1} }
5020               \l__enumext_print_keyans_iv_tl,
5021   print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5022   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
5023               { \__enumext_filter_save_key:n {#1} }
5024               \l__enumext_print_keyans_vii_tl, % starred nested
5025   print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5026 }

```

- 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`

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`.

`__enumext_printkeyans:nnn`

```

5027 \NewDocumentCommand \printkeyans { s O{ } m }
5028 {
5029   \group_begin:
5030     \tl_use:N \l__enumext_print_keyans_i_tl
5031     \tl_use:N \l__enumext_print_keyans_ii_tl
5032     \tl_use:N \l__enumext_print_keyans_iii_tl
5033     \tl_use:N \l__enumext_print_keyans_iv_tl
5034     \tl_use:N \l__enumext_print_keyans_vii_tl
5035     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5036   \group_end:
5037 }

```

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.

```

5038 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5039 {
5040   \seq_if_exist:cTF { g__enumext_#3_seq }
5041   {
5042     \seq_if_empty:cF { g__enumext_#3_seq }
5043   }

```

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.

```

5044         \bool_if:nTF {#1}
5045         {
5046             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5047             {
5048                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5049             }
5050         {
5051             \tl_use:N \l__enumext_print_keyans_starred_tl
5052             \bool_set_true:N \l__enumext_base_line_fix_bool
5053             \bool_set_true:N \l__enumext_print_keyans_star_bool
5054             \begin{enumext*}[#2]
5055                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5056             \end{enumext*}
5057             \bool_set_false:N \l__enumext_base_line_fix_bool
5058             \bool_set_false:N \l__enumext_print_keyans_star_bool
5059         }
5060     }

```

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

```

5061         {
5062             \begin{enumext}[#2]
5063             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5064             \end{enumext}
5065         }
5066     }
5067 }
5068 {
5069     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5070 }
5071 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 17.)

13.47 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`.

```

5072 \cs_new:Npn \__enumext_filter_first_level:n #1
5073 {
5074     \use:e
5075     {
5076         \keyval_parse:NNn
5077         \__enumext_filter_first_level_key:n
5078         \__enumext_filter_first_level_pair:nn {#1}
5079     }
5080 }

```

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*`.

```

5081 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5082 {
5083     \str_case:nnF {#1}
5084     {
5085         { resume } {}
5086         { resume* } {}
5087     }
5088     { , { \exp_not:n {#1} } }
5089 }

```

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.

```

5090 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5091 {
5092     \str_case:nnF {#1}
5093     {

```



```

5094     { series } {}
5095     { resume } {}
5096     { save-ans } {}
5097   }
5098   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5099 }

```

(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`.

```

5100 \keys_define:nn { enumext / meta-families }
5101 {
5102   enumext-1 .code:n =
5103     {
5104       \keys_set:ne { enumext / level-1 }
5105       {
5106         \__enumext_filter_first_level:n {#1}
5107       }
5108     } ,
5109   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5110   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5111   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5112   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5113   enumext* .code:n =
5114     {
5115       \keys_set:ne { enumext / enumext* }
5116       {
5117         \__enumext_filter_first_level:n {#1}
5118       }
5119     } ,
5120   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5121   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5122   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5123   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5124   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5125   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5126   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5127   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5128 }

```

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

```

5129 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5130 {
5131   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5132   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5133 }

```

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

```

5134 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5135 {
5136   \seq_clear:N \l__enumext_setkey_tmpa_seq
5137   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5138   \int_set:Nn \l__enumext_setkey_tmpa_int
5139   {
5140     \seq_count:N \l__enumext_setkey_tmpb_seq
5141   }
5142   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5143   {
5144     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5145     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5146     \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5147     {
5148       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5149     }
5150   }
5151   {
5152     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5153   }
5154   \seq_if_empty:NNTF \l__enumext_setkey_tmpa_seq
5155   { \seq_map_inline:Nn \c__enumext_all_families_seq }
5156   { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }

```

```

5157     {
5158         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5159     }
5160 }

```

(End of definition for `\setenumext`. This function is documented on page 6.)

```

__enumext_set_parse:n
__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

5161 \cs_new_protected:Npn __enumext_set_parse:n #1
5162 {
5163     \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5164     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5165     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5166     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5167     {
5168         \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
5169         { \tl_trim_spaces:n {#1} }
5170     }
5171     { __enumext_set_error:nn {#1} { } }
5172 }
5173 \cs_new_protected:Npn __enumext_set_error:nn #1 #2
5174 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `__enumext_set_parse:n` and `__enumext_set_error:nn`.)

13.48 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

```

```

5175 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5176 {
5177     {enumext,1} = level-1,
5178     {enumext,2} = level-2,
5179     {enumext,3} = level-3,
5180     {enumext,4} = level-4,
5181     {enumext*} = enumext*
5182 }

```

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

```

5183 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5184 {
5185     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5186     { \msg_error:nn { enumext } { prohibited-unknown } }
5187     {
5188         \bool_if:NTF {#1}
5189         {
5190             \int_step_inline:nn { 4 }
5191             { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5192             __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5193         }
5194         { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
5195     }
5196 }

```

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

```

5197 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
5198 {
5199     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5200     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5201     \prop_get:NVNTF
5202     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5203     { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5204     {
5205         \msg_error:nnn { enumext } { unknown-set } {#1}
5206         \use_none:nn
5207     }
5208 }
5209 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
5210 {

```

```

5211 \bool_lazy_or:nnTF
5212 { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5213 { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5214 { \msg_error:nnn { enumext } { already-defined } {#2} }
5215 {
5216   \keys_define:nn { enumext / #1 }
5217   {
5218     #2 .meta:n = {#3},
5219     #2 .value_forbidden:n = true
5220   }
5221 }
5222 }
5223 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

13.49 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](#).

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.

```

5224 \keys_define:nn { enumext / foreach }
5225 {
5226   before .tl_set:N = \l__enumext_foreach_before_tl,
5227   before .value_required:n = true,
5228   after .tl_set:N = \l__enumext_foreach_after_tl,
5229   after .value_required:n = true,
5230   start .int_set:N = \l__enumext_foreach_start_int,
5231   start .value_required:n = true,
5232   stop .int_set:N = \l__enumext_foreach_stop_int,
5233   stop .value_required:n = true,
5234   step .int_set:N = \l__enumext_foreach_step_int,
5235   step .value_required:n = true,
5236   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5237   wrapper .value_required:n = true,
5238   sep .tl_set:N = \l__enumext_foreach_sep_tl,
5239   sep .value_required:n = true,
5240   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5241 }
5242 \keys_precompile:nnN { enumext / foreach }
5243 {
5244   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5245 }
5246 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown *⟨keys⟩*.

```

5247 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5248 {
5249   \tl_if_blank:nTF {#2}
5250   {
5251     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5252   }
5253   {
5254     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5255   }
5256 }
5257 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5258 {
5259   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5260 }

```

We create the command.

```

5261 \NewDocumentCommand \foreachkeyans { +0{ } m }
5262 {
5263   \__enumext_foreach_keyans:nn {#1} {#2}
5264 }

```

Finally the internal functions `__enumext_foreach_keyans:nn` and `__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5265 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2

```

```

5266 {
5267   \tl_use:N \g__enumext_foreach_default_keys_tl
5268   \keys_set:nn { enumext / foreach } {#1}
5269   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5270   \prop_if_exist:cF { g__enumext_#2_prop }
5271     {
5272       \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5273     }
5274   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5275     {
5276       \int_set:Nn \l__enumext_foreach_stop_int
5277         { \prop_count:c { g__enumext_#2_prop } }
5278     }
5279   \seq_clear:N \l__enumext_foreach_print_seq
5280   \int_step_function:nnnN
5281     { \l__enumext_foreach_start_int }
5282     { \l__enumext_foreach_step_int }
5283     { \l__enumext_foreach_stop_int }
5284     \__enumext_foreach_add_body:n
5285     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5286   }
5287 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5288 {
5289   \seq_put_right:Ne \l__enumext_foreach_print_seq
5290     {
5291       \exp_not:V \l__enumext_foreach_before_tl
5292       \__enumext_foreach_wrapper:n
5293       {
5294         \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5295       }
5296       \exp_not:V \l__enumext_foreach_after_tl
5297     }
5298 }

```

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

13.50 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5299 \msg_new:nnn { enumext } { package-load }
5300 {
5301   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5302 }
5303 \msg_new:nnn { enumext } { package-not-load }
5304 {
5305   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5306 }
5307 \msg_new:nnn { enumext } { package-load-foot }
5308 {
5309   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5310 }

```

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

```

5311 \msg_new:nnn { enumext } { counters }
5312 {
5313   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5314   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5315 }

```

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

```

5316 \msg_new:nnn { enumext } { unknown-choice }
5317 {
5318   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5319 }

```

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

```

5320 \msg_new:nnnn { enumext } { anskey-env-error }
5321 {
5322   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
5323   'enumext' ~ package, ~ It~ is~ already~ defined.
5324 }
5325 {
5326   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~

```

```

5327     for ~ the ~ 'save-ans' ~ key.\\
5328 }

```

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

```

5329 \msg_new:nnn { enumext } { store-prop }
5330 {
5331   * ~ Package ~ enumext: ~ Creating ~
5332   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5333 }
5334 \msg_new:nnn { enumext } { store-seq }
5335 {
5336   * ~ Package ~ enumext: ~ Creating ~
5337   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5338 }
5339 \msg_new:nnn { enumext } { store-int }
5340 {
5341   * ~ Package ~ enumext: ~ Creating ~
5342   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5343 }
5344 \msg_new:nnn { enumext } { prop-seq-int-hook }
5345 {
5346   * ~ Package ~ enumext: ~ Elements ~ in ~
5347   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5348   * ~ Package ~ enumext: ~ Elements ~ in ~
5349   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5350   * ~ Package ~ enumext: ~ Value ~ off ~
5351   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5352 }
5353 \msg_new:nnn { enumext } { item-answer-hook }
5354 {
5355   * ~ Package ~ enumext: ~ Value ~ off ~
5356   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5357   * ~ Package ~ enumext: ~ Value ~ off ~
5358   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5359   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5360 }

```

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

```

5361 \msg_new:nnn { enumext } { invalid-key }
5362 {
5363   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5364 }
5365 \msg_new:nnn { enumext } { unknown-key-family }
5366 {
5367   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5368 }

```

Messages used in length calculation.

```

5369 \msg_new:nnn { enumext } { width-negative }
5370 {
5371   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5372   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
5373 }
5374 \msg_new:nnn { enumext } { width-zero }
5375 {
5376   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5377   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
5378 }

```

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

```

5379 \msg_new:nnn { enumext } { list-lengths }
5380 {
5381   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5382   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5383   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5384   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5385   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5386   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5387   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5388   \__enumext_show_length:nnn { skip } { topsep } {#1}
5389   \__enumext_show_length:nnn { skip } { parsep } {#1}
5390   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5391   \__enumext_show_length:nnn { skip } { itemsep } {#1}

```

```

5392      *****
5393    }

```

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

```

5394 \msg_new:nnn { enumext } { list-lengths-not-nested }
5395 {
5396   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5397   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5398   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5399   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5400   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5401   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5402   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5403   \__enumext_show_length:nnn { skip } { topsep } {#1}
5404   \__enumext_show_length:nnn { skip } { parsep } {#1}
5405   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5406   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5407   *****
5408 }

```

Messages used by `ref` key.

```

5409 \msg_new:nnn { enumext } { key-ref-empty }
5410 {
5411   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5412 }

```

Messages used by `save-ans` key.

```

5413 \msg_new:nnn { enumext } { save-ans-empty }
5414 {
5415   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5416 }
5417 \msg_new:nnn { enumext } { save-ans-log }
5418 {
5419   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5420 }
5421 \msg_new:nnn { enumext } { save-ans-log-hook }
5422 {
5423   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5424 }
5425 \msg_new:nnn { enumext } { save-ans-hook }
5426 {
5427   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5428 }

```

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

```

5429 \msg_new:nnn { enumext } { need-save-ans }
5430 {
5431   Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5432 }
5433 \msg_new:nnn { enumext } { items-same-answer }
5434 {
5435   *****\\
5436   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5437   for ~ \c_left_brace_str #2 \c_right_brace_str\\
5438   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5439   'OK', ~ all ~ items ~ with ~ answer.\\
5440   *****
5441 }
5442 \msg_new:nnn { enumext } { item-greater-answer }
5443 {
5444   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5445   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5446   Items ~ > ~ Answers.
5447 }
5448 \msg_new:nnn { enumext } { item-less-answer }
5449 {
5450   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5451   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5452   Items ~ < ~ Answers.
5453 }

```

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

```

5454 \msg_new:nnn { enumext } { missing-starred }

```

```

5455 {
5456     Missing ~ '\c_backslash_str #1*' ~ #2.
5457 }
5458 \msg_new:nnn { enumext } { many-starred }
5459 {
5460     Many ~ '\c_backslash_str #1*' ~ #2.
5461 }

```

Messages used by `\printkeyans*` command.

```

5462 \msg_new:nnn { enumext } { print-starred }
5463 {
5464     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5465     #2 ~ environment ~ \msg_line_context:.
5466 }

```

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

```

5467 \msg_new:nnn { enumext } { list-too-deep }
5468 {
5469     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5470     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5471 }

```

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

```

5472 \msg_new:nnn { enumext } { anskey-unnumber-item }
5473 {
5474     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5475 }
5476 \msg_new:nnn { enumext } { anskey-already-stored }
5477 {
5478     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5479 }
5480 \msg_new:nnn { enumext } { anskey-empty-arg }
5481 {
5482     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5483 }
5484 \msg_new:nnn { enumext } { anskey-wrong-place }
5485 {
5486     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5487     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5488 }
5489 \msg_new:nnn { enumext } { anskey-nested }
5490 {
5491     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5492 }
5493 \msg_new:nnn { enumext } { anskey-math-mode }
5494 {
5495     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5496 }
5497 \msg_new:nnn { enumext } { anskey-env-wrong }
5498 {
5499     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5500 }
5501 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5502 {
5503     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5504     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5505 }
5506 \msg_new:nnn { enumext } { command-wrong-place }
5507 {
5508     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5509     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5510 }
5511 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5512 {
5513     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5514     'anskey*' ~ and ~ is ~ being ~ ignored.
5515 }
5516 {
5517     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5518     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5519 }
5520 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }

```



```

5521 {
5522     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5523     'anskey*' ~ and ~ is ~ being ~ ignored.
5524 }
5525 {
5526     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5527     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5528 }
5529 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5530 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored. }
5531 {
5532     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5533     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5534 }
5535 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5536 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored. }
5537 {
5538     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5539     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5540 }

```

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

```

5541 \msg_new:nnn { enumext } { keyans-nested }
5542 {
5543     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5544 }
5545 \msg_new:nnn { enumext } { keyans-wrong-level }
5546 {
5547     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5548     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5549 }
5550 \msg_new:nnn { enumext } { wrong-place }
5551 {
5552     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5553     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5554 }
5555 \msg_new:nnn { enumext } { keyanspic-nested }
5556 {
5557     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5558 }
5559 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5560 {
5561     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5562     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5563 }
5564 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5565 {
5566     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5567 }
5568 \msg_new:nnnn { enumext } { keyans-unknown-key }
5569 {
5570     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5571     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5572 }
5573 {
5574     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5575     ~ have ~ a ~ key ~ called ~'#1'.\\
5576     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5577 }
5578 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5579 {
5580     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5581     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5582 }
5583 {
5584     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5585     ~ have ~ a ~ key ~ called ~'#1'.\\
5586     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5587 }

```

Message used by unknown `<keys>` in `enumext*` environment.

```

5588 \msg_new:nnnn { enumext } { starred-unknown-key }

