

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-10-02^{*}

©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	12
1.1	Description and usage	2	6.1	Keys for storage system	12
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	3	6.1.2	Keys for wrap and display	13
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	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>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	16
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	19
5	The <code>keyval</code> system	7	8	The way of non-enumerated lists	21
5.1	Keys for label and ref	7	9	References	23
5.2	Keys for spaces	8	10	Change history	24
5.2.1	Vertical spaces	8	11	Index of Documentation	25
5.2.2	Horizontal spaces	9	12	Implementation	27
5.3	Keys for add code	9	13	Index of Implementation	140
5.4	Keys for start, series and resume	10			
5.5	Keys for <code>multicols</code>	11			
5.6	Keys for <code>minipage</code>	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

Motivation and acknowledgments

Usually it is enough to use the classic `enumerate` environment to generate “simple exercise sheets” or “multiple choice questions”, the basic idea behind `enumext` is to cover three points:

1. To have a simple interface to be able to write “lists of exercises” with “answers”.
2. To have a simple interface for writing “multiple choice questions”.
3. To have a simple interface for placing “columns” and “drawings” or “tables”.

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all 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 2024-10-02.

[†]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

enumext.pdf

README.md

enumext.dtx

»

»

»

»

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

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.

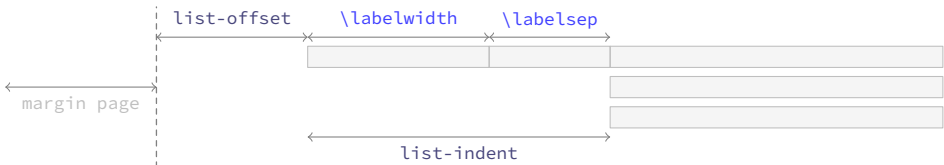


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

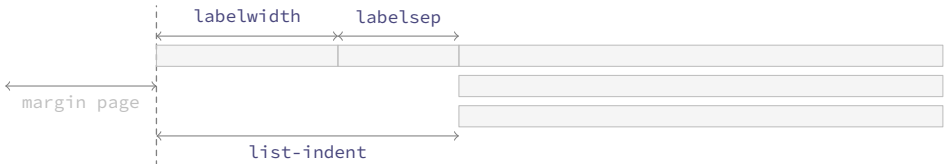


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

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*” 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 `(\alph^*)`, for third level are `\roman^*`. and for fourth level are `\Alpha^*`. For `keyans` and `keyans*` environments the default value is `\Alph^*`.

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

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

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

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

Sets the *width* of the box containing the current $\langle label \rangle$ set by `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 `\Alph^*`, ‘m’ for `\alph^*`, ‘VIII’ for `\Roman^*` and ‘viii’ for `\roman^*`.

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

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

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

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

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

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

`wrap-label = { $\langle code \{ \#1 \} \text{ more code} \rangle$ }` 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\alph^*\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`.

`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`.

`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 on the “above” the environment without hindering the value of the `topsep` key. The space is added with `\vspace` so is “discordable”.

`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 on the “*above*” the environment without hindering the value of the `topsep` key. The space is added with `\vspace*` so is “*not discardable*”.

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

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

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

Set the *extra vertical space* 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 = {⟨rigid length⟩}` 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 = {⟨rigid length⟩}` default: *0pt*

Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to *0pt*. Internally sets the value of `\rightmargin` for the current level.

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

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

`list-offset = {⟨rigid length⟩}` 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 = {⟨rigid length⟩}` default: *labelwidth + labelsep*

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

If `list-indent=0pt` is set in the environment `enumext` the `⟨label⟩` will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”. 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 `⟨keys⟩` should be used with “*caution*”, they are intended to inject `{⟨code⟩}` into different parts of the defined environments. We must keep in mind that the defined environments are based on the `list` base environment provided by `LTEX` which is defined (simplified) as plain form `\list{⟨arg one⟩}{⟨arg two⟩}`. Using the `before*` key does not allow access to the `list` parameters defined by `[⟨key = val⟩]`.

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

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

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

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

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

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

🟡 Keep in mind that the code set in this key will affect the entire “*body*” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.

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

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

5.4 Keys for start, series and resume

`start = {⟨integer | integer expression⟩}`

default: 1

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

`start* = {⟨integer | string⟩}`

default: not used

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

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

`series = {⟨series name⟩}` default: *not used*

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

`resume = {⟨series name⟩}` default: *not used*

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

`resume*` *⟨value forbidden⟩* default: *not used*

Sets the *start value* and *options* for the “first level” continuing the numbering of the environment in which the `series={⟨series name⟩}` or `resume={⟨series name⟩}` keys are NOT present, if the `save-ans` key is active it will continue the numbering from the last environment in which it was executed. The *start value* can be overwritten using `start` or `start*` keys.

- For security reasons the `series` key will never save in `{⟨series name⟩}` the keys `series`, `resume`, `resume*`, `save-ans`, `save-key`, `start*` and `start`. When using the key `resume={⟨series name⟩}` it will have hierarchy in the *⟨keys⟩* that are saved in `{⟨series name⟩}`, in order to establish the value of a *⟨key⟩* already saved in `{⟨series name⟩}` it must be placed to the “right” of `resume={⟨series name⟩}`, the same thing happens with the `resume*` key, the exception is the `save-ans` key that must be placed on the “left” if you want to start the numbering with its value. The `resume` key passed “without value” must be exactly “without value”, i.e. `resume=` cannot be used and if executed before `resume*` it will affect the *start value*.

5.5 Keys for 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 \TeX 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 $\langle key \rangle$ is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

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

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

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

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

6.1 Keys for storage system

- The only $\langle keys \rangle$ available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the $\langle keys \rangle$ 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 $\{\langle contents \rangle\}$ will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments and the current $\langle labels \rangle$ for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic`. If the *sequence* or *prop list* $\{\langle store name \rangle\}$ 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 $\langle key list \rangle$ passed to this key ignores any $\langle keys \rangle$ in the “*stored structure*” and must be passed between braces. For example, if we execute at a second level:

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

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

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

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

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* $\{\langle store name \rangle\}$ set by `save-ans` key. To reference the location of the “*stored content*” within the environment you must use `\ref{\store name : position}`, where $\langle position \rangle$ corresponds to the position occupied by the “*stored content*” in the *prop list* $\{\langle store name \rangle\}$ returned by the `show-pos` key. For example `\ref{test:4}` will return `3`. (b) which corresponds to the location of the “*stored content*” at position `4` in *prop list* `test` within the environment in which the key `save-ans=test` was set.

`mark-ref = {\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 `{content}` passed to `\anskey` when “storing” in the *sequence* `{store name}` has the form `\item{content}`, the following *keys* allow modifying the way in which it is “stored” in the *sequence*.

- `break-col` `{value forbidden}` default: `not used`
 Stores `{content}` in the *sequence* `{store name}` of the form `\columnbreak \item{content}`.
- `item-join` = {`{columns}`} default: `not set`
 Set the *number of columns* to be used for `\item({columns})` and stores `{content}` in the *sequence* `{store name}` of the form `\item({columns}){content}`.
- `item-star` `{value forbidden}` default: `not used`
 Stores `{content}` in the *sequence* `{store name}` of the form `\item*{content}`.

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

`item-pos*` = {*rigid length*} default: *not set*
 Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores {*content*} in the *sequence* {*store name*} of the form `\item*[symbol][offset] content`.

Example

```
\begin{enumext}[save-ans=test,show-ans=true]
  \item* Text containing our instructions or questions. \anskey{first answer}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question. \anskey{second answer}
    \end{enumext}
  \item Text containing our instructions or questions. \anskey{third answer}
  \item Text containing our instructions or questions. \anskey{fourth answer}
\end{enumext}
```

- | | |
|--|---|
| * 1. Text containing our instructions or questions.
* <input type="text" value="first answer"/>
2. Text containing our instructions or questions.
(a) Question.
* <input type="text" value="second answer"/> | 3. Text containing our instructions or questions.
* <input type="text" value="third answer"/>
4. Text containing our instructions or questions.
* <input type="text" value="fourth answer"/> |
|--|---|

6.3 The environment `anskey*`

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

The environment `anskey*` takes a mandatory {*body content*} and “stores” it in the *sequence* and *prop list* {*store name*} set by `save-ans` key. If `save-ref` key are active and the `hyperref`[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.

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 [`key = val`] 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` = {*file.ext*} default: *not used*
 Sets the name of the {*external file*} in which the {*contents*} of the environment will be written. The {*file.ext*} will be created in the working directory, relative or absolute paths are not supported. If {*file.ext*} does not exist, it will be created or overwritten if the `overwrite` key is used.

`overwrite` = {*true* | *false*} default: *false*
 Sets whether the {*file.ext*} generated by `write-env` from the `anskey*` environment will be rewritten.

`force-eol` = {*true* | *false*} default: *false*
 Sets if the *end of line* for the {*stored content*} is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments “%” after closing `\end{Verbatim}%`.

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

Example

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

```
\item Text containing our instructions or questions.
\begin{enumext}
  \item Question.
    \begin{anskey*}
      \langle second answer \rangle
    \end{anskey*}
  \end{enumext}
\item Text containing our instructions or questions.
\begin{anskey*}
  \langle third answer \rangle
\end{anskey*}
\item Text containing our instructions or questions.
\begin{anskey*}
  \langle fourth answer \rangle
\end{anskey*}
\end{enumext}
```

- ★ 5. Text containing our instructions or questions.

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

(a) Question.

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

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

[8] fourth answer

6.4 The environments keyans and keyans*

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

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

The `keyans` and `keyans*` environments are “*enumerated list*” environments designed for “*multiple choice*” questions activated by the `save-ans` key. This environments can NOT be nested and must always be at the “*first level*” of the `enumext` environment, the commands `\item` and `\item[\langle custom \rangle]` work in the usual and the command `\item[\langle columns \rangle]` is available for the `keyans*` environment.

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

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

The `\langle keys \rangle` set in the *optional argument* of the environment are the same (almost) as those of the `enumext` and `enumext*` environments and have *higher precedence* than those set by `\setenumext[\langle keyans \rangle]{\langle key = val \rangle}` or `\setenumext[\langle keyans* \rangle]{\langle key = val \rangle}`. If the *optional argument* is not passed or the `\langle keys \rangle` are not set by `\setenumext`, the default values will be the same as the “*second level*” of the `enumext` environment with the difference in the `\langle label \rangle` which will be set to `label=\Alph*`.

6.4.1 The \item* in keyans and keyans*

\item* \item*

\item*[\langle content \rangle]

The `\item*` and `\item*[\langle content \rangle]` command “*store*” the current `\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*}
\end{enumext}
```

```
\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

C) Choice

E) Choice

* B) Correct choice

D) 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}^{\textit{o}} upper, \textit{n}^{\textit{o}} 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.

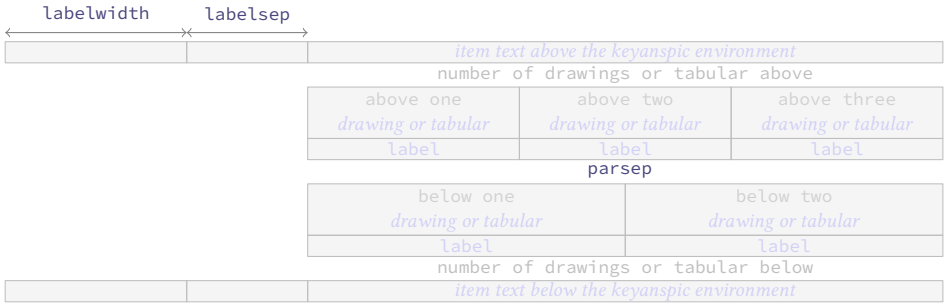


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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 “`no 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* “`content`” in *sequence* and *prop list* “`{store name}`” set by `save-ans` key.