```

```

5589 {
5590     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5591     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5592 }
5593 {
5594     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5595     ~ have ~ a ~ key ~ called ~'#1'.\\
5596     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5597 }
5598 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5599 {
5600     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5601     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5602 }
5603 {
5604     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5605     ~ have ~ a ~ key ~ called ~'#1'.\\
5606     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5607 }

```

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

```

5608 \msg_new:nnnn { enumext } { standar-unknown-key }
5609 {
5610     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5611     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5612 }
5613 {
5614     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5615     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5616     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5617 }
5618 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5619 {
5620     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5621     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5622 }
5623 {
5624     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5625     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5626     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5627 }

```

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

```

5628 \msg_new:nnnn { enumext } { for-key-unknown }
5629 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5630 {
5631     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5632     Check~that~you~have~spelled~the~key~name~correctly.
5633 }
5634 \msg_new:nnnn { enumext } { for-key-value-unknown }
5635 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5636 {
5637     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5638     Check~that~you~have~spelled~the~key~name~correctly.
5639 }

```

Messages used by **\getkeyans** command.

```

5640 \msg_new:nnn { enumext } { undefined-storage-anskey }
5641 {
5642     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5643 }

```

Messages used by **\miniright** command.

```

5644 \msg_new:nnn { enumext } { missing-miniright }
5645 {
5646     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5647     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5648 }
5649 \msg_new:nnn { enumext } { wrong-miniright-place }
5650 {
5651     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5652     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5653 }

```

```

5654 \msg_new:nnn { enumext } { wrong-miniright-use }
5655 {
5656   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
5657   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5658 }
5659 \msg_new:nnn { enumext } { wrong-miniright-starred }
5660 {
5661   Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5662 }
5663 \msg_new:nnn { enumext } { many-miniright-used }
5664 {
5665   Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5666 }

```

Messages used by `\setenumextmeta` command.

```

5667 \msg_new:nnn { enumext } { unknown-set }
5668 {
5669   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5670 }
5671 \msg_new:nnn { enumext } { already-defined }
5672 {
5673   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5674 }
5675 \msg_new:nnn { enumext } { prohibited-unknown }
5676 {
5677   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5678 }

```

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

```

5679 \msg_new:nnn { enumext } { nested }
5680 {
5681   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5682 }
5683 \msg_new:nnn { enumext } { nested-horizontal }
5684 {
5685   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
5686 }
5687 \msg_new:nnn { enumext } { item-joined }
5688 {
5689   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5690 }
5691 \msg_new:nnn { enumext } { item-joined-columns }
5692 {
5693   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5694 }

```

13.51 Finish package

Finish package implementation.

```

5695 \file_input_stop:
5696 </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>	228
<code>\+</code>	220
<code>\-</code>	220
<code>\\</code> 236, 2771, 4089, 5313, 5322, 5327, 5347, 5349, 5356, 5358, 5371, 5376, 5381, 5396, 5435, 5437, 5439, 5444, 5445, 5450, 5451, 5469, 5486, 5503, 5508, 5517, 5526, 5532, 5538, 5547, 5552, 5561, 5575, 5585, 5595, 5605, 5615, 5625, 5631, 5637, 5646, 5651, 5656	
A	
above	<u>1585</u>
above*	<u>1585</u>
<code>\addvspace</code> 1155, 1183, 1226, 1229, 1397, 1400, 1497, 1503, 1538, 1544, 1565, 1571, 3580, 3741, 3759, 3990, 3993, 4323, 4338, 4384, 4398	
after	<u>985</u>
align	<u>536</u>
<code>\Alph</code>	38, <u>43</u>
<code>\Alph</code>	488, 606, 650, 716, 5021
<code>\alph</code>	38, <u>43</u>
<code>\alph</code>	489, 604, 5013
<code>\anskey</code>	12, 77, 79, <u>2589</u>
anskey*	13, <u>2699</u>
<code>\anspic</code>	15, 105, 108, <u>3997</u>
<code>\anspic*</code>	71
<code>\arabic</code>	32, 38
<code>\arabic</code>	487, 603, 649, 5005, 5009, 5025
B	
base-fix	<u>845</u>
<code>\baselineskip</code>	52
<code>\baselineskip</code>	861, 868
before	<u>985</u>
before*	<u>985</u>
below	<u>1585</u>
below*	<u>1585</u>
bool commands:	
<code>\bool_gset_false:N</code> 357, 358, 359, 2875, 2877, 4340, 4344, 4400	
<code>\bool_gset_true:N</code> 265, 275, 1088, 2078, 2084, 4309, 4341, 4373, 4401	
<code>\bool_if:NTF</code> . 427, 439, 456, 1519, 1607, 1621, 1634, 1645, 1656, 1667, 1678, 1689, 1738, 1755, 1760, 1768, 1795, 1833, 1838, 1845, 1849, 1871, 1876, 1884, 1891, 1922, 1930, 2023, 2221, 2231, 2310, 2334, 2341, 2365, 2463, 2485, 2525, 2539, 2543, 2593, 2612, 2636, 2690, 2701, 2790, 2827, 2891, 2926, 2941, 3016, 3027, 3031, 3050, 3063, 3105, 3139, 3182, 3201, 3344, 3359, 3421, 3431, 3464, 3469, 3535, 3561, 3611, 3669, 3724, 3749, 3924, 3988, 3999, 4018, 4063, 4302, 4318, 4324, 4367, 4381, 4385, 4507, 4517, 4601, 4607, 4614, 4630, 4649, 4672, 4733, 4743, 4850, 4854, 4880, 4887, 4906, 4916, 4940	
<code>\bool_if:nTF</code> 1545, 1572, 3161, 3318, 3379, 3903, 4039, 5044, 5188	
<code>\bool_if_p:N</code> 284, 299, 855, 856, 864, 865, 1902, 1903, 1911, 1912, 2036, 2062, 2075, 2076, 2081, 2082, 2398, 2408, 2420, 2435, 2436, 2470, 2511, 2512, 2813, 3003, 3004, 3041, 3042, 3508, 3510, 3521, 4046, 4047	
<code>\bool_lazy_all:nTF</code> 282, 297, 853, 2034, 2060, 2396, 2405, 2418, 2433, 3506, 3519	
<code>\bool_lazy_and:nnTF</code> 261, 271, 863, 1512, 1901, 1910, 2074, 2080, 2469, 2476, 2510, 2654, 2666, 2812, 2818, 3002	
<code>\bool_lazy_or:nnTF</code> . . 1963, 1970, 3040, 4045, 5211	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 97, 102, 103, 108, 109, 112, 131, 138, 139, 146, 153, 154, 159, 161, 162, 176, 188, 190	
<code>\bool_not_p:n</code> 262, 272, 857, 2407, 2471, 2477, 2814, 2819, 3509, 3522	
<code>\bool_set_eq:NN</code> 3114, 3297, 4554, 4803	
<code>\bool_set_false:N</code> 436, 875, 2008, 2009, 2041, 2046, 2050, 2054, 2067, 2754, 3483, 3628, 3677, 3764, 3921, 3995, 4476, 4502, 4551, 4749, 4800, 5057, 5058	
<code>\bool_set_true:N</code> . 289, 290, 304, 305, 416, 420, 529, 890, 1591, 1596, 1858, 1980, 1981, 2253, 2261, 2755, 3108, 3110, 3142, 3144, 3293, 3305, 3444, 3482, 3515, 3528, 3601, 3674, 3701, 3905, 4291, 4356, 4475, 4558, 4565, 4566, 4610, 4747, 4807, 4814, 4815, 5052, 5053	
box commands:	
<code>\box_dp:N</code> . . 1443, 1444, 1447, 1454, 1467, 1475, 1481, 1489, 3934, 3939, 3990, 4074	
<code>\box_ht:N</code> . . 1226, 1229, 1240, 1241, 1252, 1254, 1269, 1272, 1280, 1281, 1292, 1294, 1309, 1312, 1319, 1320, 1331, 1333, 1348, 1351, 1397, 1400, 1408, 1409, 1417, 1418, 1430, 1432	
<code>\box_ht_plus_dp:N</code> 3930, 4027	
<code>\box_new:N</code> 70, 149, 150, 183, 189	
<code>\box_use_drop:N</code> 4335, 4396, 4669, 4937	
<code>\box_wd:N</code> 495	
C	
<code>\c</code>	228, 229, 752, 754, 766, 768
<code>\catcode</code>	2771
<code>\cB</code>	229
<code>\cE</code>	229
<code>\centering</code>	1547, 1574, 4111, 4328, 4389
check-ans	<u>2000</u>
Document class:	
article	45
clist commands:	
<code>\clist_const:Nn</code>	195
<code>\clist_map_function:nN</code>	4094
<code>\clist_map_inline:Nn</code> 535, 800, 984, 999, 1080, 1601	
<code>\clist_map_inline:nn</code> . 49, 60, 78, 86, 99, 111, 141, 170, 194, 566, 586, 895, 916, 1094, 1707, 1947, 2014, 2200, 2218, 2250, 2393, 2935, 3222, 3234, 3274, 3408, 3411, 3439, 3451, 3454, 3474, 5164	
<code>\columnbreak</code>	77
<code>\columnbreak</code>	2473
columns	<u>1064</u>
columns-sep	<u>1064</u>
<code>\columnsep</code>	99
<code>\columnsep</code>	3556, 3722
<code>\columnseprule</code>	99
<code>\columnseprule</code>	3559, 3723

Commands provide by **enumext**:

\anskey 30, 67, 68, 73, 74, 76, 78, 79, 86, 88, 98, 117, 127, 128, 136
 \anspic* 30, 31, 71, 74, 86, 87, 107, 108, 127, 128
 \anspic 30, 74, 105–108, 136
 \foreachkeyans 132, 138
 \getkeyans 74, 127, 138
 \item* 30, 31, 71, 74, 86, 87, 89, 90, 93, 119, 124, 125, 127, 128
 \item 89, 93, 111, 118, 120, 123, 124
 \miniright 29, 50, 58, 59, 100, 138
 \printkeyans* 128
 \printkeyans 30, 74, 128
 \setenumextmeta 131, 139
 \setenumext 30, 128, 130, 131, 134

Counters defined by **enumext**:

enumXiii 28, 38
 enumXii 28, 38
 enumXiv 28, 38
 enumXi 28, 38
 enumXviii 28, 38
 enumXvii 28, 38, 119
 enumXvi 28, 38
 enumXv 28, 38

cs commands:

\cs_generate_variant:Nn . 200, 201, 497, 513, 758, 774, 2302, 2307, 2383, 2707, 3398, 4096, 5223
 \cs_if_exist:NTF 467
 \cs_if_free:NTF 2658, 2670
 \cs_new:Nn 214
 \cs_new:Npn . 232, 1708, 1717, 1725, 2265, 2274, 2282, 5072, 5081, 5090
 \cs_new_eq:NN . 384, 385, 390, 391, 441, 442, 445, 446
 \cs_new_protected:Nn . 224, 238, 254, 280, 313, 343, 349, 355, 361, 367, 375, 393, 411, 626, 687, 738, 851, 1000, 1004, 1008, 1012, 1016, 1020, 1024, 1028, 1032, 1036, 1040, 1044, 1048, 1052, 1056, 1060, 1095, 1107, 1140, 1157, 1168, 1185, 1211, 1232, 1357, 1383, 1403, 1436, 1458, 1493, 1499, 1602, 1616, 1630, 1641, 1652, 1663, 1674, 1685, 1766, 1869, 1882, 1899, 1920, 1948, 1953, 1978, 2019, 2029, 2072, 2087, 2094, 2103, 2108, 2113, 2118, 2127, 2132, 2137, 2308, 2332, 2339, 2363, 2370, 2384, 2610, 2629, 2645, 2708, 2744, 2775, 2810, 2852, 2873, 2881, 2924, 2939, 2967, 3000, 3036, 3048, 3061, 3147, 3157, 3168, 3176, 3192, 3314, 3330, 3338, 3352, 3475, 3504, 3533, 3540, 3570, 3587, 3609, 3631, 3667, 3691, 3708, 3733, 3747, 3768, 3919, 4084, 4092, 4097, 4121, 4152, 4281, 4300, 4346, 4365, 4405, 4409, 4428, 4462, 4489, 4496, 4505, 4515, 4536, 4687, 4731, 4762, 4768, 4785, 4842, 4955
 \cs_new_protected:Npn 202, 206, 210, 449, 465, 482, 492, 498, 607, 651, 721, 745, 759, 1529, 1558, 1734, 1753, 1823, 1856, 1958, 2142, 2219, 2229, 2251, 2259, 2294, 2303, 2459, 2522, 2537, 2575, 2579, 2699, 2730, 2734, 2765, 2901, 2977, 3021, 3101, 3120, 3235, 3239, 3253, 3257, 3275, 3279, 3289, 3301, 3367, 3401, 3442, 3486, 3687, 3895, 3912, 4016, 4035, 4059, 4183, 4232, 4479, 4542, 4549, 4563, 4571, 4576, 4586, 4755, 4791, 4798, 4812, 4820, 4837, 4977, 4990, 5038, 5161, 5173, 5197, 5209, 5247, 5257, 5265, 5287
 \cs_new_protected_nopar:Nn . . . 3829, 3871, 3879, 3887, 4525, 4529, 4663, 4774, 4778, 4931
 \cs_new_protected_nopar:Npn . . 3821, 3837, 4592, 4638, 4870, 4895
 \cs_set:Npn 2394, 2431, 4983

\cs_set_eq:NN . . 4452, 4453, 4640, 4720, 4721, 4897
 \cs_set_protected:Nn 921, 937, 950, 963
 \cs_set_protected:Npn 45, 54, 71, 79, 94, 100, 134, 166, 174, 514, 536, 571, 587, 633, 775, 801, 877, 900, 976, 985, 1064, 1081, 1585, 1696, 1939, 2000, 2159, 2201, 2237, 2386, 2928, 3211, 3227, 3267, 3399, 3440
 \cs_to_str:N 484, 507
 \cs_undefine:N 2647, 2648, 2649, 2650

D

\d 220
 \DeclareDocumentEnvironment 397
 dim commands:
 \dim_abs:n 3372, 3377
 \dim_add:Nn 3938, 4146, 4177
 \dim_compare:nNnTF . 923, 939, 952, 965, 1244, 1256, 1284, 1296, 1323, 1335, 1412, 1420, 1531, 1560, 3369, 3374, 3380, 3386, 3388, 3390, 3545, 3592, 3695, 3712, 3914, 4123, 4139, 4154, 4170, 4283, 4348
 \dim_compare:nTF 2495, 2840, 3634, 3771
 \dim_eval:n 861, 4070
 \dim_gset_eq:NN 4292, 4357
 \dim_gzero:N 2879, 4343, 4403
 \dim_new:N . 67, 74, 75, 76, 96, 143, 151, 152, 182, 184, 185, 191
 \dim_set:Nn . . 495, 891, 3137, 3372, 3377, 3379, 3382, 3383, 3387, 3389, 3392, 3393, 3395, 3548, 3595, 3633, 3697, 3714, 3770, 3928, 4025, 4099, 4125, 4132, 4156, 4163, 4218, 4267, 4285, 4350, 4588
 \dim_set_eq:NN 594, 640, 709, 713, 3052, 3053, 3065, 3066, 3132, 3410, 3453, 3556, 3722, 4225, 4228, 4229, 4274, 4277, 4278, 4581, 4660, 4928
 \dim_sub:Nn 3639, 3776, 4141, 4172
 \dim_use:N 924, 932, 1532, 1542, 2373, 2376, 2381, 3152, 3154, 3197, 3546, 3550, 3551, 3553, 3593, 3598, 3599, 3605, 3636, 3641
 \dim_zero:N 3445, 3559, 3723, 3940
 \dim_zero_new:N 464
 \c_zero_dim 926, 940, 953, 966, 1532, 1560, 2497, 2842, 3369, 3374, 3380, 3387, 3546, 3593, 3636, 3695, 3712, 3773, 3914, 4123, 4139, 4154, 4170, 4283, 4348
 \dimeval 2166

E

\end . . . 2336, 2367, 3577, 3738, 3980, 4113, 5046, 5056, 5064
 end internal commands:
 \end__enumext__mini_page . 1540, 1567, 3620, 3758, 4307, 4371, 4397
 \endgroup 2771
 \endlist 385
 \endminipage 391
 enumext 5, 3645
 enumext internal commands:
 \l__enumext__ref_the_count_tl 40
 \l__enumext__resume_name_tl 63
 __enumext_add_meta_key:nnn . . . 131, 5175, 5191, 5192, 5194, 5197
 __enumext_add_pre_parsep: . 51, 1105, 1107, 1107
 __enumext_after_args_exec: 48, 1000, 1012, 3658
 __enumext_after_args_exec_v: 1016, 1028, 3791
 __enumext_after_args_exec_vii: . . 1032, 1056
 __enumext_after_args_exec_viii: 1060
 __enumext_after_env:nn 83–85, 101, 113, 121, 206, 206, 2785, 3663, 4316, 4379, 4703
 __enumext_after_hyperref: . . . 36, 409, 411, 411
 \l__enumext_after_list_args_v_tl 1030