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

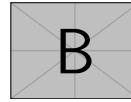
Example

```
\begin{enumext}[save-ans=test,show-ans,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*[\textit{note}]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
\end{enumext}
```

1. Question with images.



A)



B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

`sep = {code}` default: 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 {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* $\{\langle store\ name\rangle\}$ the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence* $\{\langle store\ name\rangle\}$ it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

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

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

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

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

Example

```
\begin{enumext}[save-ans=sample,columns=2,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)

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

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

1. $3(x + y + z)$

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*


7 Full examples

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

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

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

Example 1

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

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

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

*

*

*

*

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

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

 value

B

 correct

C

 value

D

 value
2. Second type of questions
- I. $2\alpha + 2\delta = 90^\circ$

II. $\alpha = \delta$

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

A

 I only

B

 II only

C

 I and II only

D

 I and III only

E

 I, II, and III
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$

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

A

 value

B

 value

C

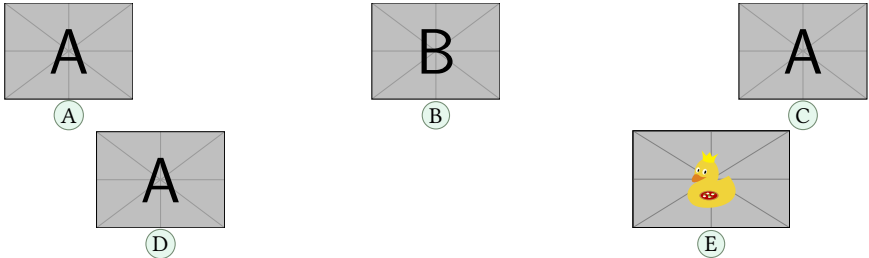
 value

D

 value

E

 value
4. Question with image and label below:



5. Question with image on left side:
- A

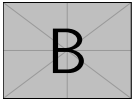
 value
- B

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

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

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

- First level item
 - Second level item
 - * Third level item
 - Fourth level item
 - First level item
- * First level item
 - ◇ Second level item
 - Third level item
 - ★ Fourth level item
 - * First level item

Fake description environment

Here 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*` key 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.

9 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.

10 Change history

v1.0 2024-10-02 – First public release.

11 Index of Documentation

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

C		I	
Document class:		\itemsep	8
article	2	K	
book	2	Keys for \anskey provide by enumext:	
exam	2	break-col	13
letter	2	item-join	13
report	2	item-pos*	14
\columnbreak	4, 13	item-star	13, 14
\columnsep	11	item-sym*	14
Commands provide by enumext:		Keys for \foreachkeyans provide by enumext:	
\anskey	12-14	after	17
\anspic	12, 13, 16	before	17
\foreachkeyans	17	sep	17
\getkeyans	13, 17	start	17
\item*	5-7, 12, 13, 15, 16	step	17
\item	5-7, 9, 11, 13, 15	stop	17
\miniright	11	wrapper	17
\printkeyans	6, 12, 17	Keys for anskey* provide by enumext:	
\setenumextmeta	6	break-col	13
\setenumext	5-7, 12, 13, 15, 18	force-eol	14
Counters defined by enumext:		item-join	13
enumXiii	4	item-pos*	14
enumXii	4	item-star	13, 14
enumXiv	4	item-sym*	14
enumXi	4	overwrite	14
enumXviii	4	write-env	14
enumXvii	4	Keys for environments provide by enumext:	
enumXvi	4	above*	8
enumXv	4	above	8
E		after	9, 11
Environments provide by enumext:		align	7, 22
anskey*	12-14	base-fix	8
enumext*	4-15, 17, 18	before*	9
enumext	4-10, 12-15, 17, 18, 21	before	9
keyans*	4-9, 11-15	below*	9
keyanspic	4, 7, 8, 12-14, 16, 21	below	9
keyans	4-9, 12-16, 21	check-ans	13
Environments:		columns-sep	4, 11
Verbatim	14	columns	4, 8, 11
center	5	first	9
description	5	font	7
enumerate	1, 3, 5, 23	item-pos*	5, 6
figure	5	item-sym*	5, 6
flushleft	5	itemindent	9
flushright	5	itemsep	8
itemize	5	labelsep	3-7, 9, 11, 13, 22
list	3, 5, 9, 23	labelwidth	3, 4, 6, 7, 9, 11, 13, 22
minipage	3-5, 11, 23	labelwith	5
multicols	3, 4, 11	label	7, 8, 10, 15, 21, 22
quotation	5	list-indent	3, 9
quote	5	list-offset	3, 9, 22
tabbing	5	listparindent	9
table	5	mark-ans	13
task	5	mark-pos	13
trivlist	5	mark-ref	12
verbatim	5	mini-env	4, 8, 11, 12
verse	5	mini-right*	7, 11, 12
F		mini-right	7, 11, 12
\footnote	5	mini-sep	4, 11, 12
		no-store	12-14

noitemsep	8	\alph*	7
nosep	8, 21	\arabic*	7
overwrite	14	\roman*	7
parsep	8, 16	\labelsep	3, 7
partopsep	8	\labelwidth	3, 7
ref	4, 7, 8	\linewidth	11
resume*	7, 11, 12	\listparindent	9
resume	7, 11, 12		
rightmargin	9	P	
save-ans	4, 6, 11–17	Packages:	
save-key	11, 12, 18	enumerate	22
save-ref	4, 7, 12–14, 17	enumext	1–5, 7, 16, 22, 23
save-sep	12	enumitem	3–5, 9, 22, 23
series	7, 11, 12	fancyvrb	14
show-ans	12, 13	footnotehyper	5
show-length	8	hyperref	4, 5, 12–14, 22, 23
show-pos	12, 13, 17	l3keys	7
start*	10, 11	l3prop	1, 22
start	10, 11	l3seq	1, 22
topsep	8, 9	multicol	1, 2, 4, 23
widest	7	scontents	1, 2, 14
wrap-ans	13	task	5, 6
wrap-label*	7, 22	xsim	2
wrap-label	7	\parsep	8
wrap-opt	13	\partopsep	8
write-env	14		
		R	
L		\raggedcolumns	4
\label	4	\ref	4
Labels provide by enumext:		\rightmargin	9
\Alph*	7, 15		
\Roman*	7	T	
		\topsep	8

12 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.

12.1 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.

12.2 Initial set up

Start the DocStrip guards.

```
1 <{*package>
```

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

```
2 <@@=enumext>
```

12.3 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   {2024-10-02}
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 }
```

12.4 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:` (§12.5.1).

```

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` (§12.10) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§12.13).

```

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 (§12.13).

```

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 (§12.24).

```

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` (§12.14) and `label` (§12.12) 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 `__enumext_current_widest_dim` and others.)

```
\__enumext_leftmargin_tmp_X_bool
\__enumext_leftmargin_tmp_X_dim
\__enumext_leftmargin_X_dim
\__enumext_itemindent_X_dim
```

The boolean variable `__enumext_leftmargin_tmp_X_bool` and the dimensional variable `__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§12.17). The variables `__enumext_leftmargin_X_dim` and `__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace`:NNNNNNNNNN (§12.37.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72 {
73   \bool_new:c { \__enumext_leftmargin_tmp_#1_bool }
74   \dim_new:c { \__enumext_leftmargin_tmp_#1_dim }
75   \dim_new:c { \__enumext_leftmargin_#1_dim }
76   \dim_new:c { \__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 `__enumext_leftmargin_tmp_X_bool` and others.)

```
\__enumext_multicols_above_X_skip
\__enumext_multicols_below_X_skip
\__enumext_multicols_right_X_skip
\__enumext_align_label_pos_X_str
```

Internal variables used by `columns` key (§12.21) and `align` key (§12.12).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \skip_new:c { \__enumext_multicols_above_#1_skip }
82   \skip_new:c { \__enumext_multicols_below_#1_skip }
83   \skip_new:c { \__enumext_multicols_right_#1_skip }
84   \str_new:c { \__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 `__enumext_multicols_above_X_skip` and others.)

```
\__enumext_minipage_stat_int
\__enumext_minipage_temp_skip
\__enumext_minipage_left_skip
\__enumext_minipage_right_skip
\__enumext_minipage_after_skip
\__enumext_minipage_right_skip
\__enumext_minipage_after_skip
\__enumext_minipage_left_X_dim
\__enumext_minipage_active_X_bool
```

Internal variables used by `\miniright` command (§12.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§12.20, §12.22).

```
87 \int_new:N \__enumext_minipage_stat_int
88 \skip_new:N \__enumext_minipage_temp_skip
89 \skip_new:N \__enumext_minipage_left_skip
90 \skip_new:N \__enumext_minipage_right_skip
91 \skip_new:N \__enumext_minipage_after_skip
92 \skip_new:N \__enumext_minipage_right_skip
93 \skip_new:N \__enumext_minipage_after_skip
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95 {
96   \dim_new:c { \__enumext_minipage_left_#1_dim }
97   \bool_new:c { \__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 `__enumext_minipage_stat_int` and others.)

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

The bool vars `__enumext_wrap_label_X_bool` and `__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§12.12), the integer `__enumext_start_X_int` are used by the `start` and `start*` keys (§12.14), the token list `__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§12.17.1), the variables `__enumext_label_fill_left_X_tl` and `__enumext_label_fill_right_X_tl` are used by the `align` key (§12.12). The boolean vars `__enumext_vspace_a_star_X_bool`, `__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§12.19).

```
100 \cs_set_protected:Npn \__enumext_tmp:n #1
101 {
102   \bool_new:c { \__enumext_wrap_label_#1_bool }
103   \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
104   \int_new:c { \__enumext_start_#1_int }
105   \tl_new:c { \__enumext_fake_item_indent_#1_tl }
106   \tl_new:c { \__enumext_label_fill_left_#1_tl }
107   \tl_new:c { \__enumext_label_fill_right_#1_tl }
108   \bool_new:c { \__enumext_vspace_a_star_#1_bool }
109   \bool_new:c { \__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 `__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 (§12.25.1) 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` (§12.29) 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*` (§12.30).

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*` (§12.36) and `\anspic*` (§12.41.2) 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` (§12.47).

```

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 (§12.46) and `\foreachkeyans` command (§12.49).

```

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` (§12.46), `show-pos` key (§12.26), `item-sym*` key (§12.34), `save-key` key (§12.26.2) 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 (§12.41.1).

```

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 (§12.25.3) 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 (§12.8). 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 (§12.26). 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` (§12.8) 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` (§12.42.4).

```

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 `__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`.)

12.5 Some utility functions

`\keys_precompile:neN` Non-standard kernel variants used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command (§12.49).

`\seq_use:NV`

```

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. 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:nF`

`__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{\l__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:`)

12.5.1 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` (§12.25.1), `\l__enumext_starred_first_bool` (§12.25.1) and `\l__enumext_anskey_env_bool` (§12.30) 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:.)

12.5.2 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:`.)

12.6 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.)

12.7 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates an internal `__enumext_mini_page` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus 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 (§12.38) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§12.43)

```

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*`.)

12.8 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`.)

12.9 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

```

12.10 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.)

12.11 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`.)

12.12 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`.)

12.13 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 `\label`, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “*label and ref*” system.

12.13.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
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 }
623       { \exp_not:V \l__enumext_ref_key_arg_tl }
624     }
625   }
626 }

```

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

```

627 \cs_new_protected:Npn \__enumext_standar_ref:
628 {
629   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
630   {
631     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
632   }
633 }

```

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

12.13.2 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
634 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
635 {
636   \keys_define:nn { enumext / #1 }
637   {
638     label .code:n = {
639       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
640       { \l__enumext_counter_#2_tl } {##1}
641       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
642       \l__enumext_current_widest_dim
643     },
644     label .initial:n = #3,
645     label .value_required:n = true,
646     ref .code:n = \__enumext_starred_ref:n {##1},
647     ref .value_required:n = true,
648   }
649 }
650 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
651 \__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`.

```

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

```

```

662         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
663         \__enumext_regex_counter_style:
664         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
665         \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
666         {
667             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
668             { \exp_not:V \l__enumext_ref_key_arg_tl }
669         }
670     }
671 }
672 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
673 {
674     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
675     {
676         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
677     }
678     {
679         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
680         \__enumext_regex_counter_style:
681         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
682         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
683         {
684             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
685             { \exp_not:V \l__enumext_ref_key_arg_tl }
686         }
687     }
688 }
689 }

```

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.

```

690 \cs_new_protected:Nn \__enumext_starred_ref:
691 {
692     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
693     {
694         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
695         {
696             \tl_use:N \l__enumext_renew_the_count_vii_tl
697         }
698     }
699     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
700     {
701         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
702         {
703             \tl_use:N \l__enumext_renew_the_count_viii_tl
704         }
705     }
706 }

```

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

12.13.3 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
707 \keys_define:nn { enumext / keyans }
708 {
709     label .code:n = {
710         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
711         { \l__enumext_counter_v_tl } {#1}
712         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
713         \l__enumext_current_widest_dim
714         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
715         { \l__enumext_counter_vi_tl } {#1}
716         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
717         \l__enumext_current_widest_dim
718     },
719     label .initial:n = \Alph*,
720     label .value_required:n = true,
721     ref .code:n = \__enumext_keyans_ref:n {#1},
722     ref .value_required:n = true,
723 }

```

(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`.

```

724 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
725 {
726   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
727   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
728   {
729     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
730   }
731   {
732     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
733     \__enumext_regex_counter_style:
734     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
735     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
736     {
737       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
738       { \exp_not:V \l__enumext_ref_key_arg_tl }
739     }
740   }
741 }

```

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

```

742 \cs_new_protected:Nn \__enumext_keyans_ref:
743 {
744   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
745   {
746     \tl_use:N \l__enumext_renew_the_count_v_tl
747   }
748 }

```

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

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

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

```

749 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
750 {
751   \__enumext_if_is_int:nTF { #3 }
752   {
753     \int_set:Nn #2 {#3}
754   }
755   {
756     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
757     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
758     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
759     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
760   }
761 }
762 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

`__enumext_widest_from:nNNn`
`__enumext_widest_from:nccn`

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

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

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

```

763 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
764 {

```

```

765     \__enumext_if_is_int:nTF {#4}
766     {
767         \setcounter{enumX#1} { #4 }
768     }
769     {
770         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
771         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
772         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
773         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
774     }
775     \__enumext_label_width_by_box:cv
776     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
777 }
778 \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.

```

779 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
780 {
781     \keys_define:nn { enumext / #1 }
782     {
783         start* .code:n = {
784             \__enumext_start_from:ccn
785             { l__enumext_label_#2_tl }
786             { l__enumext_start_#2_int } {##1}
787         },
788         start* .value_required:n = true,
789         start .code:n = {
790             \__enumext_start_from:cce
791             { l__enumext_label_#2_tl }
792             { l__enumext_start_#2_int } { \int_eval:n {##1} }
793         },
794         start .initial:n = 1,
795         start .value_required:n = true,
796         widest .code:n = {
797             \__enumext_widest_from:nccn {#2}
798             { l__enumext_label_#2_tl }
799             { l__enumext_labelwidth_#2_dim } {##1}
800         },
801         widest .value_required:n = true,
802     }
803 }
804 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.15 Setting keys for vertical spaces

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

```

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

```

```

824             itemsep = 0pt, parsep= 0pt,
825             topsep = 0pt, partopsep = 0pt,
826         },
827         noseprule .value_forbidden:n = true,
828     }
829 }

```

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

```

830 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
831 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
832 { 4.0pt plus 2.0pt minus 1.0pt }
833 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
834 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
835 { 2.0pt plus 1.0pt minus 1.0pt }
836 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
837 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
838 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
839 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
840 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
841 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
842 { 2.0pt plus 1.0pt minus 1.0pt }
843 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
844 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
845 { 4.0pt plus 2.0pt minus 1.0pt }
846 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
847 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
848 { 2.0pt plus 1.0pt minus 1.0pt }

```

(End of definition for topsep and others.)