```

\l__enumext_after_list_args_vii_tl 1058, 4658
\l__enumext_after_list_args_viii_tl .. 1062,
    4915
\__enumext_after_list_vii: 113, 117, 4460, 4496,
    4496
\__enumext_after_list_viii: ... 123, 4729, 4768,
    4768
\__enumext_after_stop_list: 48, 100, 1000, 1008,
    3625
\__enumext_after_stop_list_v: 1016, 1024, 3765
\l__enumext_after_stop_list_v_tl ..... 1026
\__enumext_after_stop_list_vii: .. 117, 1032,
    1048, 4499
\l__enumext_after_stop_list_vii_tl ... 1050
\__enumext_after_stop_list_viii: . 1052, 4771
\l__enumext_after_stop_list_viii_tl ... 1054
\l__enumext_align_label_pos_v_str .... 3356
\l__enumext_align_label_pos_X_str ..... 79
\l__enumext_align_label_vii_str ..... 4627
\l__enumext_align_label_viii_str ..... 4884
\l__enumext_align_label_X_str ..... 174
\c__enumext_all_envs_clist .. 195, 535, 800, 984,
    999, 1080, 1601
\c__enumext_all_families_seq .. 130, 5129, 5155
\l__enumext_anskey_env_bool 33, 82, 34, 290, 305,
    2701
\__enumext_anskey_env_clean_vars: . 85, 2806,
    2810, 2873
\__enumext_anskey_env_define_keys: 82, 2699,
    2708, 2779
\__enumext_anskey_env_exec: 83, 2704, 2775, 2775
\__enumext_anskey_env_make:n 67, 82, 1983, 2699,
    2699, 2707
\__enumext_anskey_env_reset_keys: 83, 84, 2744,
    2807
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2699
\__enumext_anskey_env_save_keys: .. 84, 2787,
    2810, 2810
\__enumext_anskey_env_store: .. 84, 2803, 2810,
    2852
\__enumext_anskey_env_unknown:n 82, 2727, 2730
\__enumext_anskey_env_unknown:nn . 2732, 2734
\l__enumext_anskey_level_int .. 28, 2631, 2632
\__enumext_anskey_safe_inner:.. 80, 2604, 2610,
    2629
\__enumext_anskey_safe_inner:n ..... 80
\__enumext_anskey_safe_outer: . 79, 2591, 2610,
    2610
\__enumext_anskey_show_wrap_arg:n . 78, 2522,
    2522, 2541, 2556
\__enumext_anskey_show_wrap_left:n 78, 2467,
    2537, 2537
\__enumext_anskey_unknown:n 79, 2559, 2573, 2575
\__enumext_anskey_unknown:nn . 2559, 2577, 2579
\__enumext_anskey_wrapper:n ..... 2163, 2535
\l__enumext_anspic_above_int . 142, 4100, 4101,
    4103
\__enumext_anspic_args:nnn 108, 109, 4013, 4084,
    4084
\l__enumext_anspic_args_seq 107-110, 142, 3973,
    4011, 4112
\l__enumext_anspic_below_int . 142, 4100, 4101,
    4104
\l__enumext_anspic_body_box ... 142, 4024, 4027
\__enumext_anspic_body_dim:n .. 108, 4016, 4016,
    4062
\l__enumext_anspic_body_htdp_dim .. 108, 142,
    4025, 4073
\__enumext_anspic_label:nn 108, 4035, 4035, 4065,
    4079
\l__enumext_anspic_label_box .. 142, 3927, 3930
\l__enumext_anspic_label_htdp_dim . 106, 142,
    3928, 3934, 4072
\__enumext_anspic_label_pos:nnn .. 109, 4059,
    4059, 4087
\l__enumext_anspic_mini_pos_str 106, 142, 3906,
    3909, 4110
\l__enumext_anspic_mini_width_dim 142, 4037,
    4099, 4110
\__enumext_anspic_print:n 109, 110, 3973, 3975,
    4092, 4092, 4096
\__enumext_anspic_row:n .. 110, 4092, 4094, 4097
\__enumext_anspic_start_list_tag: 3845, 3871,
    4086
\__enumext_anspic_stop_list_tag: . 3845, 3887,
    4090
\__enumext_anspic_stop_start_list_tag: 3845,
    3879, 4088
\__enumext_at_begin_document:n .. 35, 202, 202,
    382, 388
\l__enumext_base_line_fix_bool 45, 46, 128, 129,
    847, 856, 875, 5052, 5057
\__enumext_before_args_exec: . 48, 99, 117, 1000,
    1000, 3590
\__enumext_before_args_exec_v: 1016, 1016, 3694
\__enumext_before_args_exec_vii: . 1032, 1032,
    4493
\__enumext_before_args_exec_viii: 1036, 4765
\__enumext_before_env:nn 82, 206, 210, 2652, 2664,
    2676, 2777
\__enumext_before_keys_exec: .. 48, 1000, 1004,
    3655
\__enumext_before_keys_exec_v: 1016, 1020, 3788
\__enumext_before_keys_exec_vii ..... 1032
\__enumext_before_keys_exec_vii: . 1040, 4447
\__enumext_before_keys_exec_viii: 1044, 4715
\__enumext_before_list: ... 99, 3587, 3587, 3649
\__enumext_before_list_v: ... 3691, 3691, 3783
\__enumext_before_list_vii: ... 117, 4442, 4489,
    4489
\__enumext_before_list_viii: .. 123, 4711, 4762,
    4762
\l__enumext_before_no_starred_key_v_tl 1022
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1042
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1046
\l__enumext_before_starred_key_v_tl ... 1018
\l__enumext_before_starred_key_vii_tl . 1034
\l__enumext_before_starred_key_viii_tl 1038
\__enumext_calc_hspace:NNNNNNN 95, 3367, 3367,
    3398, 3403, 3446
\__enumext_check_ans_active: . 68, 99, 117, 2019,
    2019, 3591, 4492
\g__enumext_check_ans_item_tl ..... 87
\g__enumext_check_ans_key_bool 69, 70, 153, 357,
    2078, 2084, 2891
\l__enumext_check_ans_key_bool 69, 2004, 2009,
    2075, 2081

```


__enumext_check_ans_key_hook: . . 69, 100, 117,
 2072, 2072, 3626, 4500
 __enumext_check_ans_level: . 68, 69, 2019, 2025,
 2029
 __enumext_check_ans_log: 70, 85, 2118, 2118, 2895
 __enumext_check_ans_log_msg_greater: 2118,
 2124, 2137
 __enumext_check_ans_log_msg_less: 2118, 2122,
 2127
 __enumext_check_ans_log_msg_same_ok: 2118,
 2123, 2132
 __enumext_check_ans_msg_greater: 2094, 2100,
 2113
 __enumext_check_ans_msg_less: 2094, 2098, 2103
 __enumext_check_ans_msg_same_ok: 2094, 2099,
 2108
 __enumext_check_ans_show: . . 70, 85, 2094, 2094,
 2893
 \l__enumext_check_answers_bool 67, 68, 79, 89, 90,
 153, 1981, 2008, 2023, 2310, 2334, 2341, 2365, 2593,
 2790, 3016, 3105, 3139, 4607
 __enumext_check_starred_cmd:n 34, 71, 87, 122,
 2142, 2142, 3794, 3986, 4728
 \g__enumext_check_starred_cmd_int . . 94, 153,
 2145, 2151, 2156, 3312, 4044, 4849
 \l__enumext_check_start_line_env_tl . 34, 153,
 320, 328, 336, 2148, 2154, 2157
 \l__enumext_columns_sep_v_dim 3712, 3714, 3722
 \l__enumext_columns_sep_vii_dim . . 4123, 4125,
 4134, 4146, 4222, 4684
 \l__enumext_columns_sep_viii_dim . 4154, 4156,
 4165, 4177, 4271, 4952
 \l__enumext_columns_v_int 1377, 1395, 1563, 3710,
 3718, 3730, 3735
 \l__enumext_columns_vii_int . . 4128, 4131, 4135,
 4144, 4186, 4190, 4193, 4199, 4205, 4209, 4678, 4692
 \l__enumext_columns_viii_int . 4159, 4162, 4166,
 4175, 4235, 4239, 4242, 4248, 4254, 4258, 4946, 4961
 \l__enumext_counter_i_tl 45, 474
 \l__enumext_counter_ii_tl 45, 475
 \l__enumext_counter_iii_tl 45, 476
 \l__enumext_counter_iv_tl 45, 477
 \c__enumext_counter_style_tl 32, 50, 226
 \g__enumext_counter_styles_tl. 28, 38, 67, 485,
 503
 \l__enumext_counter_v_tl 45, 478, 729
 \l__enumext_counter_vi_tl 45, 479
 \l__enumext_counter_vii_tl 45, 480, 661
 \l__enumext_counter_viii_tl 45, 481, 677
 \l__enumext_current_widest_dim 28, 67, 509, 595,
 641, 710, 714
 __enumext_def_meta_key:nnn . . . 131, 5175, 5203,
 5209, 5223
 __enumext_default_item:n . . . 3101, 3101, 3165
 __enumext_define_counters:Nn 28, 465, 465, 474,
 475, 476, 477, 478, 479, 480, 481
 __enumext_endminipage: . 36, 382, 391, 405, 4337,
 4665, 4933
 \g__enumext_envir_name_tl 33, 34, 291, 306, 365,
 1951, 1956, 1966, 2106, 2111, 2116, 2130, 2135, 2140
 \l__enumext_envir_name_tl . 33, 34, 34, 260, 270,
 319, 327, 335, 5571, 5574, 5581, 5584, 5591, 5594,
 5601, 5604, 5610, 5614, 5620, 5624, 5681, 5685
 __enumext_execute_after_env: 35, 66, 70, 81, 85,
 2881, 2881, 3665, 4705
 __enumext_fake_item_indent: . . 921, 921, 3430
 \l__enumext_fake_item_indent_v_dim 940, 945
 \l__enumext_fake_item_indent_v_tl 942, 3294,
 3298, 3306
 __enumext_fake_item_indent_vii: . . 921, 950,
 3463
 \l__enumext_fake_item_indent_vii_dim 953, 958
 \l__enumext_fake_item_indent_vii_tl 955, 4659
 __enumext_fake_item_indent_viii: . 921, 963,
 3468
 \l__enumext_fake_item_indent_viii_dim . 966,
 971, 4922
 \l__enumext_fake_item_indent_viii_tl . . 968,
 4918, 4926
 \l__enumext_fake_item_indent_X_tl 100
 __enumext_fake_make_label_vii:n . 119, 4592,
 4592, 4655
 __enumext_fake_make_label_viii:n 4870, 4870,
 4912
 __enumext_filter_first_level:n . . 129, 5072,
 5072, 5106, 5117
 __enumext_filter_first_level_key:n 129, 5072,
 5077, 5081
 __enumext_filter_first_level_pair:nn . 129,
 5072, 5078, 5090
 __enumext_filter_save_key:n . . 73, 2226, 2234,
 2257, 2263, 2265, 2265, 5003, 5007, 5011, 5015, 5019,
 5023
 __enumext_filter_save_key_key:n . . 73, 2265,
 2270, 2274
 __enumext_filter_save_key_pair:nn 74, 2265,
 2271, 2282
 __enumext_filter_series:n 62, 1708, 1708, 1746,
 1758, 1763
 __enumext_filter_series_key:n 62, 1708, 1713,
 1717
 __enumext_filter_series_pair:nn . . 62, 1708,
 1714, 1725
 __enumext_first_item_tmp_vii: 116, 118, 4452,
 4525, 4525
 __enumext_first_item_tmp_viii: 122, 123, 4720,
 4774, 4774
 \g__enumext_footnote_arg_seq . 171, 4411, 4424,
 4434
 \g__enumext_footnote_int . 171, 4418, 4421, 4423,
 4425
 \g__enumext_footnote_int_seq . 171, 4412, 4425,
 4430, 4433
 __enumext_footnotes_key_bool 36
 \l__enumext_footnotes_key_bool 31, 37, 120, 161,
 420, 427, 436, 4649, 4672, 4906, 4940
 __enumext_footnotetext:nn . . . 4405, 4405, 4435
 __enumext_foreach_add_body:n . 132, 5224, 5284,
 5287
 \l__enumext_foreach_after_tl 5228, 5296
 \l__enumext_foreach_before_tl 5226, 5291
 \g__enumext_foreach_default_keys_tl 132, 126,
 5246, 5267
 __enumext_foreach_keyans:nn . . 132, 5224, 5263,
 5265
 \l__enumext_foreach_name_prop_tl . 126, 5269,
 5294
 \l__enumext_foreach_print_seq 126, 5279, 5285,
 5289


```

\l__enumext_foreach_sep_tl . . . . . 5238, 5285
\l__enumext_foreach_start_int . . . . 5230, 5281
\l__enumext_foreach_step_int . . . . 5234, 5282
\l__enumext_foreach_stop_int . 5232, 5274, 5276,
    5283
\__enumext_foreach_wrapper:n . . . . . 5236, 5292
\__enumext_getkeyans:nn . . 127, 4972, 4986, 4990
\__enumext_getkeyans_aux:n 127, 4972, 4974, 4977
\l__enumext_hyperref_bool . 31, 36, 37, 161, 416,
    439, 456, 2512, 3004, 4601
\__enumext_hypertarget:nn 37, 411, 441, 445, 461
\__enumext_if_is_int:n . . . . . 218
\__enumext_if_is_int:nTF . . . . . 218, 747, 761
\__enumext_internal_mini_page: 36, 97, 116, 393,
    393, 3477, 4464
\__enumext_is_not_nested: 28, 33, 97, 116, 254, 254,
    3478, 4465
\__enumext_is_on_first_level: . 28, 33, 97, 116,
    254, 280, 3484, 4477
\g__enumext_item_anskey_int 79, 88, 153, 352, 379,
    380, 2091, 2461, 3018
\__enumext_item_answer_diff: 70, 85, 2087, 2087,
    2888
\g__enumext_item_answer_diff_int 70, 153, 353,
    2089, 2096, 2120
\l__enumext_item_column_pos_vii_int 118, 4193,
    4199, 4205, 4209, 4216, 4532, 4678, 4681
\l__enumext_item_column_pos_viii_int . 123,
    4242, 4248, 4254, 4258, 4265, 4781, 4946, 4949
\l__enumext_item_column_pos_X_int . . . . . 174
\g__enumext_item_count_all_vii_int 118, 4217,
    4533, 4692, 4700
\g__enumext_item_count_all_viii_int 123, 4266,
    4782, 4960, 4969
\g__enumext_item_count_all_X_int . . . . . 174
\g__enumext_item_number_bool . . . . . 153
\l__enumext_item_number_bool 69, 159, 2041, 2046,
    2050, 2054, 2067, 2636, 2690, 3108, 3142, 4610
\g__enumext_item_number_int . . 69, 153, 351, 378,
    380, 2040, 2045, 2049, 2053, 2066, 2091, 3107, 3141,
    4609
\__enumext_item_peek_args_vii: 118, 4534, 4536,
    4536
\__enumext_item_peek_args_viii: 123, 124, 4783,
    4785, 4785
\__enumext_item_star_exec: 90, 3120, 3147, 3184,
    3203
\l__enumext_item_starred_vii_bool 4551, 4565,
    4614
\l__enumext_item_starred_viii_bool 4800, 4814,
    4880, 4916
\l__enumext_item_starred_X_bool . . . . . 174
\__enumext_item_std:w 36, 89, 90, 93, 382, 386, 3111,
    3117, 3145, 3294, 3298, 3306
\g__enumext_item_symbol_aux_tl . 90, 130, 3125,
    3128, 3153, 3189, 3207
\g__enumext_item_symbol_aux_vii_tl 4573, 4616,
    4619, 4623, 4625
\g__enumext_item_symbol_aux_X_tl . . . . . 174
\l__enumext_item_symbol_sep_vii_dim . 4581,
    4588, 4622, 4624
\l__enumext_item_symbol_vii_tl . . . . . 4619
\l__enumext_item_text_vii_box . . . 4641, 4669
\l__enumext_item_text_viii_box . . 4898, 4937
\l__enumext_item_text_X_box . . . . . 174
\l__enumext_item_width_vii_dim . . 4132, 4141,
    4220, 4228, 4229
\l__enumext_item_width_viii_dim . 4163, 4172,
    4269, 4277, 4278
\l__enumext_item_width_X_dim . . . . . 174
\l__enumext_itemindent_X_dim . . . . . 71
\l__enumext_itemsep_i_skip . . 1238, 1245, 1248,
    1250, 1257, 1261, 1264, 1266, 1406, 1413, 1415, 1416,
    1421, 1425, 1427, 1428
\l__enumext_itemsep_ii_skip . 1278, 1285, 1288,
    1290, 1297, 1301, 1304, 1306
\l__enumext_itemsep_iii_skip . 1317, 1324, 1327,
    1329, 1336, 1340, 1343, 1345
\l__enumext_itemsep_vii_skip . . . . . 4698
\l__enumext_itemsep_viii_skip . . . . . 4967
\l__enumext_joined_item_aux_vii_int . 4214,
    4215, 4216, 4217, 4223
\l__enumext_joined_item_aux_viii_int . 4263,
    4264, 4265, 4266, 4272
\l__enumext_joined_item_aux_X_int . . . . 174
\__enumext_joined_item_vii:w . 118, 4539, 4540,
    4542, 4542
\l__enumext_joined_item_vii_int . 4185, 4186,
    4189, 4191, 4197, 4202, 4207, 4212, 4214, 4220
\__enumext_joined_item_viii:w 124, 4788, 4789,
    4791, 4791
\l__enumext_joined_item_viii_int . 4234, 4235,
    4238, 4240, 4246, 4251, 4256, 4261, 4263, 4269
\l__enumext_joined_item_X_int . . . . . 174
\l__enumext_joined_width_vii_dim . 4218, 4225,
    4228, 4643, 4657
\l__enumext_joined_width_viii_dim 4267, 4274,
    4277, 4900, 4914
\l__enumext_joined_width_X_dim . . . . . 174
\__enumext_keyans_addto_prop:n 86, 2901, 2901,
    3309, 4041
\__enumext_keyans_addto_seq:n . 87, 2977, 2977,
    3311, 4043
\__enumext_keyans_addto_seq_link: 2977, 2998,
    3000, 4848
\__enumext_keyans_default_item:n . 93, 3289,
    3289, 3326
\l__enumext_keyans_env_bool 34, 3509, 3522, 3674,
    3764
\__enumext_keyans_fake_item_indent: 921, 937,
    3420
\l__enumext_keyans_level_h_int . 122, 28, 670,
    696, 2620, 2682, 2955, 4471, 4737, 4738
\l__enumext_keyans_level_int . . 28, 1523, 2616,
    2678, 2950, 3673, 3678, 4007
\__enumext_keyans_make_label: 39, 94, 3330, 3330,
    3418
\__enumext_keyans_make_label_box: 3330, 3334,
    3352
\__enumext_keyans_make_label_std: 3330, 3336,
    3338
\__enumext_keyans_mini_right_cmd:n 59, 1525,
    1558, 1558
\__enumext_keyans_mini_set_vskip: . . . . . 55
\__enumext_keyans_minipage_add_space: 1357,
    1383, 3703
\__enumext_keyans_minipage_set_skip: . 1357,
    1357, 1385

```

__enumext_keyans_multi_addvspace: [1157](#), [1168](#), [3727](#)
 __enumext_keyans_multi_set_vskip: [52](#), [1157](#), [1157](#), [1170](#)
 __enumext_keyans_multicols_start: [3691](#), [3706](#), [3708](#)
 __enumext_keyans_multicols_stop: [1562](#), [3691](#), [3733](#), [3762](#)
 __enumext_keyans_name_and_start: [28](#), [34](#), [122](#), [313](#), [313](#), [3675](#), [3902](#), [4742](#)
 __enumext_keyans_parse_keys:n [3687](#), [3687](#), [3782](#)
 __enumext_keyans_pic_arg_two: [106](#), [3919](#), [3919](#), [3947](#)
 \l__enumext_keyans_pic_level_int .. [28](#), [1507](#), [2624](#), [2686](#), [2904](#), [2945](#), [2980](#), [3068](#), [3897](#), [3898](#)
 \g__enumext_keyans_pic_parsep_skip [142](#), [3936](#), [3993](#)
 __enumext_keyans_pic_safe_exec:n [106](#), [3895](#), [3895](#), [3946](#)
 __enumext_keyans_pic_skip_abs:N .. [106](#), [3912](#), [3912](#), [3923](#)
 \l__enumext_keyans_pic_star_bool .. [106](#), [142](#), [3905](#), [3924](#), [3988](#), [4018](#), [4063](#)
 __enumext_keyans_pre_itemsep_skip: .. [1357](#), [1376](#), [1403](#)
 __enumext_keyans_redefine_item: .. [94](#), [3314](#), [3314](#), [3417](#)
 __enumext_keyans_ref: [43](#), [721](#), [738](#), [3419](#)
 __enumext_keyans_ref:n [43](#), [718](#), [721](#), [721](#)
 __enumext_keyans_safe_exec: . [3667](#), [3667](#), [3781](#)
 __enumext_keyans_set_item_width: [103](#), [3768](#), [3768](#), [3790](#)
 __enumext_keyans_show_ans: .. [3021](#), [3029](#), [3048](#)
 __enumext_keyans_show_item_opt: [93](#), [126](#), [3021](#), [3036](#), [3307](#), [4056](#), [4919](#)
 __enumext_keyans_show_left:n . [93](#), [3021](#), [3021](#), [3304](#), [4050](#)
 __enumext_keyans_show_pos: .. [3021](#), [3033](#), [3061](#)
 __enumext_keyans_starred_item:n .. [93](#), [3301](#), [3301](#), [3322](#)
 __enumext_keyans_store_ref: .. [86](#), [2924](#), [2924](#), [3310](#), [4042](#), [4846](#)
 __enumext_keyans_store_ref_aux_i: [86](#), [2924](#), [2936](#), [2939](#)
 __enumext_keyans_store_ref_aux_ii: [87](#), [2924](#), [2965](#), [2967](#)
 __enumext_keyans_unknown_keys:n . [3227](#), [3231](#), [3235](#)
 __enumext_keyans_unknown_keys:nn [3227](#), [3237](#), [3239](#)
 __enumext_keyans_wrapper_opt:n .. [2169](#), [3044](#)
 \l__enumext_label_copy_i_tl .. [2427](#), [2943](#), [2948](#), [2953](#), [2958](#)
 \l__enumext_label_copy_v_tl [2953](#)
 \l__enumext_label_copy_vi_tl [2948](#)
 \l__enumext_label_copy_vii_tl [2403](#), [2414](#), [2443](#), [2943](#)
 \l__enumext_label_copy_viii_tl [2958](#)
 \l__enumext_label_copy_X_tl [163](#)
 \l__enumext_label_fill_left_v_tl [3342](#)
 \l__enumext_label_fill_left_X_tl [100](#)
 \l__enumext_label_fill_right_v_tl [3349](#)
 \l__enumext_label_fill_right_X_tl [100](#)
 \l__enumext_label_font_style_v_tl [3343](#), [3358](#), [4054](#)
 \l__enumext_label_font_style_vii_tl ... [4629](#)
 \l__enumext_label_font_style_viii_tl .. [4886](#)
 \l__enumext_label_i_tl [587](#)
 \l__enumext_label_ii_tl [587](#)
 \l__enumext_label_iii_tl [587](#)
 \l__enumext_label_iv_tl [587](#)
 __enumext_label_style:Nnn [28](#), [38](#), [498](#), [498](#), [513](#), [592](#), [638](#), [707](#), [711](#)
 \l__enumext_label_v_tl [87](#), [704](#), [2909](#), [2985](#), [3055](#), [3095](#), [3303](#), [3308](#), [3785](#), [3927](#), [4049](#), [4051](#)
 \l__enumext_label_vi_tl [87](#), [704](#), [2906](#), [2982](#), [4049](#), [4051](#), [4055](#)
 \l__enumext_label_vii_tl . [633](#), [4560](#), [4583](#), [4590](#)
 \l__enumext_label_viii_tl [633](#), [4809](#), [4840](#), [4844](#)
 \l__enumext_label_width_by_box .. [67](#), [494](#), [495](#)
 __enumext_label_width_by_box:Nn [38](#), [492](#), [492](#), [497](#), [509](#), [771](#)
 \l__enumext_labelsep_i_dim ... [3053](#), [3058](#), [3066](#), [3098](#), [4852](#), [4867](#)
 \l__enumext_labelsep_v_dim [3717](#)
 \l__enumext_labelsep_vii_dim . [2528](#), [3053](#), [3066](#), [4127](#), [4137](#), [4221](#), [4527](#), [4581](#), [4636](#), [4645](#)
 \l__enumext_labelsep_viii_dim [4158](#), [4168](#), [4270](#), [4776](#), [4893](#), [4902](#), [4922](#)
 \l__enumext_labelwidth_i_dim . [3052](#), [3058](#), [3065](#), [3098](#), [4852](#), [4867](#)
 \l__enumext_labelwidth_v_dim [3356](#), [3717](#)
 \l__enumext_labelwidth_vii_dim ... [2528](#), [3052](#), [3065](#), [4127](#), [4136](#), [4221](#), [4527](#), [4627](#), [4644](#)
 \l__enumext_labelwidth_viii_dim .. [4158](#), [4167](#), [4270](#), [4776](#), [4884](#), [4901](#)
 \l__enumext_leftmargin_tmp_v_bool . [106](#), [3921](#)
 \l__enumext_leftmargin_tmp_X_bool [71](#)
 \l__enumext_leftmargin_tmp_X_dim [71](#)
 \l__enumext_leftmargin_X_dim [71](#)
 __enumext_level: [214](#), [214](#), [616](#), [619](#), [620](#), [628](#), [630](#), [924](#), [928](#), [932](#), [1002](#), [1006](#), [1010](#), [1014](#), [1097](#), [1099](#), [1101](#), [1103](#), [1145](#), [1147](#), [1149](#), [1151](#), [1155](#), [1189](#), [1195](#), [1200](#), [1202](#), [1205](#), [1208](#), [1221](#), [1224](#), [1532](#), [1536](#), [1542](#), [1605](#), [1607](#), [1609](#), [1612](#), [1619](#), [1621](#), [1623](#), [1626](#), [2221](#), [2223](#), [2225](#), [2253](#), [2254](#), [2256](#), [2312](#), [2320](#), [2324](#), [2328](#), [2532](#), [2533](#), [3110](#), [3111](#), [3115](#), [3116](#), [3117](#), [3125](#), [3133](#), [3134](#), [3137](#), [3144](#), [3145](#), [3149](#), [3152](#), [3154](#), [3180](#), [3181](#), [3182](#), [3185](#), [3188](#), [3197](#), [3198](#), [3200](#), [3201](#), [3204](#), [3515](#), [3528](#), [3535](#), [3543](#), [3546](#), [3548](#), [3550](#), [3551](#), [3552](#), [3553](#), [3556](#), [3561](#), [3567](#), [3573](#), [3580](#), [3593](#), [3595](#), [3598](#), [3599](#), [3601](#), [3605](#), [3611](#), [3636](#), [3641](#), [3652](#), [3654](#)
 \l__enumext_level_h_int [116](#), [28](#), [263](#), [286](#), [300](#), [654](#), [689](#), [1514](#), [2037](#), [2057](#), [2422](#), [2656](#), [2668](#), [3523](#), [4466](#), [4467](#)
 \l__enumext_level_int . [97](#), [28](#), [216](#), [273](#), [285](#), [301](#), [395](#), [1109](#), [1234](#), [1513](#), [2031](#), [2063](#), [2399](#), [2409](#), [2415](#), [2421](#), [2428](#), [2437](#), [2442](#), [2655](#), [2667](#), [2883](#), [3434](#), [3479](#), [3480](#), [3491](#), [3499](#), [3513](#), [3526](#), [3557](#), [3682](#), [4003](#), [4509](#), [4519](#), [4750](#), [5611](#), [5615](#), [5621](#), [5625](#)
 __enumext_list_arg_two_i: [3399](#)
 __enumext_list_arg_two_ii: [3399](#)
 __enumext_list_arg_two_iii: [3399](#)
 __enumext_list_arg_two_iv: [3399](#)
 __enumext_list_arg_two_v: . [94](#), [3399](#), [3787](#), [3922](#)
 __enumext_list_arg_two_vii: [3440](#), [4446](#)
 __enumext_list_arg_two_viii: [3440](#), [4714](#)
 \l__enumext_listoffset_v_dim . [3719](#), [3773](#), [3776](#)