12.16 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:` and then 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.

```

849 \keys_define:nn { enumext / level-1 }
850 {
851     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
852     base-fix .initial:n = false,
853     base-fix .value_forbidden:n = true,
854 }

```

(End of definition for base-fix.)

`__enumext_nested_base_line_fix:`

The function `__enumext_nested_base_line_fix:` will be responsible for applying the *baseline correction* and adjusting the `\keys` for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§12.46). This function is passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§12.38).

```

855 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
856 {
857     \bool_lazy_all:nT
858     {
859         { \bool_if_p:N \l__enumext_starred_first_bool }
860         { \bool_if_p:N \l__enumext_base_line_fix_bool }
861         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
862     }
863     {
864         \mode_leave_vertical:
865         \vspace { -\dim_eval:n { \baselineskip + \parsep } }
866     }
867     \bool_lazy_and:nnT
868     { \bool_if_p:N \l__enumext_starred_first_bool }
869     { \bool_if_p:N \l__enumext_print_keyans_star_bool }
870     {
871         \mode_leave_vertical:
872         \skip_vertical:n { -\baselineskip }
873         \skip_vertical:N \c_zero_skip
874     }
875     \keys_set:nn { enumext / level-1 }

```

```

876     {
877         topsep = 0pt, above = 0pt, above* = 0pt,
878     }
879     \bool_set_false:N \l__enumext_base_line_fix_bool
880 }

```

(End of definition for `__enumext_nested_base_line_fix:`.)

12.17 Setting keys for horizontal spaces

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

```

881 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
882 {
883     \keys_define:nn { enumext / #1 }
884     {
885         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
886         itemindent .value_required:n = true,
887         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
888         rightmargin .value_required:n = true,
889         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
890         listparindent .value_required:n = true,
891         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
892         list-offset .value_required:n = true,
893         list-indent .code:n =
894             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
895             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
896         list-indent .value_required:n = true,
897     }
898 }
899 \clist_map_inline:nn
900 {
901     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
902 }
903 { \__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.

```

904 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
905 {
906     \keys_define:nn { enumext / #1 }
907     {
908         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
909         itemindent .value_required:n = true,
910         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
911         rightmargin .value_required:n = true,
912         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
913         listparindent .value_required:n = true,
914         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
915         list-offset .value_required:n = true,
916         list-indent .meta:n = { list-offset = ##1 },
917         list-indent .value_required:n = true,
918     }
919 }
920 \clist_map_inline:nn
921 {
922     {enumext*}{vii}, {keyans*}{viii}
923 }
924 { \__enumext_tmp:nn #1 }

```

12.17.1 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 `0pt`. 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.

```

925 \cs_set_protected:Nn \__enumext_fake_item_indent:
926 {
927     \dim_compare:nNt
928     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
929     >

```



```

930 { \c_zero_dim }
931 {
932   \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
933   {
934     \exp_not:N \mode_leave_vertical:
935     \exp_not:n { \skip_horizontal:n }
936     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
937     \ignorespaces
938   }
939 }
940 }
941 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
942 {
943   \dim_compare:nNnT
944   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
945   {
946     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
947     {
948       \exp_not:N \mode_leave_vertical:
949       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
950       \ignorespaces
951     }
952   }
953 }
954 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
955 {
956   \dim_compare:nNnT
957   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
958   {
959     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
960     {
961       \exp_not:N \mode_leave_vertical:
962       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
963       \ignorespaces
964     }
965   }
966 }
967 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
968 {
969   \dim_compare:nNnT
970   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
971   {
972     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
973     {
974       \exp_not:N \mode_leave_vertical:
975       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
976       \ignorespaces
977     }
978   }
979 }

```

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

12.18 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.

```

980 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
981 {
982   \keys_define:nn { enumext / #1 }
983   {
984     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
985     show-length .initial:n = false,
986   }
987 }
988 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19 Setting before, after and first keys

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

```

before
before*
after
first
989 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
990 {
991   \keys_define:nn { enumext / #1 }
992   {
993     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
994     before .value_required:n = true,
995     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
996     before* .value_required:n = true,
997     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
998     after .value_required:n = true,
999     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1000     first .value_required:n = true,
1001   }
1002 }
1003 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19.1 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⟩}`.

```

1004 \cs_new_protected:Nn \__enumext_before_args_exec:
1005 {
1006   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1007 }

```

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⟩}`

```

1008 \cs_new_protected:Nn \__enumext_before_keys_exec:
1009 {
1010   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1011 }

```

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

```

1012 \cs_new_protected:Nn \__enumext_after_stop_list:
1013 {
1014   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1015 }

```

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.`

```

1016 \cs_new_protected:Nn \__enumext_after_args_exec:
1017 {
1018   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1019 }

```

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

12.19.2 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:
1020 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1021 {
1022   \tl_use:N \l__enumext_before_starred_key_v_tl
1023 }
1024 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1025 {
1026   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1027 }
1028 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1029 {
1030   \tl_use:N \l__enumext_after_stop_list_v_tl
1031 }
1032 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1033 {

```

```

1034     \tl_use:N \l__enumext_after_list_args_v_tl
1035   }

```

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

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

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

```

\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1036 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1037 {
1038   \tl_use:N \l__enumext_before_starred_key_vii_tl
1039 }
1040 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1041 {
1042   \tl_use:N \l__enumext_before_starred_key_viii_tl
1043 }
1044 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1045 {
1046   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1047 }
1048 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1049 {
1050   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1051 }
1052 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1053 {
1054   \tl_use:N \l__enumext_after_stop_list_vii_tl
1055 }
1056 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1057 {
1058   \tl_use:N \l__enumext_after_stop_list_viii_tl
1059 }
1060 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1061 {
1062   \tl_use:N \l__enumext_after_list_args_vii_tl
1063 }
1064 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1065 {
1066   \tl_use:N \l__enumext_after_list_args_viii_tl
1067 }

```

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

12.20 Setting keys for `multicols` and `minipage`

`mini-env` 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.

```

1068 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1069 {
1070   \keys_define:nn { enumext / #1 }
1071   {
1072     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
1073     mini-env .value_required:n = true,
1074     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
1075     mini-sep .initial:n = 0.3333em,
1076     mini-sep .value_required:n = true,
1077     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
1078     columns-sep .value_required:n = true,
1079     columns .int_set:c = { \l__enumext_columns_#2_int },
1080     columns .initial:n = 1,
1081     columns .value_required:n = true,
1082   }
1083 }
1084 \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.

```

1085 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1086 {
1087   \keys_define:nn { enumext / #1 }

```

```

1088     {
1089         mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1090         mini-right .value_required:n = true,
1091         mini-right* .code:n = {
1092             \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1093             \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1094         },
1095         mini-right* .value_required:n = true,
1096     }
1097 }
1098 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

12.21 Adjustment of vertical spaces for multicol

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



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

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

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

12.21.1 Adjustment of vertical spaces for multicol in enumext

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

We will set the default values taking into account that $\text{T}_{\text{E}}\text{X}$ is in *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

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

```

1099 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1100 {
1101     \skip_set:cn { l__enumext_multicol_above_ \__enumext_level: _skip }
1102     {
1103         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1104     }
1105     \skip_set:cn { l__enumext_multicol_below_ \__enumext_level: _skip }
1106     {
1107         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1108     }
1109     \__enumext_add_pre_parsep:
1110 }

```

(End of definition for `__enumext_multi_set_vskip:`.)

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

```

1111 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1112 {
1113     \int_case:nn { \l__enumext_level_int }
1114     {
1115         { 2 }{
1116             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1117             {
1118                 \skip_add:Nn \l__enumext_multicol_above_ii_skip
1119                 {

```

```

1120         \l__enumext_parsep_i_skip
1121     }
1122 }
1123 }
1124 { 3 }{
1125     \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1126     {
1127         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1128         {
1129             \l__enumext_parsep_ii_skip
1130         }
1131     }
1132 }
1133 { 4 }{
1134     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1135     {
1136         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1137         {
1138             \l__enumext_parsep_iii_skip
1139         }
1140     }
1141 }
1142 }
1143 }

```

(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 T_EX is in *horizontal mode* or *vertical mode*.

```

1144 \cs_new_protected:Nn \__enumext_multi_addvspace:
1145 {
1146     \__enumext_multi_set_vskip:
1147     \mode_if_vertical:T
1148     {
1149         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1150         {
1151             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1152         }
1153         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1154         {
1155             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1156         }
1157     }
1158     \par\nopagebreak
1159     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1160 }

```

(End of definition for `__enumext_multi_addvspace:`.)

12.21.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1161 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1162 {
1163     \skip_set:Nn \l__enumext_multicols_above_v_skip
1164     {
1165         \l__enumext_topsep_v_skip
1166     }
1167     \skip_set:Nn \l__enumext_multicols_below_v_skip
1168     {
1169         \l__enumext_topsep_v_skip
1170     }
1171 }
1172 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1173 {
1174     \__enumext_keyans_multi_set_vskip:
1175     \mode_if_vertical:T
1176     {
1177         \skip_add:Nn \l__enumext_multicols_above_v_skip

```

```

1178     {
1179         \skip_use:N \l__enumext_partopsep_v_skip
1180     }
1181     \skip_add:Nn \l__enumext_multicols_below_v_skip
1182     {
1183         \skip_use:N \l__enumext_partopsep_v_skip
1184     }
1185 }
1186 \par\nopagebreak
1187 \addvspace{ \l__enumext_multicols_above_v_skip }
1188 }

```

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

12.22 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.

12.22.1 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`.

```

1189 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1190 {
1191     \skip_set:Nn \l__enumext_minipage_right_skip
1192     {
1193         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1194     }
1195     \mode_if_vertical:T
1196     {
1197         \skip_add:Nn \l__enumext_minipage_right_skip
1198         {
1199             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1200         }
1201     }
1202     \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:`.

```

1203     \skip_set_eq:cN
1204     { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1205     \skip_set_eq:cN
1206     { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1207     \__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`.

```

1208 \int_compare:nNt
1209 { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1210 {
1211   \skip_zero:N \topskip
1212   \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1213 }
1214 }

```

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

```

1215 \cs_new_protected:Nn \__enumext_minipage_add_space:
1216 {
1217   \__enumext_minipage_set_skip:
1218   \__enumext_unskip_unkern:
1219   \mode_if_vertical:TF
1220   {
1221     \nopagebreak\nointerlineskip
1222   }
1223   {
1224     \par\nopagebreak\nointerlineskip
1225     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1226   }
1227   \int_compare:nNtTF
1228   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1229   {
1230     \addvspace{ 0.445\box_ht:N \strutbox }
1231   }
1232   {
1233     \addvspace{ 0.250\box_ht:N \strutbox }
1234   }
1235 }

```

(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.

```

1236 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1237 {
1238   \int_case:nn { \l__enumext_level_int }
1239   {
1240     { 2 }{
1241       \skip_if_eq:nnTF
1242       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1243       {
1244         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1245         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1246       }
1247       {
1248         \dim_compare:nNt
1249         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1250         {
1251           \skip_sub:Nn
1252           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1253           \skip_sub:Nn
1254           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1255           \skip_add:Nn
1256           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1257           \skip_add:Nn
1258           \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1259         }
1260         \dim_compare:nNt
1261         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1262         {
1263           \skip_set:Nn \l__enumext_minipage_temp_skip
1264           {

```

```

1265         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1266     }
1267     \skip_sub:Nn
1268     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1269     \skip_sub:Nn
1270     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1271     \skip_add:Nn
1272     \l__enumext_minipage_after_skip
1273     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1274     \skip_add:Nn
1275     \l__enumext_multicols_below_ii_skip
1276     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1277 }
1278 }
1279 }
1280 { 3 }{
1281     \skip_if_eq:nnTF
1282     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1283     {
1284         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1285         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1286     }
1287     {
1288         \dim_compare:nNnT
1289         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1290         {
1291             \skip_sub:Nn
1292             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1293             \skip_sub:Nn
1294             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1295             \skip_add:Nn
1296             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1297             \skip_add:Nn
1298             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1299         }
1300         \dim_compare:nNnT
1301         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1302         {
1303             \skip_set:Nn \l__enumext_minipage_temp_skip
1304             {
1305                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1306             }
1307             \skip_sub:Nn
1308             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1309             \skip_sub:Nn
1310             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1311             \skip_add:Nn
1312             \l__enumext_minipage_after_skip
1313             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1314             \skip_add:Nn
1315             \l__enumext_multicols_below_iii_skip
1316             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1317         }
1318     }
1319 }
1320 { 4 }{
1321     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1322     {
1323         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1324         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1325     }
1326     {
1327         \dim_compare:nNnT
1328         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1329         {
1330             \skip_sub:Nn
1331             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1332             \skip_sub:Nn
1333             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1334             \skip_add:Nn
1335             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1336         \skip_add:Nn
1337         \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1338     }
1339     \dim_compare:nNtT
1340     { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1341     {
1342         \skip_set:Nn \l__enumext_minipage_temp_skip
1343         {
1344             \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1345         }
1346         \skip_sub:Nn
1347         \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1348         \skip_sub:Nn
1349         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1350         \skip_add:Nn
1351         \l__enumext_minipage_after_skip
1352         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1353         \skip_add:Nn
1354         \l__enumext_multicols_below_iv_skip
1355         { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1356     }
1357 }
1358 }
1359 }
1360 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

12.22.2 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](#).

```

1361 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1362 {
1363     \skip_zero:N \l__enumext_minipage_after_skip
1364     \skip_zero:N \l__enumext_minipage_left_skip
1365     \skip_zero:N \l__enumext_minipage_right_skip
1366     \skip_set:Nn \l__enumext_minipage_right_skip
1367     {
1368         \l__enumext_topsep_v_skip
1369     }
1370     \mode_if_vertical:T
1371     {
1372         \skip_add:Nn \l__enumext_minipage_right_skip
1373         {
1374             \l__enumext_partopsep_v_skip
1375         }
1376     }
1377     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1378     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1379     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1380     \__enumext_keyans_pre_itemsep_skip:
1381     \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
1382     {
1383         \skip_zero:N \topskip
1384         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1385     }
1386 }
1387 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1388 {
1389     \__enumext_keyans_minipage_set_skip:
1390     \__enumext_unskip_unkern:
1391     \mode_if_vertical:TF
1392     {
1393         \nopagebreak\nointerlineskip
1394     }
1395     {
1396         \par\nopagebreak\nointerlineskip
1397         \skip_zero:N \l__enumext_partopsep_v_skip
1398     }
1399     \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }

```

```

1400     {
1401         \addvspace{ 0.445\box_ht:N \strutbox }
1402     }
1403     {
1404         \addvspace{ 0.250\box_ht:N \strutbox }
1405     }
1406 }
1407 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1408 {
1409     \skip_if_eq:nnTF
1410     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1411     {
1412         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1413         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1414     }
1415     {
1416         \dim_compare:nNnT
1417         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1418         {
1419             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1420             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1421             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1422             \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1423         }
1424         \dim_compare:nNnT
1425         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1426         {
1427             \skip_set:Nn \l__enumext_minipage_temp_skip
1428             {
1429                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1430             }
1431             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1432             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1433             \skip_add:Nn \l__enumext_minipage_after_skip
1434             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1435             \skip_add:Nn \l__enumext_multicols_below_v_skip
1436             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1437         }
1438     }
1439 }

```

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

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

The functions `__enumext_mini_set_vskip_vii:` and `__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `__enumext_mini_page` environment in `enumext*` and `keyans*`.

```

1440 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1441 {
1442     \skip_zero_new:N \l__enumext_minipage_left_skip
1443     \skip_gzero_new:N \g__enumext_minipage_right_skip
1444     \skip_gzero_new:N \g__enumext_minipage_after_skip
1445     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1446     {
1447         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1448         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1449     }
1450     {
1451         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1452         \skip_gset:Nn \g__enumext_minipage_right_skip
1453         {
1454             \l__enumext_topsep_vii_skip
1455         }
1456         \skip_gset:Nn \g__enumext_minipage_after_skip
1457         {
1458             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1459         }
1460     }
1461 }
1462 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:

```

```

1463 {
1464   \skip_zero_new:N \l__enumext_minipage_after_skip
1465   \skip_zero_new:N \l__enumext_minipage_left_skip
1466   \skip_zero_new:N \l__enumext_minipage_right_skip
1467   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1468   {
1469     \skip_set:Nn \l__enumext_minipage_left_skip
1470     {
1471       0.5\box_dp:N \strutbox
1472     }
1473     \skip_set:Nn \l__enumext_minipage_right_skip
1474     {
1475       \l__enumext_partopsep_viii_skip
1476     }
1477     \skip_set:Nn \l__enumext_minipage_after_skip
1478     {
1479       1.6\box_dp:N \strutbox
1480     }
1481   }
1482   {
1483     \skip_set:Nn \l__enumext_minipage_left_skip
1484     {
1485       0.5875\box_dp:N \strutbox
1486     }
1487     \skip_set:Nn \l__enumext_minipage_right_skip
1488     {
1489       \l__enumext_topsep_viii_skip
1490     }
1491     \skip_set:Nn \l__enumext_minipage_after_skip
1492     {
1493       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1494     }
1495   }
1496 }

```

(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 $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$, since `\partopsep` is equal to `0pt` in both environments.

```

1497 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1498 {
1499   \__enumext_mini_set_vskip_vii:
1500   \par\nopagebreak
1501   \addvspace { \l__enumext_minipage_left_skip }
1502 }
1503 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1504 {
1505   \__enumext_mini_set_vskip_viii:
1506   \par\nopagebreak
1507   \addvspace { \l__enumext_minipage_left_skip }
1508 }

```

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

12.22.4 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 \TeX justification is maintained in the `__enumext_mini_page` on the “right side”.

`\miniright`

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

```

1509 \NewDocumentCommand \miniright { s }
1510 {
1511   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

1512     {
1513       \msg_error:nnn { enumext } { wrong-miniright-place }
1514     }
1515   % outside
1516   \bool_lazy_and:nnT
1517   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1518   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1519   {
1520     \msg_error:nnn { enumext } { wrong-miniright-place }
1521   }
1522   % starred env
1523   \bool_if:NT \l__enumext_starred_bool
1524   {
1525     \msg_error:nnn { enumext } { wrong-miniright-starred }
1526   }
1527   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1528   {
1529     \__enumext_keyans_mini_right_cmd:n {#1}
1530   }
1531   { \__enumext_mini_right_cmd:n {#1} }
1532 }

```

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

`__enumext_mini_right_cmd:n`

The function `__enumext_mini_right_cmd:n` takes as argument the *starred* ‘`*`’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\l__enumext_minipage_right_X_dim`, if so we close the `multicols` environment with the `__enumext_mini_page` environment on the “left side”, then we open the `__enumext_mini_page` environment on the “right side”, apply our adjusted “vertical spaces”, followed by adding the `\centering` command when the *starred* argument ‘`*`’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1533 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1534 {
1535   \dim_compare:nNnTF
1536   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1537   {
1538     \__enumext_multicols_stop:
1539     \int_compare:nNnT
1540     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1541     {
1542       \par\addvspace{ \l__enumext_minipage_after_skip }
1543     }
1544     \end__enumext_mini_page
1545     \hfill
1546     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1547     \par\nointerlineskip
1548     \addvspace { \l__enumext_minipage_right_skip }
1549     \bool_if:nF {#1}
1550     {
1551       \centering
1552     }
1553     \int_gzero:N \g__enumext_minipage_stat_int
1554   }
1555   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1556   % paranoia
1557   \RenewDocumentCommand \miniright { s }
1558   {
1559     \msg_error:nn { enumext } { many-miniright-used }
1560   }
1561 }

```

(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.

```

1562 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1563 {
1564   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1565   {
1566     \__enumext_keyans_multicols_stop:
1567     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }

```



```

1568         {
1569             \par\addvspace{ \l__enumext_minipage_after_skip }
1570         }
1571     \end__enumext_mini_page
1572     \hfill
1573     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1574     \par\nointerlineskip
1575     \addvspace { \l__enumext_minipage_right_skip }
1576     \bool_if:nF {#1}
1577     {
1578         \centering
1579     }
1580     \int_gzero:N \g__enumext_minipage_stat_int
1581 }
1582 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1583 % paranoia
1584 \RenewDocumentCommand \miniright { s }
1585 {
1586     \msg_error:nn { enumext } { many-miniright-used }
1587 }
1588 }

```

(End of definition for __enumext_keyans_mini_right_cmd:n.)

12.23 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.

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

```

above* 1589 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1590 {
below* 1591     \keys_define:nn { enumext / #1 }
1592     {
1593         above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1594         above .value_required:n = true,
1595         above* .code:n = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1596             \keys_set:nn { enumext / #1 } { above = {##1} },
1597         above* .value_required:n = true,
1598         below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1599         below .value_required:n = true,
1600         below* .code:n = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1601             \keys_set:nn { enumext / #1 } { below = {##1} },
1602         below* .value_required:n = true,
1603     }
1604 }
1605 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

12.23.1 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.

```

1606 \cs_new_protected:Nn \__enumext_vspace_above:
1607 {
1608     \skip_if_eq:nnF
1609     { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1610     {
1611         \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1612         {
1613             \vspace*{ \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1614         }
1615         {
1616             \vspace { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1617         }
1618     }
1619 }

```

(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.

```

1620 \cs_new_protected:Nn \__enumext_vspace_below:
1621 {
1622   \skip_if_eq:nnF
1623   { \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1624   {
1625     \bool_if:cTF { \l__enumext_vspace_b_star_ \__enumext_level: _bool }
1626     {
1627       \vspace*{ \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } }
1628     }
1629     {
1630       \vspace { \skip_use:c { \l__enumext_vspace_below_ \__enumext_level: _skip } }
1631     }
1632   }
1633 }

```

(End of definition for `__enumext_vspace_below:`.)

12.23.2 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.

```

1634 \cs_new_protected:Nn \__enumext_vspace_above_v:
1635 {
1636   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1637   {
1638     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1639     {
1640       \vspace*{ \l__enumext_vspace_above_v_skip }
1641     }
1642     { \vspace { \l__enumext_vspace_above_v_skip } }
1643   }
1644 }

```

(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.

```

1645 \cs_new_protected:Nn \__enumext_vspace_below_v:
1646 {
1647   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1648   {
1649     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1650     {
1651       \vspace*{ \l__enumext_vspace_below_v_skip }
1652     }
1653     { \vspace { \l__enumext_vspace_below_v_skip } }
1654   }
1655 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

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

`__enumext_vspace_above_viii:`

```

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

```

```

1671         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1672         {
1673             \vspace*{ \l__enumext_vspace_above_viii_skip }
1674         }
1675         { \vspace { \l__enumext_vspace_above_viii_skip } }
1676     }
1677 }

```

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

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

`__enumext_vspace_below_viii:`

```

1678 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1679 {
1680     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1681     {
1682         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1683         {
1684             \vspace*{ \l__enumext_vspace_below_vii_skip }
1685         }
1686         { \vspace { \l__enumext_vspace_below_vii_skip } }
1687     }
1688 }
1689 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1690 {
1691     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1692     {
1693         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1694         {
1695             \vspace*{ \l__enumext_vspace_below_viii_skip }
1696         }
1697         { \vspace { \l__enumext_vspace_below_viii_skip } }
1698     }
1699 }

```

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

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

`series`
`resume`
`resume*`

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

```

1700 \cs_set_protected:Npn \__enumext_tmp:n #1
1701 {
1702     \keys_define:nn { enumext / #1 }
1703     {
1704         series .str_set:N = \l__enumext_series_str,
1705         series .value_required:n = true,
1706         resume .code:n = \__enumext_resume_series:n {##1},
1707         resume* .code:n = \__enumext_resume_starred:,
1708         resume* .value_forbidden:n = true,
1709     }
1710 }
1711 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

12.24.1 Internal functions for series key

`__enumext_filter_series:n`
`__enumext_filter_series_key:n`
`__enumext_filter_series_pair:nn`

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

```

1712 \cs_new:Npn \__enumext_filter_series:n #1
1713 {
1714     \use:e
1715     {
1716         \keyval_parse:NNn
1717         \__enumext_filter_series_key:n
1718         \__enumext_filter_series_pair:nn {#1}
1719     }
1720 }

```

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.

```

1721 \cs_new:Npn \__enumext_filter_series_key:n #1
1722 {
1723   \str_case:nnF {#1}
1724   {
1725     { resume } {} { resume* } {} { base-fix } {}
1726   }
1727   { , { \exp_not:n {#1} } }
1728 }

```

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.

```

1729 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1730 {
1731   \str_case:nnF {#1}
1732   {
1733     { series } {} { resume } {} { start } {}
1734     { start* } {} { save-ans } {} { save-key } {}
1735   }
1736   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1737 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§12.38) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§12.43).

```

1738 \cs_new_protected:Npn \__enumext_parse_series:n #1
1739 {
1740   \str_if_empty:NTF \l__enumext_series_str
1741   {
1742     \bool_if:NF \l__enumext_resume_active_bool
1743     {
1744       \__enumext_resume_last:n {#1}
1745     }
1746   }
1747   {
1748     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1749     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1750     { \__enumext_filter_series:n {#1} }
1751     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1752     {
1753       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1754     }
1755   }
1756 }

```

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.

```

1757 \cs_new_protected:Npn \__enumext_resume_last:n #1
1758 {
1759   \bool_if:NT \l__enumext_standar_first_bool
1760   {
1761     \tl_gclear:N \g__enumext_standar_series_tl
1762     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1763   }
1764   \bool_if:NT \l__enumext_starred_first_bool
1765   {
1766     \tl_gclear:N \g__enumext_starred_series_tl
1767     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1768   }
1769 }

```

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

12.2.4.2 Internal function to save counter value

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§12.38) and the `enumext*` environment definition (§12.43).

```

1770 \cs_new_protected:Nn \__enumext_resume_save_counter:
1771 {
1772   \bool_if:NT \g__enumext_standar_bool
1773   {
1774     \tl_if_empty:NF \l__enumext_series_str
1775     {
1776       \int_gset_eq:cN
1777       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1778     }
1779     \tl_if_empty:NTF \l__enumext_resume_name_tl
1780     {
1781       \str_if_empty:NT \l__enumext_series_str
1782       {
1783         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1784       }
1785     }
1786     {
1787       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1788       {
1789         \int_gset_eq:cN
1790         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1791       }
1792     }
1793     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1794     {
1795       \int_gset_eq:cN
1796       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1797     }
1798   }
1799   \bool_if:NT \g__enumext_starred_bool
1800   {
1801     \tl_if_empty:NF \l__enumext_series_str
1802     {
1803       \int_gset_eq:cN
1804       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1805     }
1806     \tl_if_empty:NTF \l__enumext_resume_name_tl
1807     {
1808       \str_if_empty:NT \l__enumext_series_str
1809       {
1810         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1811       }
1812     }
1813     {
1814       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1815       {
1816         \int_gset_eq:cN
1817         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1818       }
1819     }
1820     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1821     {
1822       \int_gset_eq:cN
1823       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1824     }
1825   }
1826 }

```

(End of definition for `__enumext_resume_save_counter:`.)

12.24.3 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.