 \l__enumext_listparindent_vii_dim [4660](#)

`\l__enumext_listparindent_viii_dim` ... 4928
`__enumext_log_answer_vars:` . 35, 367, 375, 2890
`__enumext_log_global_vars:` . 35, 367, 367, 2889
`__enumext_make_label:` . 39, 91, 3168, 3168, 3428
`__enumext_make_label_box:` ... 3168, 3172, 3192
`__enumext_make_label_std:` ... 3168, 3174, 3176
`\l__enumext_mark_answer_sym_tl` 75, 2175, 2378, 2545, 3070, 3083, 4856
`\l__enumext_mark_position_str` 130, 2179, 2180, 2206, 2207, 2376
`\l__enumext_mark_ref_sym_tl` .. 2192, 2517, 3012
`\l__enumext_meta_path_tl` . 126, 5199, 5200, 5202, 5203
`\c__enumext_meta_paths_prop` 131, 5175
`__enumext_mini_addvspace_vii:` 57, 1493, 1493, 4295
`__enumext_mini_addvspace_viii:` 57, 1493, 1499, 4360
`__enumext_mini_env*` 393
`__enumext_mini_page` 1542, 1569, 3605, 3704, 4297, 4362, 4383
`__enumext_mini_right_cmd:n` . 58, 59, 1527, 1529, 1529
`__enumext_mini_set_vskip_vii:` 56, 1436, 1436, 1495
`__enumext_mini_set_vskip_viii:` 56, 1436, 1458, 1501
`__enumext_minipage:w` 36, 382, 390, 399, 4320, 4657, 4914
`\l__enumext_minipage_active_v_bool` 3701, 3724, 3749
`\g__enumext_minipage_active_vii_bool` .. 113, 4309, 4318, 4340
`\l__enumext_minipage_active_vii_bool` . 4291, 4302
`\g__enumext_minipage_active_viii_bool` 4373, 4381, 4400
`\l__enumext_minipage_active_viii_bool` 4356, 4367
`\g__enumext_minipage_active_X_bool` ... 174
`\l__enumext_minipage_active_X_bool` 87
`__enumext_minipage_add_space:` . 53, 100, 1185, 1211, 3603
`\g__enumext_minipage_after_skip` 87, 1440, 1452, 4338, 4398
`\l__enumext_minipage_after_skip` .. 52, 100, 87, 1198, 1238, 1240, 1245, 1248, 1252, 1257, 1261, 1264, 1268, 1280, 1285, 1288, 1292, 1297, 1301, 1304, 1308, 1319, 1324, 1327, 1331, 1336, 1340, 1343, 1347, 1359, 1373, 1406, 1408, 1413, 1415, 1417, 1421, 1425, 1427, 1429, 1460, 1473, 1487, 1538, 1565, 3759
`\g__enumext_minipage_center_vii_bool` . 4324, 4341
`\g__enumext_minipage_center_viii_bool` 4385, 4401
`\g__enumext_minipage_center_X_bool` ... 174
`\l__enumext_minipage_hsep_v_dim` 3699
`\l__enumext_minipage_hsep_vii_dim` 4289
`\l__enumext_minipage_hsep_viii_dim` ... 4354
`\l__enumext_minipage_left_skip` 87, 1360, 1438, 1443, 1447, 1461, 1465, 1479, 1497, 1503
`\l__enumext_minipage_left_v_dim` .. 3697, 3704
`\l__enumext_minipage_left_vii_dim` 4285, 4297
`\l__enumext_minipage_left_viii_dim` 4350, 4362
`\l__enumext_minipage_left_X_dim` 87
`\g__enumext_minipage_right_skip` 87, 1439, 1444, 1448, 4323, 4384
`\l__enumext_minipage_right_skip` . 52, 87, 1187, 1193, 1198, 1200, 1202, 1361, 1362, 1368, 1373, 1374, 1375, 1380, 1462, 1469, 1483, 1544, 1571
`\l__enumext_minipage_right_v_dim` . 1560, 1569, 3695, 3699
`\g__enumext_minipage_right_vii_dim` 113, 4293, 4320, 4343
`\l__enumext_minipage_right_vii_dim` 113, 4283, 4288, 4294
`\g__enumext_minipage_right_viii_dim` .. 4358, 4383, 4403
`\l__enumext_minipage_right_viii_dim` .. 4348, 4353, 4359
`\g__enumext_minipage_right_X_dim` 174
`\g__enumext_minipage_right_X_skip` 174
`__enumext_minipage_set_skip:` . 52, 1185, 1185, 1213
`\g__enumext_minipage_stat_int` .. 100, 87, 1549, 1576, 3602, 3613, 3618, 3702, 3751, 3756
`\l__enumext_minipage_temp_skip` 87, 1259, 1269, 1272, 1299, 1309, 1312, 1338, 1348, 1351, 1423, 1430, 1432
`\l__enumext_miniright_code_vii_box` 4331, 4335
`\g__enumext_miniright_code_vii_tl` 114, 4326, 4333, 4342
`\l__enumext_miniright_code_viii_box` .. 4392, 4396
`\g__enumext_miniright_code_viii_tl` 4387, 4394, 4402
`\l__enumext_miniright_code_X_box` 174
`__enumext_multi_addvspace:` . 51, 99, 1140, 1140, 3564
`__enumext_multi_set_vskip:` 50, 1095, 1095, 1142
`\l__enumext_multicols_above_ii_skip` ... 1114
`\l__enumext_multicols_above_iii_skip` .. 1123
`\l__enumext_multicols_above_iv_skip` ... 1132
`\l__enumext_multicols_above_v_skip` 1159, 1173, 1183, 1374
`\l__enumext_multicols_above_X_skip` 79
`\l__enumext_multicols_below_ii_skip` .. 1241, 1250, 1254, 1266, 1271
`\l__enumext_multicols_below_iii_skip` . 1281, 1290, 1294, 1306, 1311
`\l__enumext_multicols_below_iv_skip` .. 1320, 1329, 1333, 1345, 1350
`\l__enumext_multicols_below_v_skip` 1163, 1177, 1375, 1409, 1416, 1418, 1428, 1431, 3741
`\l__enumext_multicols_below_X_skip` 79
`\g__enumext_multicols_right_X_skip` 79
`__enumext_multicols_start:` 99, 100, 3540, 3540, 3607
`__enumext_multicols_stop:` 99, 1534, 3570, 3570, 3623
`__enumext_nested_base_line_fix:` . 45, 98, 851, 851, 3495
`__enumext_newlabel:nn` 31, 37, 77, 449, 449, 2453, 2971
`\l__enumext_newlabel_arg_one_tl` 31, 37, 77, 86, 163, 2446, 2454, 2516, 2960, 2972, 3010
`\l__enumext_newlabel_arg_two_tl` 31, 37, 76, 163, 2402, 2412, 2425, 2440, 2455, 2947, 2952, 2957, 2973

`__enumext_parse_foreach_keys:n` . . . [5224](#), [5240](#), [5257](#)
`__enumext_parse_foreach_keys:nn` . [5224](#), [5247](#), [5259](#)
`__enumext_parse_keys:n` [45](#), [62](#), [3486](#), [3486](#), [3648](#)
`__enumext_parse_keys_vii:n` [62](#), [4441](#), [4479](#), [4479](#)
`__enumext_parse_keys_viii:n` . [4710](#), [4755](#), [4755](#)
`__enumext_parse_save_key:n` [73](#), [2246](#), [2251](#), [2251](#)
`__enumext_parse_save_key_vii:n` [73](#), [2241](#), [2251](#), [2259](#)
`__enumext_parse_series:n` [62](#), [98](#), [116](#), [1734](#), [1734](#), [3494](#), [4485](#)
`__enumext_parse_store_keys:n` [98](#)
`\l__enumext_parsep_i_skip` [1112](#), [1116](#)
`\l__enumext_parsep_ii_skip` [1121](#), [1125](#)
`\l__enumext_parsep_iii_skip` [1130](#), [1134](#)
`\l__enumext_parsep_vii_skip` [4661](#)
`\l__enumext_parsep_viii_skip` [4929](#)
`\l__enumext_partopsep_v_skip` . [1175](#), [1179](#), [1370](#), [1393](#)
`\l__enumext_partopsep_viii_skip` [1471](#)
`__enumext_phantomsection:` [37](#), [411](#), [442](#), [446](#), [462](#)
`__enumext_pre_itemsep_skip:` . . [53](#), [1203](#), [1232](#), [1232](#)
`__enumext_print_footnote:` . . . [4405](#), [4428](#), [4674](#), [4942](#)
`__enumext_print_keyans_box:NN` [75](#), [2370](#), [2370](#), [2383](#), [2527](#), [2531](#), [3057](#), [3097](#), [4852](#), [4867](#)
`\l__enumext_print_keyans_i_tl` [5008](#), [5030](#)
`\l__enumext_print_keyans_ii_tl` . . . [5012](#), [5031](#)
`\l__enumext_print_keyans_iii_tl` . . [5016](#), [5032](#)
`\l__enumext_print_keyans_iv_tl` . . . [5020](#), [5033](#)
`\l__enumext_print_keyans_star_bool` . [45](#), [128](#), [129](#), [130](#), [857](#), [865](#), [5053](#), [5058](#)
`\l__enumext_print_keyans_starred_tl` [128](#), [130](#), [5004](#), [5051](#)
`\l__enumext_print_keyans_vii_tl` [128](#), [5024](#), [5034](#)
`\l__enumext_print_keyans_X_tl` [130](#)
`__enumext_printkeyans:nnn` [128](#), [5027](#), [5035](#), [5038](#)
`__enumext_redefine_item:` . [90](#), [3157](#), [3157](#), [3427](#)
`\l__enumext_ref_key_arg_tl` . [40](#), [41](#), [50](#), [229](#), [609](#), [610](#), [622](#), [653](#), [656](#), [666](#), [672](#), [682](#), [723](#), [724](#), [734](#)
`\l__enumext_ref_the_count_tl` . [41](#), [50](#), [616](#), [619](#), [622](#), [661](#), [663](#), [666](#), [677](#), [679](#), [682](#), [729](#), [731](#), [734](#)
`__enumext_regex_counter_style:` . . [32](#), [40](#), [224](#), [224](#), [617](#), [662](#), [678](#), [730](#)
`__enumext_register_counter_style:Nn` . . [482](#), [482](#), [487](#), [488](#), [489](#), [490](#), [491](#)
`__enumext_remove_extra_parsep_vii:` . . [4459](#), [4687](#), [4687](#)
`__enumext_remove_extra_parsep_viii:` . [4727](#), [4955](#), [4955](#)
`__enumext_renew_footnote:` . . . [4405](#), [4409](#), [4651](#), [4908](#)
`\l__enumext_renew_the_count_v_tl` [732](#), [740](#), [742](#)
`\l__enumext_renew_the_count_vii_tl` [664](#), [691](#), [693](#)
`\l__enumext_renew_the_count_viii_tl` [680](#), [698](#), [700](#)
`\l__enumext_renew_the_count_X_tl` [50](#)
`__enumext_rescan_anskey_env:n` . . [83](#), [84](#), [2765](#), [2860](#), [2868](#)
`__enumext_reset_global_bool:` . . [343](#), [346](#), [355](#)
`__enumext_reset_global_int:` . . . [343](#), [345](#), [349](#)
`__enumext_reset_global_tl:` [343](#), [347](#), [361](#)
`__enumext_reset_global_vars:` . [35](#), [85](#), [343](#), [343](#), [2898](#)
`\l__enumext_resume_active_bool` [62](#), [64](#), [61](#), [1738](#), [1858](#)
`__enumext_resume_counter:` . . [64](#), [65](#), [1856](#), [1862](#), [1869](#)
`__enumext_resume_counter:n` . [62](#), [64](#), [1827](#), [1832](#), [1856](#), [1856](#), [1926](#), [1934](#)
`__enumext_resume_counter_save_ans:` [65](#), [1856](#), [1867](#), [1899](#)
`__enumext_resume_counter_series:` . [65](#), [1856](#), [1865](#), [1882](#)
`\g__enumext_resume_int` . . . [61](#), [1779](#), [1873](#), [1874](#)
`__enumext_resume_last:n` [62](#), [63](#), [1734](#), [1740](#), [1753](#)
`\l__enumext_resume_name_tl` [61](#), [1775](#), [1783](#), [1786](#), [1802](#), [1810](#), [1813](#), [1859](#), [1860](#), [1888](#), [1895](#)
`__enumext_resume_save_counter:` . [63](#), [100](#), [117](#), [1766](#), [1766](#), [3629](#), [4503](#)
`__enumext_resume_series:n` . [64](#), [1702](#), [1823](#), [1823](#)
`__enumext_resume_starred:` . [66](#), [1703](#), [1920](#), [1920](#)
`\g__enumext_resume_vii_int` [61](#), [1806](#), [1878](#), [1879](#)
`\l__enumext_rightmargin_vii_dim` . . [4139](#), [4143](#), [4148](#)
`\l__enumext_rightmargin_viii_dim` . [4170](#), [4174](#), [4179](#)
`__enumext_safe_exec:` . . [36](#), [97](#), [3475](#), [3475](#), [3647](#)
`__enumext_safe_exec_vii:` . [36](#), [4440](#), [4462](#), [4462](#)
`__enumext_safe_exec_viii:` [122](#), [4709](#), [4731](#), [4731](#)
`__enumext_second_part:` . . [100](#), [3609](#), [3609](#), [3661](#)
`__enumext_second_part_v:` . . . [3691](#), [3747](#), [3795](#)
`\l__enumext_series_name_tl` [64](#), [65](#)
`\l__enumext_series_str` . . [63](#), [98](#), [116](#), [1700](#), [1736](#), [1744](#), [1745](#), [1747](#), [1749](#), [1770](#), [1773](#), [1777](#), [1797](#), [1800](#), [1804](#), [3490](#), [4483](#)
`__enumext_set_error:nn` [5161](#), [5171](#), [5173](#)
`__enumext_set_item_width:` [101](#), [3631](#), [3631](#), [3657](#)
`__enumext_set_parse:n` [5145](#), [5161](#), [5161](#)
`\l__enumext_setkey_tmpa_int` . . . [121](#), [5138](#), [5142](#)
`\l__enumext_setkey_tmpa_seq` . . [121](#), [5136](#), [5146](#), [5152](#), [5154](#), [5156](#), [5168](#)
`\l__enumext_setkey_tmpa_tl` [121](#), [5144](#), [5148](#)
`\l__enumext_setkey_tmpb_seq` . . [121](#), [5137](#), [5140](#), [5144](#), [5145](#)
`\l__enumext_setkey_tmpb_tl` [121](#), [5163](#), [5165](#), [5166](#)
`\l__enumext_show_answer_bool` . [2186](#), [2210](#), [2539](#), [3027](#), [3041](#), [4046](#), [4850](#)
`__enumext_show_length:nnn` . . [47](#), [232](#), [232](#), [5382](#), [5383](#), [5384](#), [5385](#), [5386](#), [5387](#), [5388](#), [5389](#), [5390](#), [5391](#), [5397](#), [5398](#), [5399](#), [5400](#), [5401](#), [5402](#), [5403](#), [5404](#), [5405](#), [5406](#)
`\l__enumext_show_position_bool` . . [2189](#), [2213](#), [2543](#), [3031](#), [3042](#), [4047](#), [4854](#)
`\g__enumext_standar_bool` [33](#), [97](#), [34](#), [262](#), [265](#), [284](#), [358](#), [1768](#), [1833](#), [1845](#), [1871](#), [1884](#), [1922](#), [2062](#), [2076](#), [2407](#), [2420](#), [2435](#), [3510](#)
`\l__enumext_standar_bool` [97](#), [100](#), [34](#), [2408](#), [3482](#), [3628](#), [4476](#)
`\l__enumext_standar_first_bool` [33](#), [97](#), [34](#), [289](#), [1755](#), [1902](#), [1964](#), [1971](#)
`__enumext_standar_item_vii:w` . [118](#), [4547](#), [4549](#), [4549](#)
`__enumext_standar_item_viii:w` [124](#), [4796](#), [4798](#), [4798](#)