```

1827 \cs_new_protected:Npn \__enumext_resume_series:n #1
1828 {
1829   \tl_if_empty:nTF {#1}
1830   {
1831     \__enumext_resume_counter:n {#1}
1832   }
1833   {
1834     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1835     {
1836       \__enumext_resume_counter:n {#1}
1837       \bool_if:NT \g__enumext_standar_bool
1838       {
1839         \keys_set:nv { enumext / level-1 }
1840         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1841       }
1842       \bool_if:NT \g__enumext_starred_bool
1843       {
1844         \keys_set:nv { enumext / enumext* }
1845         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1846       }
1847     }
1848     {
1849       \bool_if:NT \g__enumext_standar_bool
1850       {
1851         \msg_error:nnn { enumext } { unknown-series } {#1}
1852       }
1853       \bool_if:NT \g__enumext_starred_bool
1854       {
1855         \msg_error:nnn { enumext } { unknown-series } {#1}
1856       }
1857     }
1858   }
1859 }

```

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

`__enumext_resume_counter:n`

`__enumext_resume_counter:`

`__enumext_resume_counter_series:`

`__enumext_resume_counter_save_ans:`

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

```

1860 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1861 {
1862   \bool_set_true:N \l__enumext_resume_active_bool
1863   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1864   \tl_if_empty:NTF \l__enumext_resume_name_tl
1865   {
1866     \__enumext_resume_counter:
1867   }
1868   {
1869     \__enumext_resume_counter_series:
1870   }
1871   \__enumext_resume_counter_save_ans:
1872 }

```

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.

```

1873 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1874 {
1875   \bool_if:NT \g__enumext_standar_bool
1876   {
1877     \int_gincr:N \g__enumext_resume_int
1878     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1879   }
1880   \bool_if:NT \g__enumext_starred_bool
1881   {
1882     \int_gincr:N \g__enumext_resume_vii_int
1883     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1884   }
1885 }

```

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

```

1886 \cs_new_protected:Nn \__enumext_resume_counter_series:
1887 {
1888   \bool_if:NT \g__enumext_standar_bool
1889   {
1890     \int_set:Nn \l__enumext_start_i_int
1891     {
1892       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1893     }
1894   }
1895   \bool_if:NT \g__enumext_starred_bool
1896   {
1897     \int_set:Nn \l__enumext_start_vii_int
1898     {
1899       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1900     }
1901   }
1902 }

```

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.

```

1903 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1904 {
1905   \bool_lazy_and:nnT
1906   { \bool_if_p:N \l__enumext_standar_first_bool }
1907   { \bool_if_p:N \l__enumext_store_active_bool }
1908   {
1909     \int_set:Nn \l__enumext_start_i_int
1910     {
1911       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1912     }
1913   }
1914   \bool_lazy_and:nnT
1915   { \bool_if_p:N \l__enumext_starred_first_bool }
1916   { \bool_if_p:N \l__enumext_store_active_bool }
1917   {
1918     \int_set:Nn \l__enumext_start_vii_int
1919     {
1920       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1921     }
1922   }
1923 }

```

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

12.24.4 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.

```

1924 \cs_new_protected:Nn \__enumext_resume_starred:
1925 {
1926   \bool_if:NT \g__enumext_standar_bool
1927   {
1928     \tl_if_empty:NF \g__enumext_standar_series_tl

```



```

1929         {
1930             \__enumext_resume_counter:n { }
1931             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1932         }
1933     }
1934     \bool_if:NT \g__enumext_starred_bool
1935     {
1936         \tl_if_empty:NF \g__enumext_starred_series_tl
1937         {
1938             \__enumext_resume_counter:n { }
1939             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1940         }
1941     }
1942 }

```

(End of definition for __enumext_resume_starred:.)

12.25 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.

12.25.1 Setting save-ans key

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

```

1943 \cs_set_protected:Npn \__enumext_tmp:n #1
1944 {
1945     \keys_define:nn { enumext / #1 }
1946     {
1947         save-ans .code:n = \__enumext_storing_set:n {##1},
1948         save-ans .value_required:n = true,
1949     }
1950 }
1951 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

12.25.2 Internal functions for save-ans key

`__enumext_start_save_ans_msg:`
`__enumext_stop_save_ans_msg:`

The functions `__enumext_start_save_ans_msg:` and `__enumext_stop_save_ans_msg:` will display in the terminal and `.log` file the environment in which the `save-ans` key was executed along with the line at the beginning and end of it. The function `__enumext_start_save_ans_msg:` will be passed to `__enumext_storing_set:n` and the function `__enumext_stop_save_ans_msg:` will be passed to the function `__enumext_execute_after_env:`.

```

1952 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1953 {
1954     \msg_term:nnVV { enumext } { save-ans-log }
1955     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1956 }
1957 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1958 {
1959     \msg_term:nnVV { enumext } { save-ans-log-hook }
1960     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1961 }

```

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

`__enumext_storing_set:n`
`__enumext_storing_exec:`

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

```

1962 \cs_new_protected:Npn \__enumext_storing_set:n #1
1963 {
1964     \tl_set:Nc \l__enumext_store_name_tl {#1}
1965     \tl_if_empty:NTF \l__enumext_store_name_tl
1966     {
1967         \bool_lazy_or:nnT
1968         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1969         {
1970             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1971         }
1972     }

```

```

1973     {
1974         \bool_lazy_or:nnT
1975         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1976         {
1977             \__enumext_start_save_ans_msg:
1978             \__enumext_storing_exec:
1979         }
1980     }
1981 }

```

The function `__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to “true” the variable `\l__enumext_check_answers_bool` used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, copy `{⟨store name⟩}` into the variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§12.30).

```

1982 \cs_new_protected:Nn \__enumext_storing_exec:
1983 {
1984     \bool_set_true:N \l__enumext_store_active_bool
1985     \bool_set_true:N \l__enumext_check_answers_bool
1986     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1987     \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

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

```

1988     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1989     {
1990         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1991         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1992     }
1993     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1994     {
1995         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1996         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1997     }
1998     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1999     {
2000         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2001         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2002     }
2003 }

```

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

12.25.3 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.

12.25.4 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

no-store

```

2004 \cs_set_protected:Npn \__enumext_tmp:n #1
2005 {
2006   \keys_define:nn { enumext / #1 }
2007   {
2008     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2009     check-ans .initial:n = false,
2010     check-ans .value_required:n = true,
2011     no-store .code:n = {
2012       \bool_set_false:N \l__enumext_check_answers_bool
2013       \bool_set_false:N \l__enumext_check_ans_key_bool
2014     },
2015     no-store .value_forbidden:n = true,
2016   }
2017 }
2018 \clist_map_inline:nn
2019 {
2020   level-1, level-2, level-3, level-4, enumext*
2021 }
2022 { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

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.

```

2023 \cs_new_protected:Nn \__enumext_check_ans_active:
2024 {
2025   \tl_if_empty:NF \l__enumext_store_name_tl
2026   {
2027     \bool_if:NT \l__enumext_check_answers_bool
2028     {
2029       \__enumext_check_ans_level:
2030     }
2031   }
2032 }
```

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”.

```

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

```

2056         { 4 }{
2057             \int_gdecr:N \g__enumext_item_number_int
2058             \bool_set_false:N \l__enumext_item_number_bool
2059         }
2060     }

```

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.

```

2061     \int_case:nn { \l__enumext_level_h_int }
2062     {
2063         { 1 }{
2064             \bool_lazy_all:nT
2065             {
2066                 { \bool_if_p:N \g__enumext_standar_bool }
2067                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2068             }
2069             {
2070                 \int_gdecr:N \g__enumext_item_number_int
2071                 \bool_set_false:N \l__enumext_item_number_bool
2072             }
2073         }
2074     }
2075 }

```

(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.

```

2076 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2077 {
2078     \bool_lazy_and:nnT
2079     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2080     { \bool_if_p:N \g__enumext_standar_bool }
2081     {
2082         \bool_gset_true:N \g__enumext_check_ans_key_bool
2083     }
2084     \bool_lazy_and:nnT
2085     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2086     { \bool_if_p:N \g__enumext_starred_bool }
2087     {
2088         \bool_gset_true:N \g__enumext_check_ans_key_bool
2089     }
2090 }

```

(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:`.

```

2091 \cs_new_protected:Nn \__enumext_item_answer_diff:
2092 {
2093     \int_gset:Nn \g__enumext_item_answer_diff_int
2094     {
2095         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2096     }
2097 }

```

(End of definition for `__enumext_item_answer_diff:`)

`__enumext_check_ans_show:`

`__enumext_check_ans_msg_less:`

`__enumext_check_ans_msg_same_ok:`

`__enumext_check_ans_msg_greater:`

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

```

2098 \cs_new_protected:Nn \__enumext_check_ans_show:
2099 {
2100     \int_case:nn { \g__enumext_item_answer_diff_int }
2101     {
2102         { -1 }{ \__enumext_check_ans_msg_less: }
2103         { 0 }{ \__enumext_check_ans_msg_same_ok: }

```

```

2104         { 1 } { \__enumext_check_ans_msg_greater: }
2105     }
2106 }
2107 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2108 {
2109     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2110     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2111 }
2112 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2113 {
2114     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2115     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2116 }
2117 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2118 {
2119     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2120     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2121 }

```

(End of definition for __enumext_check_ans_show: and others.)

```

\__enumext_check_ans_log:
\__enumext_check_ans_log_msg_less:
\__enumext_check_ans_log_msg_same_ok:
\__enumext_check_ans_log_msg_greater:

```

The function __enumext_check_ans_log: will be executed within 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:`.

```

2122 \cs_new_protected:Nn \__enumext_check_ans_log:
2123 {
2124     \int_case:nn { \g__enumext_item_answer_diff_int }
2125     {
2126         { -1 } { \__enumext_check_ans_log_msg_less: }
2127         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2128         { 1 } { \__enumext_check_ans_log_msg_greater: }
2129     }
2130 }
2131 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2132 {
2133     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2134     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2135 }
2136 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2137 {
2138     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2139     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2140 }
2141 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2142 {
2143     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2144     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2145 }

```

(End of definition for __enumext_check_ans_log: and others.)

12.25.6 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.

```

2146 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2147 {
2148     \int_compare:nNnT
2149     { \g__enumext_check_starred_cmd_int } = { 0 }
2150     {
2151         \msg_warning:nnnV
2152         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2153     }
2154     \int_compare:nNnT
2155     { \g__enumext_check_starred_cmd_int } > { 1 }
2156     {
2157         \msg_warning:nnnV
2158         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2159     }
2160     \int_gzero:N \g__enumext_check_starred_cmd_int

```

```

2161 \tl_clear:N \l__enumext_check_start_line_env_tl
2162 }

```

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

12.26 Keys and functions associated with storage

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*`.

```

2163 \cs_set_protected:Npn \__enumext_tmp:n #1
2164 {
2165   \keys_define:nn { enumext / #1 }
2166   {
2167     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2168     wrap-ans .initial:n =
2169       {
2170         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2171       },
2172     wrap-ans .value_required:n = true,
2173     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2174     wrap-opt .initial:n = [{##1}],
2175     wrap-opt .value_required:n = true,
2176     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2177     save-sep .initial:n = {, ~ },
2178     save-sep .value_required:n = true,
2179     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2180     mark-ans .initial:n = \textasteriskcentered,
2181     mark-ans .value_required:n = true,
2182     mark-pos .choice:,
2183     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2184     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2185     mark-pos / unknown .code:n =
2186       \msg_error:nneee { enumext } { unknown-choice }
2187       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2188     mark-pos .initial:n = right,
2189     mark-pos .value_required:n = true,
2190     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2191     show-ans .initial:n = false,
2192     show-ans .value_required:n = true,
2193     show-pos .bool_set:N = \l__enumext_show_position_bool,
2194     show-pos .initial:n = false,
2195     show-pos .value_required:n = true,
2196     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2197     mark-ref .initial:n = \textasteriskcentered,
2198     mark-ref .value_required:n = true,
2199     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2200     save-ref .initial:n = false,
2201     save-ref .value_required:n = true,
2202   }
2203 }
2204 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

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

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

```

2205 \cs_set_protected:Npn \__enumext_tmp:n #1
2206 {
2207   \keys_define:nn { enumext / #1 }
2208   {
2209     mark-pos .choice:,
2210     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2211     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2212     mark-pos .initial:n = right,
2213     mark-pos .value_required:n = true,
2214     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2215     show-ans .initial:n = false,
2216     show-ans .value_required:n = true,
2217     show-pos .bool_set:N = \l__enumext_show_position_bool,
2218     show-pos .initial:n = false,
2219     show-pos .value_required:n = true,
2220   }

```

```

2221     }
2222     \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

12.26.1 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*.

```

2223 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2224 {
2225     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2226     {
2227         \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2228         \tl_set:ce
2229             { \l__enumext_store_save_key_ \__enumext_level: _tl }
2230             { \__enumext_filter_save_key:n {#1} }
2231     }
2232 }
2233 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2234 {
2235     \bool_if:NF \l__enumext_store_save_key_vii_bool
2236     {
2237         \tl_clear:N \l__enumext_store_save_key_vii_tl
2238         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2239     }
2240 }

```

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

12.26.2 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.

```

2241 \cs_set_protected:Npn \__enumext_tmp:n #1
2242 {
2243     \keys_define:nn { enumext / enumext* }
2244     {
2245         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2246         save-key .value_required:n = true,
2247     }
2248     \keys_define:nn { enumext / #1 }
2249     {
2250         save-key .code:n = \__enumext_parse_save_key:n {##1},
2251         save-key .value_required:n = true,
2252     }
2253 }
2254 \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*`.

```

2255 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2256 {
2257     \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2258     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2259     \tl_set:ce

```



```

2260     { l__enumext_store_save_key_ \__enumext_level: _tl }
2261     { \__enumext_filter_save_key:n {#1} }
2262   }
2263   \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2264   {
2265     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2266     \tl_clear:N \l__enumext_store_save_key_vii_tl
2267     \tl_set:Nx \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2268   }

```

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

12.26.3 Internal functions to store optional arguments

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

```

2269 \cs_new:Npn \__enumext_filter_save_key:n #1
2270 {
2271   \use:e
2272   {
2273     \keyval_parse:NNn
2274     \__enumext_filter_save_key_key:n
2275     \__enumext_filter_save_key_pair:nn {#1}
2276   }
2277 }

```

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.

```

2278 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2279 {
2280   \str_case:nnF {#1}
2281   {
2282     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2283   }
2284   { , { \exp_not:n {#1} } }
2285 }

```

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.

```

2286 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2287 {
2288   \str_case:nnF {#1}
2289   {
2290     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2291     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2292     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2293     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2294     { mini-right* } {}
2295   }
2296   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2297 }

```

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

12.26.4 Function for storing content in prop list

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.

```

2298 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2299 {
2300   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2301   {
2302     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2303   }
2304   { #1 }
2305 }
2306 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

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

12.26.5 Function for storing content in sequence

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

`__enumext_store_addto_seq:v` The form in which the $\{\langle content \rangle\}$ is stored in *sequence* is in a internal `enumext` or `enumext*` environments with the “*same structure*” in which the command was executed.

The “*stored content*” is retrieved by means of the `\printkeyans` command.