```

\__enumext_standar_ref: . . . . 41, 607, 626, 3429
\__enumext_standar_ref:n . . . . 40, 599, 607, 607
\g__enumext_standar_series_tl . 61, 1757, 1758,
    1924, 1927
\__enumext_standar_unknown_keys:n 3267, 3271,
    3275
\__enumext_standar_unknown_keys:nn 3267, 3277,
    3279
\g__enumext_starred_bool 33, 116, 34, 272, 275, 299,
    359, 1795, 1838, 1849, 1876, 1891, 1930, 2036, 2082,
    2398, 2941, 4344
\l__enumext_starred_bool 116, 117, 122, 34, 1519,
    2436, 2471, 2477, 2525, 2814, 2819, 3050, 3063, 3483,
    4475, 4502, 4743, 4747
\__enumext_starred_columns_set_vii: . . 4121,
    4121, 4450
\__enumext_starred_columns_set_viii: . 4121,
    4152, 4718
\l__enumext_starred_first_bool 33, 116, 34, 304,
    855, 864, 1760, 1911, 1964, 1971
\__enumext_starred_item:nn . . . 3120, 3120, 3163
\__enumext_starred_item_exec: . 125, 4842, 4842,
    4882
\__enumext_starred_item_vii:w . 118, 119, 4546,
    4563, 4563
\__enumext_starred_item_vii_aux_i:w . . 4563,
    4568, 4571
\__enumext_starred_item_vii_aux_ii:w . 4563,
    4569, 4574, 4576
\__enumext_starred_item_vii_aux_iii:w 4563,
    4579, 4586
\__enumext_starred_item_viii:w 124, 4795, 4812,
    4812
\__enumext_starred_item_viii_aux_i:w . . 124,
    4812, 4817, 4820
\__enumext_starred_item_viii_aux_ii:w . 124,
    4812, 4818, 4835, 4837
\__enumext_starred_joined_item_vii:n 111, 118,
    4183, 4183, 4544
\__enumext_starred_joined_item_viii:n . 111,
    124, 4183, 4232, 4793
\__enumext_starred_ref: . . . . 42, 651, 687, 3460
\__enumext_starred_ref:n . . . . 41, 645, 651, 651
\g__enumext_starred_series_tl. 61, 1762, 1763,
    1932, 1935
\__enumext_starred_unknown_keys:n 3249, 3251,
    3253
\__enumext_starred_unknown_keys:nn 3249, 3255,
    3257
\__enumext_start_from:NNn 43, 745, 745, 758, 780,
    786
\l__enumext_start_i_int . . . . 1874, 1886, 1905
\__enumext_start_item_tmp_vii: 116, 4453, 4529,
    4529
\__enumext_start_item_tmp_viii: . . 122, 4721,
    4778, 4778
\__enumext_start_item_vii:w 118, 120, 4555, 4560,
    4583, 4590, 4638, 4638
\__enumext_start_item_viii:w . 124, 4804, 4809,
    4840, 4895, 4895
\g__enumext_start_line_tl 33, 34, 292, 307, 364,
    2106, 2111, 2116, 2130, 2135, 2140
\__enumext_start_list:nn . 36, 95, 382, 384, 3651,
    3784, 4444, 4712
\__enumext_start_list_tag:n . . 3797, 3821, 4654,
    4911
\__enumext_start_mini_vii: 117, 4281, 4281, 4494
\__enumext_start_mini_viii: . . 123, 4346, 4346,
    4766
\__enumext_start_save_ans_msg: . . 66, 67, 1948,
    1948, 1973
\__enumext_start_store_level: . 98, 3504, 3504,
    3650
\__enumext_start_store_level_vii: 117, 4443,
    4505, 4505
\l__enumext_start_vii_int . . . 1879, 1893, 1914
\l__enumext_start_X_int . . . . . 100
\__enumext_stop_item_tmp_vii: . . 116, 118, 120,
    4452, 4458, 4531, 4640
\__enumext_stop_item_tmp_viii: 122, 123, 4720,
    4726, 4780, 4897
\__enumext_stop_item_vii: 120, 121, 4638, 4640,
    4663
\__enumext_stop_item_viii: . . . 4895, 4897, 4931
\__enumext_stop_list: 36, 113, 382, 385, 3575,
    3583, 3737, 3744, 4304, 4312, 4369, 4376
\__enumext_stop_list_tag:n . . . 3797, 3837, 4666,
    4934
\__enumext_stop_mini_vii: 113, 117, 4281, 4300,
    4498
\__enumext_stop_mini_viii: 123, 4346, 4365, 4770
\__enumext_stop_save_ans_msg: . 66, 1948, 1953,
    2887
\__enumext_stop_start_list_tag: . . 3797, 3829,
    4656, 4913
\__enumext_stop_store_level: 98, 99, 3533, 3533,
    3576, 3584
\__enumext_stop_store_level_vii: . . 113, 117,
    4305, 4313, 4505, 4515
\l__enumext_store_active_bool 30, 67, 112, 1903,
    1912, 1980, 2612, 3508, 3521, 3669, 3677, 3995, 3999,
    4507, 4517, 4733, 4749
\__enumext_store_active_keys:n 72, 73, 98, 2219,
    2219, 3501
\__enumext_store_active_keys_vii:n 72, 73, 116,
    2219, 2229, 4486
\__enumext_store_addto_prop:n 74, 86, 2294, 2294,
    2302, 2462, 2922, 4845
\__enumext_store_addto_seq:n 74, 87, 2303, 2303,
    2307, 2314, 2328, 2336, 2345, 2359, 2367, 2520, 3015
\l__enumext_store_anskey_arg_tl 30, 77, 78, 112,
    2468, 2473, 2475, 2480, 2487, 2490, 2500, 2505, 2508,
    2514, 2520
\__enumext_store_anskey_code:n 77, 80, 84, 2459,
    2459, 2605, 2858, 2866
\l__enumext_store_anskey_env_tl . . 30, 83, 112,
    2788, 2792, 2798, 2860, 2868
\l__enumext_store_anskey_opt_tl . . 30, 84, 112,
    2789, 2816, 2822, 2829, 2835, 2845, 2855, 2864
\__enumext_store_anskey_safe_outer: . . . . 80
\g__enumext_store_columns_break_bool . 2712,
    2813, 2875
\l__enumext_store_columns_break_bool . 2470,
    2561
\l__enumext_store_current_label_tl 30, 86, 87,
    124, 112, 2903, 2906, 2909, 2915, 2920, 2922, 2979,
    2982, 2985, 2991, 2996, 3006, 3015, 4822, 4827, 4831,
    4844, 4845, 4847
\l__enumext_store_current_label_tmp_tl . 30,

```

[112](#), [3303](#), [3308](#)
`\l__enumext_store_current_opt_arg_tl` [30](#), [124](#),
[112](#), [3025](#), [3038](#), [3044](#), [4833](#)
`__enumext_store_internal_ref:` [76](#), [77](#), [2384](#),
[2384](#), [2465](#)
`\g__enumext_store_item_join_int` [2715](#), [2820](#),
[2824](#), [2876](#)
`\l__enumext_store_item_join_int` [2478](#), [2482](#),
[2564](#)
`\g__enumext_store_item_star_bool` [2717](#), [2827](#),
[2877](#)
`\l__enumext_store_item_star_bool` [2485](#), [2566](#)
`\g__enumext_store_item_symbol_sep_dim` [2722](#),
[2842](#), [2847](#), [2879](#)
`\l__enumext_store_item_symbol_sep_dim` [2497](#),
[2502](#), [2571](#)
`\g__enumext_store_item_symbol_tl` [2720](#), [2833](#),
[2837](#), [2878](#)
`\l__enumext_store_item_symbol_tl` [2488](#), [2492](#),
[2569](#)
`\l__enumext_store_keyans_item_opt_sep_-`
`tl` [2172](#), [2913](#), [2917](#), [2989](#), [2993](#), [4825](#), [4829](#)
`__enumext_store_level_close:` [74](#), [2308](#), [2332](#),
[3537](#)
`__enumext_store_level_close_vii:` [75](#), [2339](#),
[2363](#), [4521](#)
`__enumext_store_level_open:` [74](#), [98](#), [2308](#), [2308](#),
[3516](#), [3529](#)
`__enumext_store_level_open_vii:` [75](#), [2339](#),
[2339](#), [4511](#)
`\g__enumext_store_name_tl` [30](#), [67](#), [112](#), [363](#), [370](#),
[371](#), [372](#), [373](#), [1956](#), [1982](#), [2105](#), [2110](#), [2115](#), [2129](#),
[2134](#), [2139](#), [2885](#)
`\l__enumext_store_name_tl` [30](#), [67](#), [68](#), [112](#), [1789](#),
[1792](#), [1816](#), [1819](#), [1907](#), [1916](#), [1951](#), [1960](#), [1961](#), [1982](#),
[1983](#), [1984](#), [1986](#), [1987](#), [1989](#), [1991](#), [1992](#), [1994](#), [1996](#),
[1997](#), [2021](#), [2296](#), [2298](#), [2305](#), [2448](#), [2449](#), [2551](#), [2794](#),
[2962](#), [2963](#), [3076](#), [3089](#), [4862](#)
`\l__enumext_store_ref_key_bool` [77](#), [2195](#), [2463](#),
[2511](#), [2926](#), [3003](#)
`\l__enumext_store_save_key_vii_bool` [2231](#),
[2261](#)
`\l__enumext_store_save_key_vii_tl` [2233](#), [2234](#),
[2262](#), [2263](#), [2343](#), [2351](#), [2355](#), [2359](#)
`\l__enumext_store_save_key_X_bool` [72](#), [130](#)
`\l__enumext_store_save_key_X_tl` [72](#), [73](#), [130](#)
`\l__enumext_store_upper_level_X_bool` [130](#)
`__enumext_storing_exec:` [67](#), [82](#), [1958](#), [1974](#), [1978](#)
`__enumext_storing_set:n` [66](#), [67](#), [1943](#), [1958](#), [1958](#)
`\l__enumext_the_counter_v_tl` [731](#)
`\l__enumext_the_counter_vii_tl` [663](#)
`\l__enumext_the_counter_viii_tl` [679](#)
`\l__enumext_the_counter_X_tl` [50](#)
`__enumext_tmp:n` [45](#), [49](#), [54](#), [60](#), [71](#), [78](#), [79](#), [86](#), [94](#), [99](#),
[100](#), [111](#), [134](#), [141](#), [166](#), [170](#), [174](#), [194](#), [1696](#), [1707](#),
[1939](#), [1947](#), [2000](#), [2018](#), [2159](#), [2200](#), [2201](#), [2218](#), [2237](#),
[2250](#), [2386](#), [2393](#), [2394](#), [2415](#), [2428](#), [2431](#), [2442](#), [2928](#),
[2935](#), [3227](#), [3234](#), [3267](#), [3274](#), [3399](#), [3439](#), [3440](#), [3474](#)
`__enumext_tmp:nn` [514](#), [535](#), [536](#), [570](#), [571](#), [586](#), [775](#),
[800](#), [877](#), [899](#), [900](#), [920](#), [976](#), [984](#), [985](#), [999](#), [1064](#), [1080](#),
[1081](#), [1094](#), [1585](#), [1601](#), [3211](#), [3226](#)
`__enumext_tmp:nnn` [587](#), [603](#), [604](#), [605](#), [606](#), [633](#), [649](#),
[650](#)
`__enumext_tmp:nnnnnn` [801](#), [826](#), [829](#), [832](#), [834](#), [836](#),
[839](#), [842](#)
`__enumext_tmp:w` [4983](#), [4986](#)
`\l__enumext_tmpa_vii_int` [4131](#), [4134](#), [4143](#), [4174](#)
`\l__enumext_tmpa_viii_int` [4162](#), [4165](#)
`\l__enumext_tmpa_X_dim` [174](#)
`\l__enumext_tmpa_X_int` [174](#)
`\l__enumext_topsep_v_skip` [1161](#), [1165](#), [1364](#)
`\l__enumext_topsep_vii_skip` [1441](#), [1450](#), [1454](#)
`\l__enumext_topsep_viii_skip` [1463](#), [1485](#), [1489](#)
`__enumext_undefine_anskey_env:` [81](#), [85](#), [2645](#),
[2645](#), [2896](#)
`__enumext_unskip_unkern:` [33](#), [238](#), [238](#), [1214](#),
[1386](#), [3578](#), [3579](#), [3619](#), [3739](#), [3740](#), [3757](#)
`\l__enumext_vspace_a_star_v_bool` [1634](#)
`\l__enumext_vspace_a_star_vii_bool` [1656](#)
`\l__enumext_vspace_a_star_viii_bool` [1667](#)
`\l__enumext_vspace_a_star_X_bool` [100](#)
`__enumext_vspace_above:` [60](#), [99](#), [1602](#), [1602](#), [3589](#)
`__enumext_vspace_above_v:` [60](#), [1630](#), [1630](#), [3693](#)
`\l__enumext_vspace_above_v_skip` [1632](#), [1636](#),
[1638](#)
`__enumext_vspace_above_vii:` [61](#), [117](#), [1652](#), [1652](#),
[4491](#)
`\l__enumext_vspace_above_vii_skip` [1654](#), [1658](#),
[1660](#)
`__enumext_vspace_above_viii:` [61](#), [1652](#), [1663](#),
[4764](#)
`\l__enumext_vspace_above_viii_skip` [1665](#), [1669](#),
[1671](#)
`\l__enumext_vspace_b_star_v_bool` [1645](#)
`\l__enumext_vspace_b_star_vii_bool` [1678](#)
`\l__enumext_vspace_b_star_viii_bool` [1689](#)
`\l__enumext_vspace_b_star_X_bool` [100](#)
`__enumext_vspace_below:` [60](#), [100](#), [1616](#), [1616](#), [3627](#)
`__enumext_vspace_below_v:` [60](#), [1641](#), [1641](#), [3766](#)
`\l__enumext_vspace_below_v_skip` [1643](#), [1647](#),
[1649](#)
`__enumext_vspace_below_vii:` [61](#), [117](#), [1674](#), [1674](#),
[4501](#)
`\l__enumext_vspace_below_vii_skip` [1676](#), [1680](#),
[1682](#)
`__enumext_vspace_below_viii:` [61](#), [1674](#), [1685](#),
[4772](#)
`\l__enumext_vspace_below_viii_skip` [1687](#), [1691](#),
[1693](#)
`__enumext_widest_from:nnnn` [43](#), [759](#), [759](#), [774](#),
[793](#)
`\g__enumext_widest_label_tl` [28](#), [38](#), [67](#), [502](#), [506](#),
[510](#)
`\l__enumext_wrap_label_opt_v_bool` [3297](#)
`\l__enumext_wrap_label_opt_vii_bool` [118](#), [4554](#)
`\l__enumext_wrap_label_opt_viii_bool` [124](#),
[4803](#)
`\l__enumext_wrap_label_opt_X_bool` [100](#)
`\l__enumext_wrap_label_v_bool` [3293](#), [3297](#), [3305](#),
[3344](#), [3359](#)
`\l__enumext_wrap_label_vii_bool` [118](#), [4554](#),
[4558](#), [4566](#), [4630](#)
`\l__enumext_wrap_label_viii_bool` [124](#), [4803](#),
[4807](#), [4815](#), [4887](#)