```
2307 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2308 {
2309     \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2310 }
2311 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

12.26.6 Functions for storing structure in the sequence

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

```
2312 \cs_new_protected:Nn \__enumext_store_level_open:
2313 {
2314     \bool_if:NT \l__enumext_check_answers_bool
2315     {
2316         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2317         {
2318             \__enumext_store_addto_seq:n
2319             {
2320                 \item \begin{enumext}
2321             }
2322         }
2323         {
2324             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2325             {
2326                 \item \begin{enumext} [
2327             }
2328             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2329             {
2330                 ]
2331             }
2332             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2333         }
2334     }
2335 }
2336 \cs_new_protected:Nn \__enumext_store_level_close:
2337 {
2338     \bool_if:NT \l__enumext_check_answers_bool
2339     {
2340         \__enumext_store_addto_seq:n { \end{enumext} }
2341     }
2342 }
```

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

`__enumext_store_level_open_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.

```
2343 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2344 {
2345     \bool_if:NT \l__enumext_check_answers_bool
2346     {
2347         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2348         {
2349             \__enumext_store_addto_seq:n
2350             {
2351                 \item \begin{enumext*}
2352             }
2353         }
2354         {
2355             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2356             {
2357                 \item \begin{enumext*}[
```

```

2358         }
2359         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2360         {
2361             ]
2362         }
2363         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2364     }
2365 }
2366 }
2367 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2368 {
2369     \bool_if:NT \l__enumext_check_answers_bool
2370     {
2371         \__enumext_store_addto_seq:n { \end{enumext*} }
2372     }
2373 }

```

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

12.26.7 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

```

2374 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2375 {
2376     \mode_leave_vertical:
2377     \skip_horizontal:n { -\dim_use:N #2 }
2378     \makebox[0pt][ r ]
2379     {
2380         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2381         {
2382             \tl_use:N \l__enumext_mark_answer_sym_tl
2383         }
2384     }
2385     \skip_horizontal:n { \dim_use:N #2 }
2386 }
2387 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

12.27 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.

```

2388 \cs_new_protected:Nn \__enumext_store_internal_ref:
2389 {
2390     \cs_set_protected:Npn \__enumext_tmp:n ##1
2391     {
2392         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2393         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2394         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2395         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2396     }
2397     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2398     \cs_set:Npn \__enumext_tmp:n ##1
2399     { . \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.

```

2400     \bool_lazy_all:nT
2401     {
2402         { \bool_if_p:N \g__enumext_starred_bool }
2403         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2404     }
2405     {
2406         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2407         { \tl_use:N \l__enumext_label_copy_vii_tl }
2408     }
2409     \bool_lazy_all:nT
2410     {
2411         { \bool_not_p:n { \g__enumext_standar_bool } }
2412         { \bool_if_p:N \l__enumext_standar_bool }
2413         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2414     }
2415     {
2416         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2417         {
2418             \tl_use:N \l__enumext_label_copy_vii_tl
2419             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2420         }
2421     }

```

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.

```

2422     \bool_lazy_all:nT
2423     {
2424         { \bool_if_p:N \g__enumext_standar_bool }
2425         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2426         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2427     }
2428     {
2429         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2430         {
2431             \tl_use:N \l__enumext_label_copy_i_tl
2432             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2433         }
2434     }
2435     \cs_set:Npn \l__enumext_tmp:n ##1
2436     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2437     \bool_lazy_all:nT
2438     {
2439         { \bool_if_p:N \g__enumext_standar_bool }
2440         { \bool_if_p:N \l__enumext_starred_bool }
2441         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2442     }
2443     {
2444         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2445         {
2446             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2447             \tl_use:N \l__enumext_label_copy_vii_tl
2448         }
2449     }

```

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

```

2450     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2451     {
2452         \l__enumext_store_name_tl \c_colon_str
2453         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2454     }

```

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

```

2455     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2456     {
2457         \l__enumext_newlabel:nn
2458         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2459         { \l__enumext_newlabel_arg_two_tl }
2460     }
2461     \l__enumext_write_aux_file_tl
2462 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

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

`\l__enumext_store_anskey_code:n`

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

```

2463 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2464 {
2465   \int_gincr:N \g__enumext_item_anskey_int
2466   \__enumext_store_addto_prop:n {#1}
2467   \bool_if:NT \l__enumext_store_ref_key_bool
2468   {
2469     \__enumext_store_internal_ref:
2470   }
2471   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2472   \tl_clear:N \l__enumext_store_anskey_arg_tl
2473   \bool_lazy_and:nnT
2474   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2475   { \bool_not_p:n { \l__enumext_starred_bool } }
2476   {
2477     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2478   }
2479   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2480   \bool_lazy_and:nnT
2481   { \bool_not_p:n { \l__enumext_starred_bool } }
2482   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2483   {
2484     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2485     {
2486       ( \exp_not:V \l__enumext_store_item_join_int )
2487     }
2488   }

```

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

```

2489   \bool_if:NTF \l__enumext_store_item_star_bool
2490   {
2491     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2492     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2493     {
2494       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2495       {
2496         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2497       }
2498     }
2499     \dim_compare:nT
2500     {
2501       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2502     }
2503     {
2504       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2505       {
2506         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2507       }
2508     }
2509     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2510   }
2511   {
2512     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2513   }

```

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 $\backslash hyperlink$ with “symbol” set by `mark-ref` key and then store in *sequence*.

```

2514   \bool_lazy_and:nnT
2515   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2516   { \bool_if_p:N \l__enumext_hyperref_bool }
2517   {
2518     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2519     {
2520       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }

```

```

2521             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2522         }
2523     }
2524     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2525 }

```

(End of definition for __enumext_store_anskey_code:n.)

__enumext_anskey_show_wrap_arg:n

The function __enumext_anskey_show_wrap_arg:n “wraps” the $\{\langle argument \rangle\}$ passed to \anskey and the $\langle body \rangle$ for anskey* when using the wrap-ans key.

```

2526 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2527 {
2528     \par
2529     \bool_if:NTF \l__enumext_starred_bool
2530     {
2531         \__enumext_print_keyans_box:NN
2532         \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2533     }
2534     {
2535         \__enumext_print_keyans_box:cc
2536         { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2537         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2538     }
2539     \__enumext_anskey_wrapper:n { #1 }
2540 }

```

(End of definition for __enumext_anskey_show_wrap_arg:n.)

__enumext_anskey_show_wrap_left:n

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

```

2541 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2542 {
2543     \bool_if:NT \l__enumext_show_answer_bool
2544     {
2545         \__enumext_anskey_show_wrap_arg:n { #1 }
2546     }
2547     \bool_if:NT \l__enumext_show_position_bool
2548     {
2549         \tl_set:Nx \l__enumext_mark_answer_sym_tl
2550         {
2551             \group_begin:
2552             \exp_not:N \normalfont
2553             \exp_not:N \footnotesize [ \int_eval:n
2554             {
2555                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2556             }
2557             ]
2558             \group_end:
2559         }
2560         \__enumext_anskey_show_wrap_arg:n { #1 }
2561     }
2562 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

12.29 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[$\langle key = val \rangle$]{ $\langle content \rangle$ }.

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

```

2563 \keys_define:nn { enumext / anskey }
2564 {
2565     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2566     break-col .default:n = true,
2567     break-col .value_forbidden:n = true,
2568     item-join .int_set:N = \l__enumext_store_item_join_int,

```

```

2569     item-join .value_required:n = true,
2570     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2571     item-star .default:n = true,
2572     item-star .value_forbidden:n = true,
2573     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2574     item-sym* .value_required:n = true,
2575     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2576     item-pos* .value_required:n = true,
2577     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2578 }

```

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

```

2579 \cs_new_protected:Npn \l__enumext_anskey_unknown:n #1
2580 {
2581   \exp_args:NV \l__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2582 }
2583 \cs_new_protected:Npn \l__enumext_anskey_unknown:nn #1 #2
2584 {
2585   \tl_if_blank:nTF {#2}
2586   {
2587     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2588   }
2589   {
2590     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2591   }
2592 }

```

(End of definition for `\l__enumext_anskey_unknown:n` and `\l__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 `\l__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 `\l__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 `\l__enumext_store_anskey_code:n`.

```

2593 \NewDocumentCommand \anskey { o +m }
2594 {
2595   \l__enumext_anskey_safe_outer:
2596   \group_begin:
2597     \bool_if:NT \l__enumext_check_answers_bool
2598     {
2599       \tl_if_novalue:nF {#1}
2600       {
2601         \keys_set:nn { enumext / anskey } {#1}
2602       }
2603       \tl_if_blank:nTF {#2}
2604       {
2605         \msg_error:nn { enumext } { anskey-empty-arg }
2606       }
2607       {
2608         \l__enumext_anskey_safe_inner:
2609         \l__enumext_store_anskey_code:n {#2}
2610       }
2611     }
2612   \group_end:
2613 }

```

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

12.29.1 Internal functions for the command

`\l__enumext_anskey_safe_outer:` The `\l__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.

`\l__enumext_anskey_safe_inner:`

```

2614 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2615 {
2616   \bool_if:NF \l__enumext_store_active_bool
2617   {
2618     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2619   }

```



```

2620 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2621 {
2622   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2623 }
2624 \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2625 {
2626   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2627 }
2628 \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2629 {
2630   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2631 }
2632 }

```

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.

```

2633 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2634 {
2635   \int_incr:N \__enumext_anskey_level_int
2636   \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2637   {
2638     \msg_error:nn { enumext } { anskey-nested }
2639   }
2640   \bool_if:NF \__enumext_item_number_bool
2641   {
2642     \msg_error:nn { enumext } { anskey-unnumber-item }
2643   }
2644   \mode_if_math:T
2645   {
2646     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2647   }
2648 }

```

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

12.30 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:` (§12.31) which is executed after the environment in which the key `save-ans` is active.

```

2649 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2650 {
2651   \cs_undefine:c { anskey* }
2652   \cs_undefine:c { endanskey* }
2653   \cs_undefine:c { __scontents_anskey*_env_begin: }
2654   \cs_undefine:c { __scontents_anskey*_env_end: }
2655 }

```

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

```

2656 \__enumext_before_env:nn { enumext }
2657 {
2658   \bool_lazy_and:nnT
2659   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2660   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2661   {
2662     \cs_if_free:cf { __scontents_anskey*_env_begin: }
2663     {

```

```

2664         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2665     }
2666 }
2667 }
2668 \__enumext_before_env:nn { enumext* }
2669 {
2670     \bool_lazy_and:nnT
2671     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2672     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2673     {
2674         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2675         {
2676             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2677         }
2678     }
2679 }

```

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.

```

2680 \__enumext_before_env:nn { anskey* }
2681 {
2682     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2683     {
2684         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2685     }
2686     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2687     {
2688         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2689     }
2690     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2691     {
2692         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2693     }
2694     \bool_if:NF \__enumext_item_number_bool
2695     {
2696         \msg_error:nn { enumext } { anskey-unnumber-item }
2697     }
2698     \mode_if_math:T
2699     {
2700         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2701     }
2702 }

```

(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: (§12.25.1)` and we will execute it only if the variable `__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`.

```

2703 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2704 {
2705     \bool_if:NT \__enumext_anskey_env_bool
2706     {
2707         \newenvsc{anskey*}[store-env=#1,print-env=false]
2708         \__enumext_anskey_env_exec:
2709     }
2710 }
2711 \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`.

```

2712 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2713 {
2714     \keys_define:nn { scontents / scontents }
2715     {
2716         break-col .bool_gset:N = \__enumext_store_columns_break_bool,
2717         break-col .default:n = true,

```

```

2718     break-col .value_forbidden:n = true,
2719     item-join .int_gset:N = \g__enumext_store_item_join_int,
2720     item-join .value_required:n = true,
2721     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2722     item-star .default:n = true,
2723     item-star .value_forbidden:n = true,
2724     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2725     item-sym* .value_required:n = true,
2726     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2727     item-pos* .value_required:n = true,
2728     print-env .undefine:,
2729     store-env .undefine:,
2730     write-out .undefine:,
2731     unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2732   }
2733 }

```

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`.

```

2734 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2735 {
2736   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2737 }
2738 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2739 {
2740   \tl_if_blank:nTF {#2}
2741   {
2742     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2743   }
2744   {
2745     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2746   }
2747 }

```

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`.

```

2748 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2749 {
2750   \keys_define:nn { scontents / scontents }
2751   {
2752     break-col .undefine:,
2753     item-join .undefine:,
2754     item-star .undefine:,
2755     item-sym* .undefine:,
2756     item-pos* .undefine:,
2757     write-out .code:n = {
2758       \bool_set_false:N \l__scontents_storing_bool
2759       \bool_set_true:N \l__scontents_writing_bool
2760       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2761     },
2762     write-out .value_required:n = true,
2763     print-env .meta:nn = { scontents } { print-env = ##1 },
2764     print-env .default:n = true,
2765     store-env .meta:nn = { scontents } { store-env = ##1 },
2766     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2767   }
2768 }

```

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*.