`\l__enumext_wrap_label_X_bool` [100](#)
`__enumext_wrapper_label_v:n` [3346](#), [3361](#), [4055](#)
`__enumext_wrapper_label_vii:n` [4632](#)
`__enumext_wrapper_label_viii:n` [4889](#)
`\l__enumext_write_aux_file_tl` [31](#), [77](#), [87](#), [163](#),
[2451](#), [2457](#), [2969](#), [2975](#)

enumext* 5, [4438](#)
enumXi [474](#)
enumXii [474](#)
enumXiii [474](#)
enumXiv [474](#)
enumXv [474](#)
enumXvi [474](#)
enumXvii [474](#)
enumXviii [474](#)

Environments provide by [enumext](#):

anskey* . 30, 67, 73, 76, 78, 81–83, 85, 98, 117, 127, 128, 133, 136
enumext* 27, 28, 31–33, 36, 38, 41, 42, 44, 46–50, 56, 57, 61–64, 66–69, 71–77, 79, 81, 84–86, 91, 92, 97, 98, 103, 104, 110, 111, 113–115, 117, 119–123, 125–129, 131, 135, 137, 139
enumext 27, 28, 32, 33, 36, 38–41, 43–52, 55, 58–64, 66–69, 71–74, 76, 77, 79, 81, 84–86, 89–93, 95, 98, 101, 102, 106, 110, 113, 116, 117, 119, 122, 128, 129, 131, 134, 136, 138
keyans* 27, 28, 30–34, 38, 41–44, 46–50, 56, 57, 61, 67, 68, 71, 72, 74, 81, 86, 92, 97, 103, 104, 111, 112, 115, 122, 135, 137, 139
keyanspic 27, 28, 30, 31, 34, 38, 39, 42, 67, 68, 71, 74, 81, 86–88, 103–110, 137
keyans 27, 28, 30, 31, 33, 34, 38, 39, 42, 44, 46–49, 52, 55, 58–60, 67, 68, 71, 72, 74, 81, 86–88, 92–95, 101–103, 105, 106, 109, 113, 123, 135, 137

Environments:

center 110
description 110
enumerate 110
flushleft 110
flushright 110
itemize 110
list 32, 35, 36, 79, 95, 99, 101, 103–107, 110, 113
lrbox 120
minipage . 32, 35, 36, 50, 52, 53, 105, 106, 110, 113, 114, 120, 121
multicols 50–53, 58, 99, 100
quotation 110
quote 110
scontents 82, 84
tabbing 110
trivlist 110
verbatim 110
verse 110

exp commands:

\exp_after:wN 4986
\exp_args:Ne 2857, 2865, 3498, 4974
\exp_args:NV . . . 2577, 2732, 3237, 3255, 3277, 5259
\exp_not:N . 58, 505, 622, 666, 682, 734, 930, 944, 945, 957, 958, 970, 971, 2516, 2548, 2549, 3008, 3073, 3074, 3086, 3087, 4859, 4860, 4983
\exp_not:n 294, 309, 322, 330, 338, 561, 581, 622, 666, 682, 734, 931, 1723, 1732, 2183, 2280, 2292, 2454, 2482, 2492, 2502, 2516, 2517, 2824, 2837, 2847, 2972, 3010, 3012, 5088, 5098, 5291, 5296

F

\fbox 2166
\fboxrule 2166
\fboxsep 2166
file commands:
 \file_input_stop: 5695

first [985](#)
font [514](#)
\footnote 115
\footnote 115, 4413
\footnotemark 4423
\footnotesize 2549, 3074, 3087, 4860
\footnotetext 4407
\foreachkeyans 17, 132, [5224](#)

G

\getkeyans 17, 127, [4972](#)
group commands:
 \group_begin: . . 2547, 2592, 2767, 2854, 3072, 3085, 4858, 5029
 \group_end: 2554, 2608, 2871, 3079, 3092, 4865, 5036

H

\hbadness 4668, 4936
hbox commands:
 \hbox_overlap_left:n 3153, 4623
 \hbox_set:Nn 494, 3927
 \hbox_set_end: 4667, 4935
 \hbox_set_to_wd:Nnw 4641, 4898
\hfill 544, 549, 555, 556, 1541, 1568, 2516, 3008, 4308, 4372
hook commands:
 \hook_gput_code:nnn 9, 204, 208, 212, 409
 \hook_gremove_code:nn 84, 2783
 \hook_gset_rule:nnnn 410
 \hook_if_empty:nTF 2781
\hyperlink 78, 87
\hyperlink 2516, 3008
\hypertarget 37
\hypertarget 441

I

\IfDocumentMetadataTF 3170, 3332, 3823, 3831, 3839, 3873, 3881, 3889, 3948, 3958, 3966, 3976, 3981, 4020, 4029, 4106, 4114, 4306, 4370, 4449, 4457, 4599, 4647, 4670, 4717, 4725, 4904, 4938
\IfHyperBoolean 417
\IfPackageLoadedTF 11, 19, 413, 429
\ignorespaces . 933, 946, 959, 972, 4454, 4636, 4722, 4893
\inputlineno 294, 309, 322, 330, 338
int commands:
 \int_add:Nn 4216, 4265
 \int_case:nn . . . 1109, 1234, 2031, 2057, 2096, 2120
 \int_case:nnTF 240
 \int_compare:nNnTF . . 395, 654, 670, 689, 696, 1204, 1223, 1377, 1395, 1507, 1523, 1535, 1563, 2144, 2150, 2616, 2620, 2624, 2632, 2678, 2682, 2686, 2883, 2904, 2945, 2950, 2955, 2980, 3068, 3480, 3491, 3513, 3526, 3542, 3557, 3572, 3613, 3678, 3682, 3710, 3735, 3751, 3898, 4003, 4007, 4186, 4196, 4212, 4235, 4245, 4261, 4467, 4471, 4509, 4519, 4677, 4689, 4738, 4750, 4945, 4957, 5142, 5274
 \int_compare_p:nNn . . . 263, 273, 285, 286, 300, 301, 1513, 1514, 2037, 2063, 2399, 2409, 2421, 2422, 2437, 2478, 2655, 2656, 2667, 2668, 2820, 3523
 \int_decr:N 4215, 4264
 \int_eval:n . . 380, 788, 2298, 2449, 2549, 2963, 3074, 3087, 3414, 3459, 4204, 4253, 4860
 \int_from_alph:n 753, 767
 \int_from_roman:n 755, 769
 \int_gadd:Nn 4217, 4266
 \int_gdecr:N 2040, 2045, 2049, 2053, 2066

\int_gincr:N	1873, 1878, 2461, 3018, 3107, 3141, 3312, 3602, 3702, 4044, 4533, 4609, 4782, 4849
\int_gset:Nn	2089, 4421
\int_gset_eq:NN	1772, 1779, 1785, 1791, 1799, 1806, 1812, 1818, 4418
\int_gzero:N	351, 352, 353, 1549, 1576, 2156, 2876, 3618, 3756, 4700, 4969
\int_if_exist:NTF	1747, 1783, 1789, 1810, 1816, 1994
\int_incr:N	2631, 3479, 3673, 3897, 4466, 4532, 4737, 4781
\int_mod:nn	4691, 4959
\int_new:N	28, 29, 30, 31, 32, 33, 61, 62, 87, 104, 123, 144, 145, 156, 157, 158, 160, 171, 177, 178, 179, 180, 181, 1749, 1997
\int_set:Nn	749, 753, 755, 1886, 1893, 1905, 1914, 2768, 4100, 4101, 4131, 4162, 4185, 4191, 4207, 4234, 4240, 4256, 4668, 4936, 5138, 5276
\int_set_eq:NN	1874, 1879, 4214, 4263
\int_sign:n	2091
\int_step_function:nnN	2415, 2428, 2442
\int_step_function:nnnN	5280
\int_step_inline:nn	5190
\int_step_inline:nnn	4102
\int_to_roman:n	216, 2395, 2432
\int_use:N	373, 378, 379, 1205, 1224, 1536, 1888, 1895, 1907, 1916, 3414, 3434, 3459, 3499, 3543, 3552, 3567, 3573, 4189, 4190, 4202, 4238, 4239, 4251, 5611, 5615, 5621, 5625
\int_zero:N	4681, 4949
\item	89, 93, 118, 120, 123, 126, 386, 2316, 2322, 2347, 2353, 2475, 2982, 2985, 3159, 3316, 3952, 3954, 4451, 4453, 4719, 4721, 4847
\item*	5, 14, 71, 3314
item-pos*	3211
item-sym*	3211
\itemindent	96
\itemindent	95
itemindent	877
\itemsep	3942
\itemwidth	464, 2166, 3633, 3639, 3770, 3776, 4225, 4229, 4274, 4278

K

keyans	14, 3779
keyans*	14, 4707
keyanspic	15, 3944
Keys for \anskey provide by enumext:	
break-col	77, 79, 82-84
item-join	77, 79, 82-84
item-pos*	78, 79, 82-84
item-star	78, 79, 82-84
item-sym*	78, 79, 82-84
Keys for anskey* provide by enumext:	
break-col	77, 79, 82-84
item-join	77, 79, 82-84
item-pos*	78, 79, 82-84
item-star	78, 79, 82-84
item-sym*	78, 79, 82-84
Keys for environments provide by enumext:	
above*	29, 46, 59-61, 99, 117
above	29, 46, 59-61, 99, 117, 123
after	48, 100, 117, 123
align	29, 39, 40, 91, 94, 119, 133
base-fix	45, 62, 73, 98
before*	48, 99, 117, 123

before	48
below*	29, 59-61, 100, 117
below	29, 59-61, 100, 117, 123
check-ans	31-33, 66-71, 74, 85, 88, 100, 101, 117, 121, 135
columns-sep	49, 99, 121
columns	29, 49, 59, 99
first	48, 120
font	39, 91, 94, 108, 119
item-pos*	90, 92
item-sym*	30, 90, 92
itemindent	29, 46, 47, 89, 90, 93, 94, 120, 126
itemsep	44, 97, 121
labelsep	39, 96, 119
labelwidth	38-43, 96, 119
label	28, 38, 40, 43, 106, 110
lisparindent	97
list-indent	29, 46, 106
list-offset	46, 101, 103
listparindent	46, 120
mark-ans	71, 74, 78
mark-pos	71, 72, 133
mark-ref	71, 74, 76, 78
mini-env	29, 36, 49, 58, 59, 74, 100, 110, 113, 114, 117, 123
mini-right*	29, 32, 50, 74, 114, 117
mini-right	29, 32, 50, 57, 74, 114, 117
mini-sep	29, 49, 74, 100
no-store	31, 66-68, 73, 79, 89, 90
noitemsep	44
nosep	44
parindent	97
parsep	44, 97, 120
partopsep	44
ref	28, 32, 40-42, 135
resume*	28, 61, 62, 66, 67, 73, 100, 117, 129
resume	28, 35, 61-67, 73, 74, 100, 117, 129
rightmargin	46, 111
save-ans	30, 35, 62-68, 70, 72-74, 79-82, 85-87, 93, 101, 108, 119, 122, 123, 125, 127-129, 135
save-key	30, 62, 73, 98, 116
save-pos	74
save-ref	31, 37, 71, 74, 76-78, 86, 87, 94, 125
save-sep	71, 74, 86, 124
series	28, 61-66, 74, 98, 100, 116, 117, 129
show-ans	71, 72, 74, 75, 77, 78, 93, 108, 125
show-length	33, 47, 134, 135
show-pos	30, 71, 72, 75, 77, 78, 88, 93, 108, 125
start*	29, 43, 44, 62
start	29, 32, 43, 44, 62
store-key	72
topsep	44, 46
widest	28, 32, 43, 44
wrap-ans	37, 71, 74, 75, 78
wrap-label*	29, 39, 89, 91, 93, 94, 118, 119, 124
wrap-label	29, 39, 89-91, 93, 94, 106, 108, 118, 119, 124
wrap-opt	71, 74, 93, 108, 126
keys commands:	
\keys_define:nn	516, 538, 573, 589, 635, 704, 777, 803, 845, 879, 902, 978, 987, 1066, 1083, 1587, 1698, 1941, 2002, 2161, 2203, 2239, 2244, 2559, 2710, 2746, 3213, 3229, 3249, 3269, 5000, 5100, 5216, 5224
\keys_if_exist_p:nn	5212, 5213
\l_keys_key_str	79, 82, 2577, 2732, 3237, 3255, 3277, 5259, 5367

\keys_precompile:nnN . . . 128, 200, 200, 5002, 5006, 5010, 5014, 5018, 5022, 5242

\keys_set:nn . 530, 871, 1089, 1592, 1597, 1835, 1840, 1927, 1935, 2597, 3493, 3498, 3689, 4484, 4759, 5104, 5109, 5110, 5111, 5112, 5115, 5120, 5121, 5122, 5123, 5124, 5125, 5126, 5158, 5268

\keys_set_known:nn 2864

keyval commands:

\keyval_parse:NNn 1712, 2269, 5076

L

label 587, 633, 704

Labels provide by enumext:

\Alph* 38

\Roman* 38

\alph* 38

\arabic* 32, 38

\roman* 38

\labelsep 3938

labelsep 514

\labelwidth 38

\labelwidth 3938

labelwidth 514

\lastkern 249

\lastnodetype 240

\lastskip 244

\leftmargin 96

\leftmargin 95, 3938

legacy commands:

\legacy_if:nTF 4594, 4597, 4872, 4875

\legacy_if_gset_false:n 400, 4321

\legacy_if_set_false:n 4596, 4874

\legacy_if_set_true:n 4559, 4582, 4589, 4603, 4808, 4839

\linewidth 100

\linewidth 3597, 3633, 3699, 3770, 4099, 4134, 4165, 4287, 4352

\list 384

list-indent 877

list-offset 877

\listparindent 3940

listparindent 877

M

\makebox 110

\makebox 2374, 2376, 3196, 3356, 4037, 4627, 4884

\makelabel 89, 91, 94, 110

\makelabel 89, 93, 3178, 3194, 3340, 3354

\makesavenoteenv 435

mark-ans 2159

mark-pos 2159, 2201

mark-ref 2159

mini-env 1064

mini-sep 1064

\minipage 390

\miniright 10, 58, 1505, 1553, 1580, 3616, 3754

mode commands:

\mode_if_math:TF 2640, 2694

\mode_if_vertical:TF 1143, 1171, 1191, 1215, 1366, 1387

\mode_leave_vertical: 860, 867, 930, 944, 957, 970, 2372, 3151, 4621

msg commands:

\msg_error:nn . . 1555, 1582, 2601, 2634, 2638, 2692, 2800, 3680, 3684, 3900, 3956, 4005, 4469, 4740, 4752,

5127, 5186

\msg_error:nnn 612, 658, 674, 726, 1509, 1516, 1521, 1551, 1578, 1847, 1851, 1966, 2583, 2642, 2660, 2672, 2680, 2684, 2688, 2696, 2738, 3243, 3261, 3283, 4473, 4745, 4988, 4997, 5069, 5174, 5205, 5214, 5251, 5272

\msg_error:nnnn 2586, 2614, 2618, 2622, 2626, 2741, 3246, 3264, 3286, 3671, 4001, 4009, 4735, 5048, 5254

\msg_error:nnnnn 560, 580, 2182

\msg_fatal:nn 3481

\msg_fatal:nnn 468

\msg_info:nnn 13, 16, 21, 24, 415, 431

\msg_line_context: . . 5332, 5337, 5342, 5371, 5376, 5381, 5396, 5411, 5415, 5419, 5423, 5427, 5431, 5438, 5445, 5451, 5465, 5469, 5474, 5478, 5482, 5486, 5491, 5495, 5499, 5503, 5508, 5543, 5547, 5552, 5557, 5561, 5566, 5642, 5646, 5651, 5656, 5661, 5665, 5669, 5673, 5677, 5681, 5685, 5689, 5693

\msg_log:nnn 1986, 1991, 1996

\msg_log:nnnnn 377, 2129, 2134, 2139

\msg_log:nnnnnn 369

\msg_new:nnn 5299, 5303, 5307, 5311, 5316, 5329, 5334, 5339, 5344, 5353, 5361, 5365, 5369, 5374, 5379, 5394, 5409, 5413, 5417, 5421, 5425, 5429, 5433, 5442, 5448, 5454, 5458, 5462, 5467, 5472, 5476, 5480, 5484, 5489, 5493, 5497, 5501, 5506, 5541, 5545, 5550, 5555, 5559, 5564, 5640, 5644, 5649, 5654, 5659, 5663, 5667, 5671, 5675, 5679, 5683, 5687, 5691