```

2769 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2770 {
2771   \group_begin:
2772   \int_set:Nn \tex_newlinechar:D { `^^J }
2773   \__scontents_rescan_tokens:x
2774   {
2775     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2776     #1
2777   }
2778 }

```

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

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

```
2779 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2780 {
2781   \__enumext_before_env:nn { anskey* }
2782   {
2783     \__enumext_anskey_env_define_keys:
2784   }
```

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.

```
2785   \hook_if_empty:nF {env/anskey*/after}
2786   {
2787     \hook_gremove_code:nn {env/anskey*/after} { * }
2788   }
2789   \__enumext_after_env:nn { anskey* }
2790   {
2791     \__enumext_anskey_env_save_keys:
2792     \tl_clear:N \l__enumext_store_anskey_env_tl
2793     \tl_clear:N \l__enumext_store_anskey_opt_tl
2794     \bool_if:NT \l__enumext_check_answers_bool
2795     {
2796       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2797       {
2798         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2799       }
2800       \regex_match:nVTF
2801       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2802       \l__enumext_store_anskey_env_tl
2803       {
2804         \msg_error:nn { enumext } { anskey-empty-arg }
2805       }
2806       {
2807         \__enumext_anskey_env_store:
2808       }
2809     }
2810     \__enumext_anskey_env_clean_vars:
2811     \__enumext_anskey_env_reset_keys:
2812   }
2813 }
```

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:` 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`.

```
2814 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2815 {
2816   \bool_lazy_and:nnT
2817   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2818   { \bool_not_p:n { \l__enumext_starred_bool } }
2819   {
2820     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2821   }
```

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

```
2822 \bool_lazy_and:nnT
2823 { \bool_not_p:n { \l__enumext_starred_bool } }
2824 { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2825 {
2826   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2827   {
2828     ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2829   }
2830 }
```

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

```

2831 \bool_if:NT \g__enumext_store_item_star_bool
2832 {
2833   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2834   {
2835     ,item-star,
2836   }
2837   \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2838   {
2839     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2840     {
2841       ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2842     }
2843   }
2844   \dim_compare:nT
2845   {
2846     \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2847   }
2848   {
2849     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2850     {
2851       ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2852     }
2853   }
2854 }
2855 }

```

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`.

```

2856 \cs_new_protected:Nn \__enumext_anskey_env_store:
2857 {
2858   \group_begin:
2859   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2860   {
2861     \exp_args:Ne
2862     \__enumext_store_anskey_code:n
2863     {
2864       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2865     }
2866   }
2867   {
2868     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2869     \exp_args:Ne
2870     \__enumext_store_anskey_code:n
2871     {
2872       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2873     }
2874   }
2875   \group_end:
2876 }

```

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

```

2877 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2878 {
2879   \bool_gset_false:N \g__enumext_store_columns_break_bool
2880   \int_gzero:N \g__enumext_store_item_join_int
2881   \bool_gset_false:N \g__enumext_store_item_star_bool
2882   \tl_gclear:N \g__enumext_store_item_symbol_tl
2883   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2884 }

```

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

12.31 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*` (§12.30) 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.

```

2885 \cs_new_protected:Nn \__enumext_execute_after_env:
2886 {
2887   \int_compare:nNtT { \__enumext_level_int } = { 0 }
2888   {
2889     \tl_if_empty:NF \g__enumext_store_name_tl
2890     {
2891       \__enumext_stop_save_ans_msg:
2892       \__enumext_item_answer_diff:
2893       \__enumext_log_global_vars:
2894       \__enumext_log_answer_vars:
2895       \bool_if:NTF \g__enumext_check_ans_key_bool
2896       {
2897         \__enumext_check_ans_show:
2898       }
2899       { \__enumext_check_ans_log: }
2900       \__enumext_undefine_anskey_env:
2901     }
2902     \__enumext_reset_global_vars:
2903   }
2904 }

```

(End of definition for `__enumext_execute_after_env:`.)

- This function is passed to the function `__enumext_after_env:n` for the environments `enumext` (§12.38) and `enumext*` (§12.43) and it is executed only when the environments are not nested or at some level of these..

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

12.32.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the the current `⟨label⟩` for `\item*` in `keyans` environment and the current `⟨label⟩` for `\anspic*` in `keyanspic` environment followed by the `⟨contents⟩` of the *optional argument* of both commands to the `__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`.

```

2905 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2906 {
2907   \tl_clear:N \__enumext_store_current_label_tl
2908   \int_compare:nNtTF { \__enumext_keyans_pic_level_int } = { 1 }
2909   {
2910     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
2911   }
2912   {
2913     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
2914   }

```

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

```

2915   \tl_if_novalue:nF { #1 }
2916   {
2917     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
2918     {
2919       \tl_put_right:Ne \__enumext_store_current_label_tl
2920       {
2921         \__enumext_store_keyans_item_opt_sep_tl
2922       }
2923     }
2924     \tl_put_right:Ne \__enumext_store_current_label_tl { #1 }
2925   }
2926   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
2927 }

```

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

12.32.2 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 `⟨label⟩` for `\item*` and `\anspic*` with the `⟨contents⟩` 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:` The function `__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the `save-ref` key for `\item*` and `\anspic*` commands. First we will create copies of the current `(labels)` and remove the dots “.” from them, we do not want to get double dots in references.

```

2928 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2929 {
2930   \bool_if:NT \l__enumext_store_ref_key_bool
2931   {
2932     \cs_set_protected:Npn \__enumext_tmp:n ##1
2933     {
2934       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2935       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2936       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2937       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2938     }
2939     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2940     \__enumext_keyans_store_ref_aux_i:
2941   }
2942 }

```

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

```

2943 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2944 {
2945   \bool_if:NT \g__enumext_starred_bool
2946   {
2947     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2948   }
2949   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2950   {
2951     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2952     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2953   }
2954   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2955   {
2956     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2957     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2958   }
2959   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2960   {
2961     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2962     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2963   }
2964   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2965   {
2966     \l__enumext_store_name_tl \c_colon_str
2967     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2968   }
2969   \__enumext_keyans_store_ref_aux_ii:
2970 }

```

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.

```

2971 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2972 {
2973   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2974   {
2975     \__enumext_newlabel:nn
2976     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2977     { \l__enumext_newlabel_arg_two_tl }
2978   }
2979   \l__enumext_write_aux_file_tl
2980 }

```

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

12.32.3 Storing content in sequence

`__enumext_keyans_addto_seq:n` 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.

```

2981 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2982 {
2983   \tl_clear:N \l__enumext_store_current_label_tl
2984   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2985   {
2986     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2987   }
2988   {
2989     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2990   }
2991   \tl_if_novalue:nF { #1 }
2992   {
2993     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2994     {
2995       \tl_put_right:Ne \l__enumext_store_current_label_tl
2996       {
2997         \l__enumext_store_keyans_item_opt_sep_tl
2998       }
2999     }
3000     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3001   }
3002   \__enumext_keyans_addto_seq_link:
3003 }

```

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.

```

3004 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3005 {
3006   \bool_lazy_and:nnT
3007   { \bool_if_p:N \l__enumext_store_ref_key_bool }
3008   { \bool_if_p:N \l__enumext_hyperref_bool }
3009   {
3010     \tl_put_right:Ne \l__enumext_store_current_label_tl
3011     {
3012       \hfill \exp_not:N \hyperlink
3013       {
3014         \exp_not:V \l__enumext_newlabel_arg_one_tl
3015       }
3016       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3017     }
3018   }
3019   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3020   \bool_if:NT \l__enumext_check_answers_bool
3021   {
3022     \int_gincr:N \g__enumext_item_anskey_int
3023   }
3024 }

```

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

12.32.4 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.

```

3025 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3026 {
3027   \tl_if_novalue:nF { #1 }
3028   {
3029     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3030   }
3031   \bool_if:NT \l__enumext_show_answer_bool
3032   {
3033     \__enumext_keyans_show_ans:

```

```

3034     }
3035     \bool_if:NT \l__enumext_show_position_bool
3036     {
3037         \__enumext_keyans_show_pos:
3038     }
3039 }
3040 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3041 {
3042     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3043     {
3044         \bool_lazy_or:nnT
3045         { \bool_if_p:N \l__enumext_show_answer_bool }
3046         { \bool_if_p:N \l__enumext_show_position_bool }
3047         {
3048             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3049         }
3050     }
3051 }
3052 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3053 {
3054     \bool_if:NT \l__enumext_starred_bool
3055     {
3056         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3057         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3058     }
3059     \tl_put_left:Nn \l__enumext_label_v_tl
3060     {
3061         \__enumext_print_keyans_box:NN
3062         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3063     }
3064 }
3065 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3066 {
3067     \bool_if:NT \l__enumext_starred_bool
3068     {
3069         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3070         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3071     }
3072     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3073     {
3074         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3075         {
3076             \group_begin:
3077             \exp_not:N \normalfont
3078             \exp_not:N \footnotesize [ \int_eval:n
3079             {
3080                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3081             }
3082             ]
3083             \group_end:
3084         }
3085     }
3086     {
3087         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3088         {
3089             \group_begin:
3090             \exp_not:N \normalfont
3091             \exp_not:N \footnotesize [ \int_eval:n
3092             {
3093                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3094             }
3095             ]
3096             \group_end:
3097         }
3098     }
3099     \tl_put_left:Nn \l__enumext_label_v_tl
3100     {
3101         \__enumext_print_keyans_box:NN
3102         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3103     }
3104 }

```

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

12.33 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`.

```

3105 \cs_new_protected:Npn \__enumext_default_item:n #1
3106 {
3107   \tl_if_novalue:nTF {#1}
3108   {
3109     \bool_if:NT \l__enumext_check_answers_bool
3110     {
3111       \int_gincr:N \g__enumext_item_number_int
3112       \bool_set_true:N \l__enumext_item_number_bool
3113     }
3114     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3115     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3116   }
3117   {
3118     \bool_set_eq:cc
3119     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3120     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3121     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3122   }
3123 }

```

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

`__enumext_starred_item:nn`

`__enumext_item_star_exec:`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as “first” *optional argument* in the global variable `\g__enumext_item_symbol_aux_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item-pos*` or by the “second” *optional argument*, then we will see the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`.

```

3124 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3125 {
3126   \tl_if_novalue:nTF {#1}
3127   {
3128     \tl_gset_eq:Nc
3129     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3130   }
3131   {
3132     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3133   }
3134   \tl_if_novalue:nTF {#2}
3135   {
3136     \dim_set_eq:cc
3137     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3138     { \l__enumext_labelsep_ \__enumext_level: _dim }
3139   }
3140   {
3141     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3142   }
3143   \bool_if:NT \l__enumext_check_answers_bool
3144   {

```

```

3145         \int_gincr:N \g__enumext_item_number_int
3146         \bool_set_true:N \l__enumext_item_number_bool
3147     }
3148     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3149     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3150 }

```

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

```

3151 \cs_new_protected:Nn \__enumext_item_star_exec:
3152 {
3153     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3154     {
3155         \mode_leave_vertical:
3156         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3157         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3158         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3159     }
3160 }

```

(End of definition for `__enumext_starred_item:nn` and `__enumext_item_star_exec:`.)

`__enumext_redefine_item:`

The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§12.38).

```

3161 \cs_new_protected:Nn \__enumext_redefine_item:
3162 {
3163     \RenewDocumentCommand \item { s o o }
3164     {
3165         \bool_if:nTF {##1}
3166         {
3167             \__enumext_starred_item:nn {##2} {##3}
3168         }
3169         { \__enumext_default_item:n {##2} }
3170     }
3171 }

```

(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:`
`__enumext_make_label_std:`
`__enumext_make_label_box:`

The function `__enumext_make_label:` redefine `\makelabel` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§12.38).

```

3172 \cs_new_protected:Nn \__enumext_make_label:
3173 {
3174     \IfDocumentMetadataTF
3175     {
3176         \__enumext_make_label_box:
3177     }
3178     { \__enumext_make_label_std: }
3179 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3180 \cs_new_protected:Nn \__enumext_make_label_std:
3181 {
3182     \RenewDocumentCommand \makelabel { m }
3183     {
3184         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3185         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3186         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3187         {
3188             \__enumext_item_star_exec:
3189             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3190         }
3191         { ##1 }
3192         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3193         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3194     }
3195 }

```

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

```

3196 \cs_new_protected:Nn \__enumext_make_label_box:
3197 {
3198   \RenewDocumentCommand \make_label { m }
3199   {
3200     \makebox
3201     [ \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim } ]
3202     [ \str_use:c { \__enumext_align_label_pos_ \__enumext_level: _str } ]
3203     {
3204       \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3205       \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3206       {
3207         \__enumext_item_star_exec:
3208         \use:c { \__enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3209       }
3210       { ##1 }
3211       \tl_gclear:N \g__enumext_item_symbol_aux_tl
3212     }
3213   }
3214 }

```

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

12.34 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*`.

```

3215 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3216 {
3217   \keys_define:nn { enumext / #1 }
3218   {
3219     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
3220     item-sym* .value_required:n = true,
3221     item-sym* .initial:n = { $\star$ },
3222     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
3223     item-pos* .value_required:n = true,
3224   }
3225 }
3226 \clist_map_inline:nn
3227 {
3228   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3229 }
3230 { \__enumext_tmp:nn #1 }

```

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

12.35 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).

12.35.1 Handling unknown keys for `keyans` and `keyans*`

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

```

3231 \cs_set_protected:Npn \__enumext_tmp:n #1
3232 {
3233   \keys_define:nn { enumext / #1 }
3234   {
3235     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3236   }
3237 }
3238 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3239 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3240 {
3241   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3242 }
3243 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3244 {
3245   \tl_if_blank:nTF {#2}

```

```

3246     {
3247         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3248     }
3249     {
3250         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3251     }
3252 }

```

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

12.35.2 Handling unknown keys for `enumext*`

`unknown`

Define and set `unknown` key for `enumext*` environment.

`__enumext_starred_unknown_keys:n`
`__enumext_starred_unknown_keys:nn`

```

3253 \keys_define:nn { enumext / enumext* }
3254 {
3255     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3256 }

```

Internal functions for handling `unknown` key.