\msg_new:nnnn . . 5320, 5511, 5520, 5529, 5535, 5568, 5578, 5588, 5598, 5608, 5618, 5628, 5634

\msg_term:nnnn . 1950, 1955, 3423, 3433, 3465, 3470

\msg_term:nnnnn 2110

\msg_warning:nn 3615, 3753

\msg_warning:nnnn 2147, 2153, 3371, 3376, 4188, 4201, 4237, 4250

\msg_warning:nnnnn 2105, 2115

\multicolsep 99

\multicolsep 1208, 1380, 3563, 3726

N

\NeedsTeXFormat 3

\NewCommandCopy 386

\newcounter 471

\NewDocumentCommand 1505, 2589, 3997, 4972, 5027, 5134, 5183, 5261

\NewDocumentEnvironment . 3645, 3779, 3944, 4438, 4707

\newenvsc 2703

\newlabel 37

\newlabel 453

no-store 2000

\noindent 3604, 4296, 4361, 4680, 4948

\nointerlineskip 1217, 1220, 1389, 1392, 1543, 1570, 4296, 4361

noitemsep 801

\nopagebreak 1154, 1182, 1217, 1220, 1389, 1392, 1496, 1502

\normalfont 2548, 3073, 3086, 4859

nosep 801

P

Packages:

caption 114

enumext 27, 37, 40, 66, 95, 105, 133, 134

enumitem 38

expl3 110

footnotehyper 37

hyperref 31, 32, 36, 37, 78, 87, 119, 133

ltxcmd 35

lua-visual-debug	52
multicol	27, 133
scontents	27, 81, 82
shortlst	110, 116, 120
\par ..	1154, 1182, 1220, 1392, 1496, 1502, 1538, 1543, 1565, 1570, 2524, 3580, 3741, 3759, 3990, 3993, 4119, 4323, 4338, 4384, 4398, 4680, 4948
para commands:	
\para_end:	4697, 4966
\parbox	2166
\parindent	4660, 4928
\parsep	51, 106
\parsep	861, 3456, 3923, 3932, 3936
parsep	801
\parskip	4661, 4929
\partopsep	3457, 3757, 3941
partopsep	801
peek commands:	
\peek_meaning:N	4538, 4552, 4567, 4578, 4787, 4801, 4816
\peek_meaning_remove:N	4545, 4794
\peek_remove_spaces:n	3320
\phantomsection	37
\phantomsection	442
prg commands:	
\prg_do_nothing:	446
\prg_new_protected_conditional:Npnn	218
\prg_replicate:nn	235
\prg_return_false:	222
\prg_return_true:	221
\printkeyans	17, 128, 5027
prop commands:	
\prop_const_from_keyval:Nn	5175
\prop_count:N	371, 2298, 2449, 2551, 2963, 3076, 3089, 4862, 5277
\prop_get:NnNTF	5201
\prop_gput_if_not_in:Nnn	2296
\prop_if_exist:N	1984, 4992, 5270
\prop_item:Nn	4994, 5294
\prop_new:N	1987
\ProvidesExplPackage	4

R

\raggedcolumns	3566, 3729
\raisebox	4068
\ref	76, 86
ref	587, 633, 704
\refstepcounter	4606, 4877
regex commands:	
\regex_match:nnTF	220, 752, 754, 766, 768, 2796
\regex_replace_once:nnN	228
\renewcommand	622, 666, 682, 734
\RenewDocumentCommand	1553, 1580, 3159, 3178, 3194, 3316, 3340, 3354, 3954, 4413
\RequirePackage	17, 25
resume	1696
resume*	1696
rightmargin	877
\Roman	38, 43
\Roman	490
\roman	38, 43
\roman	491, 605, 5017

S

\s	2797
----	------

save-ans	1939
save-key	2237
save-ref	2159
save-sep	2159
scan commands:	
\scan_stop:	3952, 4451, 4719, 4983, 4986
scontents internal commands:	
\l_scontents_fname_out_tl	2756
__scontents_parse_environment_keys:n	2762
__scontents_rescan_tokens:n	2769
\l_scontents_storing_bool	2754
\l_scontents_writing_bool	2755
seq commands:	
\seq_clear:N	5136, 5279
\seq_const_from_clist:Nn	5129
\seq_count:N	372, 3973, 5140
\seq_gclear:N	4411, 4412
\seq_gput_right:Nn	2305, 4424, 4425
\seq_if_empty:N	4430, 5042, 5154
\seq_if_exist:N	1989, 5040
\seq_if_in:NnNTF	5046
\seq_item:Nn	2794, 4112
\seq_map_function:NN	5145
\seq_map_inline:Nn	5055, 5063, 5155, 5156
\seq_map_pairwise_function:NNN	4432
\seq_new:N	124, 125, 127, 142, 172, 173, 1992
\seq_pop_left:NN	5144
\seq_put_right:Nn	4011, 5152, 5168, 5289
\seq_set_from_clist:Nn	5137
\seq_set_map_e:NNn	5146
\seq_use:Nn	200, 201, 5285
series	1696
\setcounter	763, 767, 769, 3414, 3459, 3987
\setenumext	6, 129, 5134
\setenumextmeta	6, 131, 5175
show-ans	2159, 2201
show-length	976
show-pos	2201
skip commands:	
\skip_add:Nn	1114, 1123, 1132, 1145, 1149, 1173, 1177, 1193, 1251, 1253, 1267, 1270, 1291, 1293, 1307, 1310, 1330, 1332, 1346, 1349, 1368, 1417, 1418, 1429, 1431, 3932, 3939
\skip_gset:Nn	1444, 1448, 1452
\skip_gset_eq:NN	3936
\skip_gzero_new:N	1439, 1440
\skip_horizontal:N	945, 958, 971, 4624, 4636, 4684, 4893, 4952
\skip_horizontal:n	931, 2373, 2381, 3152, 3154, 4527, 4622, 4776, 4920
\skip_if_eq:nnTF	1112, 1121, 1130, 1237, 1277, 1317, 1405, 1441, 1463, 1604, 1618, 1632, 1643, 1654, 1665, 1676, 1687
\skip_new:N	81, 82, 83, 88, 89, 90, 91, 92, 93, 148, 192
\skip_set:Nn	1097, 1101, 1159, 1163, 1187, 1240, 1241, 1259, 1280, 1281, 1299, 1319, 1320, 1338, 1362, 1408, 1409, 1423, 1443, 1447, 1465, 1469, 1473, 1479, 1483, 1487, 3916
\skip_set_eq:NN	1198, 1199, 1201, 1208, 1373, 1374, 1375, 1380, 3412, 3455, 3456, 4661, 4929
\skip_sub:Nn	1247, 1249, 1263, 1265, 1287, 1289, 1303, 1305, 1326, 1328, 1342, 1344, 1415, 1416, 1427, 1428
\skip_use:N	1099, 1103, 1147, 1151, 1155, 1175, 1179, 1189, 1195, 1605, 1609, 1612, 1619, 1623, 1626, 3580

<code>\skip_vertical:N</code>	401, 404, 869, 4322, 4336, 4699, 4968
<code>\skip_vertical:n</code>	868, 4698, 4967
<code>\skip_zero:N</code>	1207, 1221, 1359, 1360, 1361, 1379, 1393, 3457, 3563, 3726, 3941, 3942
<code>\skip_zero_new:N</code>	1438, 1460, 1461, 1462
<code>\c_zero_skip</code>	401, 404, 869, 1112, 1121, 1130, 1278, 1317, 1441, 1463, 1605, 1619, 1632, 1643, 1654, 1665, 1676, 1687, 4322, 4336, 4699, 4968
<code>\small</code>	5005, 5009, 5013, 5017, 5021, 5025
socket commands:	
<code>\socket_assign_plug:nn</code>	3825, 3833, 3841, 3875, 3883, 3891
<code>\socket_new:nn</code>	3797, 3845
<code>\socket_new_plug:nnn</code>	3798, 3805, 3813, 3846, 3853, 3862
<code>\socket_use:n</code>	3826, 3876
<code>\socket_use:nn</code>	3834, 3842, 3884, 3892
<code>\star</code>	3217
<code>start</code>	775
<code>start*</code>	775
<code>start-list-tags</code>	3797, 3845
<code>\stepcounter</code>	3926, 4061, 4417
<code>stop-list-tags</code>	3797, 3845
<code>stop-start-tags</code>	3797, 3845
str commands:	
<code>\c_backslash_str</code>	2642, 5332, 5337, 5342, 5347, 5349, 5351, 5356, 5358, 5456, 5460, 5464, 5474, 5478, 5486, 5487, 5491, 5503, 5504, 5508, 5509, 5530, 5532, 5536, 5538, 5566, 5629, 5631, 5635, 5637, 5646, 5647, 5651, 5656, 5657, 5661, 5665, 5669
<code>\c_colon_str</code>	2448, 2962, 4983
<code>\c_left_brace_str</code>	5437, 5444, 5450
<code>\c_right_brace_str</code>	5437, 5444, 5450
<code>\str_case:nn</code>	256, 315
<code>\str_case:nnTF</code>	1719, 1727, 2276, 2284, 5083, 5092
<code>\str_clear:N</code>	3490, 4483
<code>\str_count:n</code>	235
<code>\str_if_empty:N</code>	1736, 1777, 1804
<code>\str_if_eq:nnTF</code>	3415, 3461, 5185
<code>\str_if_in:nnTF</code>	4979
<code>\str_new:N</code>	84, 132, 147, 187
<code>\str_set:Nn</code>	545, 551, 557, 576, 577, 578, 2179, 2180, 2206, 2207, 3906, 3909
<code>\str_use:N</code>	3198
<code>\string</code>	435
<code>\strutbox</code>	1226, 1229, 1240, 1241, 1252, 1254, 1269, 1272, 1280, 1281, 1292, 1294, 1309, 1312, 1319, 1320, 1331, 1333, 1348, 1351, 1397, 1400, 1408, 1409, 1417, 1418, 1430, 1432, 1443, 1444, 1447, 1454, 1467, 1475, 1481, 1489, 3934, 3939, 3990, 4074

T

tag commands:	
<code>\tag_mc_begin:n</code>	3803, 3851, 3860
<code>\tag_mc_end:</code>	3807, 3855, 3864
<code>\tag_resume:n</code>	3800, 3848, 3960, 3968, 4031, 4116, 4306, 4370
<code>\tag_struct_begin:n</code>	3801, 3802, 3809, 3810, 3811, 3849, 3850, 3857, 3858, 3859, 3969
<code>\tag_struct_end:</code>	3983, 3984
<code>\tag_struct_end:n</code>	3808, 3815, 3816, 3817, 3818, 3856, 3865, 3866, 3867, 3868, 4457, 4725
<code>\tag_suspend:n</code>	3819, 3869, 3950, 3962, 3978, 4022, 4108, 4449, 4717

<code>\tag_tool:n</code>	3961
TeX and L ^A T _E X 2 _ε commands:	
<code>\@auxout</code>	451
<code>\@currentenv</code>	256, 315
<code>\protected@write</code>	451
tex commands:	
<code>\tex_newlinechar:D</code>	2768
text commands:	
<code>\text_expand:n</code>	4975
<code>\textasteriskcentered</code>	2176, 2193
<code>\the</code>	244, 249
<code>\thepage</code>	457
tl commands:	
<code>\c_space_tl</code>	3044, 5381, 5396, 5419, 5423, 5610, 5611, 5620, 5621, 5681, 5685
<code>\tl_clear:N</code>	543, 550, 2157, 2223, 2233, 2254, 2262, 2468, 2788, 2789, 2903, 2979, 4822
<code>\tl_clear_new:N</code>	500
<code>\tl_const:Nn</code>	50, 484
<code>\tl_gclear:N</code>	363, 364, 365, 1757, 1762, 2878, 3189, 3207, 4342, 4402, 4625
<code>\tl_gclear_new:N</code>	1744
<code>\tl_gput_right:Nn</code>	485
<code>\tl_greplace_all:Nnn</code>	506
<code>\tl_gset:Nn</code>	291, 292, 306, 307, 1745, 1758, 1763, 1982, 2792, 3128, 4573
<code>\tl_gset_eq:NN</code>	502, 3124, 4618
<code>\tl_if_blank:nTF</code>	2581, 2599, 2736, 3241, 3259, 3281, 4616, 5249
<code>\tl_if_empty:N</code>	610, 628, 656, 672, 691, 698, 724, 740, 1770, 1775, 1797, 1802, 1860, 1924, 1932, 1961, 2021, 2312, 2343, 2488, 2833, 2855, 2885, 2913, 2989, 3038, 3149, 4825, 5166
<code>\tl_if_empty:nTF</code>	1825
<code>\tl_if_exist:N</code>	1830
<code>\tl_if_novalue:nTF</code>	2595, 2911, 2987, 3023, 3103, 3122, 3130, 3291, 3488, 3971, 4415, 4481, 4757, 4823
<code>\tl_map_inline:Nn</code>	226, 503
<code>\tl_new:N</code>	42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 105, 106, 107, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 126, 128, 129, 130, 133, 136, 137, 155, 163, 164, 165, 168, 186
<code>\tl_put_left::Ne</code>	2822
<code>\tl_put_left:Nn</code>	2320, 2351, 2473, 2816, 2829, 2835, 2845, 3055, 3095, 4326, 4387, 4844, 4847
<code>\tl_put_right:Nn</code>	501, 620, 664, 680, 732, 2324, 2355, 2402, 2412, 2425, 2440, 2446, 2451, 2475, 2480, 2487, 2490, 2500, 2505, 2508, 2514, 2906, 2909, 2915, 2920, 2947, 2952, 2957, 2960, 2969, 2982, 2985, 2991, 2996, 3006, 4827, 4831
<code>\tl_remove_all:Nn</code>	5165
<code>\tl_remove_once:Nn</code>	2390, 2932
<code>\tl_replace_all:Nnn</code>	505, 5200
<code>\tl_reverse:N</code>	2389, 2391, 2931, 2933
<code>\tl_set:Nn</code>	58, 260, 270, 319, 320, 327, 328, 335, 336, 470, 544, 549, 555, 556, 609, 653, 723, 928, 942, 955, 968, 1859, 1960, 2224, 2234, 2255, 2263, 2545, 2756, 3025, 3070, 3083, 4833, 4856, 5163, 5199, 5269
<code>\tl_set_eq:NN</code>	511, 615, 618, 661, 663, 677, 679, 729, 731, 2388, 2930, 2943, 3303, 3308, 4049, 4051
<code>\tl_to_str:n</code>	1830, 1836, 1841, 4975
<code>\tl_trim_spaces:n</code>	501, 5152, 5163, 5169, 5185
<code>\tl_use:N</code>	507, 510, 630, 693, 700, 742, 1002, 1006, 1010, 1014, 1018, 1022, 1026, 1030, 1034, 1038, 1042, 1046, 1050, 1054, 1058, 1062, 2378, 2395, 2403, 2414, 2427,

2432, 2443, 3111, 3117, 3145, 3180, 3181, 3188, 3200,
3294, 3298, 3306, 3342, 3343, 3349, 3358, 3652, 3785,
4054, 4333, 4394, 4629, 4658, 4659, 4886, 4915, 4918,
4926, 5030, 5031, 5032, 5033, 5034, 5051, 5148, 5267

token commands:

\token_to_str:N 453

\topsep 3757, 3939

topsep 801

\topskip 1207, 1379

\typeout 244, 249, 419, 423, 434, 435

U

\u 229, 2797

\unkern 250

unknown 3227, 3249, 3267

\unskip 245

use commands:

\use:N 236, 3185, 3204, 3654

\use:n 1710, 2267, 4981, 5074

\use_none:nn 445, 5206

\usecounter 3413, 3458

V

\value 1773, 1779, 1786, 1792, 1800, 1806, 1813, 1819

vbox commands:

\vbox_set:Nn 4024

\vbox_set_top:Nn 4331, 4392

\vspace 861, 1609, 1612, 1623, 1626, 1636, 1638, 1647, 1649,
1658, 1660, 1669, 1671, 1680, 1682, 1691, 1693

W

widest 775

wrap-ans 2159

wrap-label 514

wrap-label* 514

wrap-opt 2159

Z

\z 2797