```

3257 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3258 {
3259     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3260 }
3261 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3262 {
3263     \tl_if_blank:nTF {#2}
3264     {
3265         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3266     }
3267     {
3268         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3269     }
3270 }

```

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

12.35.3 Handling unknown keys for `enumext`

`unknown`

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

`__enumext_standar_unknown_keys:n`
`__enumext_standar_unknown_keys:nn`

```

3271 \cs_set_protected:Npn \__enumext_tmp:n #1
3272 {
3273     \keys_define:nn { enumext / #1 }
3274     {
3275         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3276     }
3277 }
3278 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3279 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3280 {
3281     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3282 }
3283 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3284 {
3285     \tl_if_blank:nTF {#2}
3286     {
3287         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3288     }
3289     {
3290         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3291     }
3292 }

```

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

12.36 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.

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

```

3293 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3294 {
3295   \tl_if_novalue:nTF { #1 }
3296   {
3297     \bool_set_true:N \l__enumext_wrap_label_v_bool
3298     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3299   }
3300   {
3301     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3302     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3303   }
3304 }

```

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

`__enumext_keyans_starred_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.

```

3305 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3306 {
3307   \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3308   \__enumext_keyans_show_left:n { #1 }
3309   \bool_set_true:N \l__enumext_wrap_label_v_bool
3310   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3311   \__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.

```

3312   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3313   \__enumext_keyans_addto_prop:n { #1 }
3314   \__enumext_keyans_store_ref:
3315   \__enumext_keyans_addto_seq:n { #1 }
3316   \int_gincr:N \g__enumext_check_starred_cmd_int
3317 }

```

(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 (§12.37.2).

`__enumext_keyans_redefine_item:`

```

3318 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3319 {
3320   \RenewDocumentCommand \item { s o }
3321   {
3322     \bool_if:nTF {##1}
3323     {
3324       \peek_remove_spaces:n
3325       {
3326         \__enumext_keyans_starred_item:n {##2}
3327       }
3328     }
3329     {
3330       \__enumext_keyans_default_item:n {##2}
3331     }
3332   }
3333 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 15.)

`__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 (§12.37.2).

`__enumext_keyans_make_label_std:`
`__enumext_keyans_make_label_box:`

```

3334 \cs_new_protected:Nn \__enumext_keyans_make_label:

```



```

3335 {
3336   \IfDocumentMetadataTF
3337   {
3338     \__enumext_keyans_make_label_box:
3339   }
3340   { \__enumext_keyans_make_label_std: }
3341 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3342 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3343 {
3344   \RenewDocumentCommand \makeLabel { m }
3345   {
3346     \tl_use:N \l__enumext_label_fill_left_v_tl
3347     \tl_use:N \l__enumext_label_font_style_v_tl
3348     \bool_if:NTF \l__enumext_wrap_label_v_bool
3349     {
3350       \__enumext_wrapper_label_v:n { ##1 }
3351     }
3352     { ##1 }
3353     \tl_use:N \l__enumext_label_fill_right_v_tl
3354   }
3355 }

```

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

```

3356 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3357 {
3358   \RenewDocumentCommand \makeLabel { m }
3359   {
3360     \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3361     {
3362       \tl_use:N \l__enumext_label_font_style_v_tl
3363       \bool_if:NTF \l__enumext_wrap_label_v_bool
3364       {
3365         \__enumext_wrapper_label_v:n { ##1 }
3366       }
3367       { ##1 }
3368     }
3369   }
3370 }

```

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

12.37 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`.

12.37.1 Calculation of `\leftmargin` and `\itemindent`

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

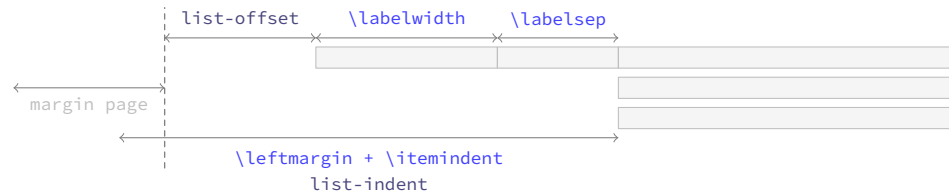


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

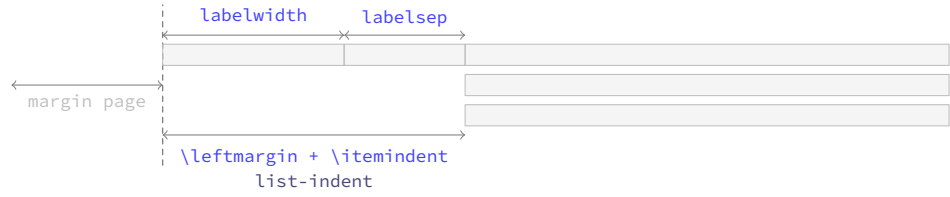
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

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

```

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

```

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

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§12.37.2).

```

3371 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3372 {
3373   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3374   {
3375     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3376     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3377   }
3378   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3379   {
3380     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3381     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3382   }

```

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

```

3383   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3384   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3385   {
3386     \dim_set:Nn #6 { #1 + #2 - #4 }
3387     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3388   }
3389   {
3390     \dim_compare:nNnT { #4 } = { #1 + #2 }
3391     { \dim_set:Nn #6 { \c_zero_dim } }
3392     \dim_compare:nNnT { #4 } < { #1 + #2 }
3393     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3394     \dim_compare:nNnT { #4 } > { #1 + #2 }
3395     {
3396       \dim_set:Nn #6 { -#1 - #2 + #4 }
3397       \dim_set:Nn #6 { #6*-1 }
3398     }
3399     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3400   }
3401 }
3402 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

12.37.2 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.

```

3403 \cs_set_protected:Npn \__enumext_tmp:n #1
3404 {
3405   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3406   {
3407     \__enumext_calc_hspace:ccccc
3408     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3409     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3410     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3411     { \__enumext_leftmargin_tmp_#1_bool }
3412     \clist_map_inline:nn
3413       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3414       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3415     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3416       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3417     \usecounter { enumX#1 }
3418     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3419     \str_if_eq:nnTF {#1} { v }
3420     {
3421       \__enumext_keyans_redefine_item:
3422       \__enumext_keyans_make_label:
3423       \__enumext_keyans_ref:
3424       \__enumext_keyans_fake_item_indent:
3425       \bool_if:cT { \__enumext_show_length_#1_bool }
3426       {
3427         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3428       }
3429     }
3430     {
3431       \__enumext_redefine_item:
3432       \__enumext_make_label:
3433       \__enumext_standar_ref:
3434       \__enumext_fake_item_indent:
3435       \bool_if:cT { \__enumext_show_length_#1_bool }
3436       {
3437         \msg_term:nnne { enumext } { list-lengths } {#1}
3438         { \int_use:N \__enumext_level_int }
3439       }
3440     }
3441   }
3442 }
3443 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3444 \cs_set_protected:Npn \__enumext_tmp:n #1
3445 {
3446   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3447   {
3448     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3449     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3450     \__enumext_calc_hspace:ccccc
3451     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3452     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3453     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3454     { \__enumext_leftmargin_tmp_#1_bool }
3455     \clist_map_inline:nn
3456       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3457       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3458     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3459       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3460     \skip_zero:Nc \parsep { \__enumext_itemsep_#1_skip }
3461     \skip_zero:N \partopsep
3462     \usecounter { enumX#1 }

```

```

3463     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3464     \__enumext_starred_ref:
3465     \str_if_eq:nnTF {#1} { vii }
3466     {
3467         \__enumext_fake_item_indent_vii:
3468         \bool_if:cT { \__enumext_show_length_vii_bool }
3469         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3470     }
3471     {
3472         \__enumext_fake_item_indent_viii:
3473         \bool_if:cT { \__enumext_show_length_#1_bool }
3474         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3475     }
3476 }
3477 }
3478 \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:.)

12.38 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”.

```

3479 \cs_new_protected:Nn \__enumext_safe_exec:
3480 {
3481     \__enumext_internal_mini_page:
3482     \__enumext_is_not_nested:
3483     \int_incr:N \l__enumext_level_int
3484     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3485     { \msg_fatal:nn { enumext } { list-too-deep } }
3486     \bool_set_true:N \l__enumext_standar_bool
3487     \bool_set_false:N \l__enumext_starred_bool
3488     \__enumext_is_on_first_level:
3489 }

```

(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.

```

3490 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3491 {
3492     \tl_if_novalue:nF {#1}
3493     {
3494         \str_clear:N \l__enumext_series_str
3495         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3496         {
3497             \keys_set:nn { enumext / level-1 } {#1}
3498             \__enumext_parse_series:n {#1}
3499             \__enumext_nested_base_line_fix:
3500         }
3501         {
3502             \exp_args:Ne \keys_set:nn
3503             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3504         }
3505         \__enumext_store_active_keys:n {#1}
3506     }
3507 }

```

(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*`.

```

3508 \cs_new_protected:Nn \__enumext_start_store_level:

```

```

3509 {
3510   \bool_lazy_all:nT
3511   {
3512     { \bool_if_p:N \l__enumext_store_active_bool }
3513     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3514     { \bool_if_p:N \g__enumext_standar_bool }
3515   }
3516   {
3517     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3518     {
3519       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3520       \__enumext_store_level_open:
3521     }
3522   }

```

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

```

3523   \bool_lazy_all:nT
3524   {
3525     { \bool_if_p:N \l__enumext_store_active_bool }
3526     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3527     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3528   }
3529   {
3530     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3531     {
3532       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3533       \__enumext_store_level_open:
3534     }
3535   }
3536 }

```

(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*`.

```

3537 \cs_new_protected:Nn \__enumext_stop_store_level:
3538 {
3539   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3540   {
3541     \__enumext_store_level_close:
3542   }
3543 }

```

(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.

```

3544 \cs_new_protected:Nn \__enumext_multicols_start:
3545 {
3546   \int_compare:nNnT
3547   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3548   {
3549     \dim_compare:nNnT
3550     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3551     {
3552       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3553       {
3554         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3555           + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3556         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3557         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3558       }
3559     }
3560     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3561     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3562     {
3563       \dim_zero:N \columnseprule
3564     }

```

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.

```

3565     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3566     {
3567         \skip_zero:N \multicolsep
3568         \__enumext_multi_addvspace:
3569     }
3570     \raggedcolumns
3571     \begin{multicols}{\int_use:c { \__enumext_columns_ \__enumext_level: _int }}
3572 }
3573 }

```

(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:`.

```

3574 \cs_new_protected:Nn \__enumext_multicols_stop:
3575 {
3576     \int_compare:nNnTF
3577     { \int_use:c { \__enumext_columns_ \__enumext_level: _int }} > { 1 }
3578     {
3579         \__enumext_stop_list:
3580         \__enumext_stop_store_level:
3581         \end{multicols}
3582         \__enumext_unskip_unkern:
3583         \__enumext_unskip_unkern:
3584         \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip }}
3585     }
3586     {
3587         \__enumext_stop_list:
3588         \__enumext_stop_store_level:
3589     }
3590 }

```

(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.

```

3591 \cs_new_protected:Nn \__enumext_before_list:
3592 {
3593     \__enumext_vspace_above:
3594     \__enumext_before_args_exec:
3595     \__enumext_check_ans_active:

```

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

```

3596     \dim_compare:nNnT
3597     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }} > { \c_zero_dim }
3598     {
3599         \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
3600         {
3601             \linewidth
3602             - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
3603             - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
3604         }

```

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

```

3605     \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
3606     \int_gincr:N \g__enumext_minipage_stat_int
3607     \__enumext_minipage_add_space:

```

```

3608         \noindent
3609         \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
3610     }
3611     \__enumext_multicols_start:
3612 }

```

(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.

```

3613 \cs_new_protected:Nn \__enumext_second_part:
3614 {
3615     \bool_if:cTF { \__enumext_minipage_active_ \__enumext_level: _bool }
3616     {
3617         \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3618         {
3619             \msg_warning:nn { enumext } { missing-miniright }
3620             \miniright
3621         }
3622         \int_gzero:N \g__enumext_minipage_stat_int
3623         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3624         \end__enumext_mini_page
3625     }
3626     {
3627         \__enumext_multicols_stop:
3628     }

```

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.

```

3629     \__enumext_after_stop_list:
3630     \__enumext_check_ans_key_hook:
3631     \__enumext_vspace_below:
3632     \bool_set_false:N \l__enumext_standar_bool
3633     \__enumext_resume_save_counter:
3634 }

```

(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.

```

3635 \cs_new_protected:Nn \__enumext_set_item_width:
3636 {
3637     \dim_set:Nn \itemwidth { \linewidth }
3638     \dim_compare:nT
3639     {
3640         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3641     }
3642     {
3643         \dim_sub:Nn \itemwidth
3644         {
3645             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3646         }
3647     }
3648 }

```

(End of definition for __enumext_set_item_width:.)

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

```

3649 \NewDocumentEnvironment{enumext}{0}{ }
3650 {
3651     \__enumext_safe_exec:
3652     \__enumext_parse_keys:n {#1}
3653     \__enumext_before_list:
3654     \__enumext_start_store_level:
3655     \__enumext_start_list:nn
3656     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }

```



```

3657     {
3658         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3659         \__enumext_before_keys_exec:
3660     }
3661     \__enumext_set_item_width:
3662     \__enumext_after_args_exec:
3663 }
3664 {
3665     \__enumext_second_part:
3666 }

```

(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`.

```

3667 \__enumext_after_env:nn {enumext}
3668 {
3669     \__enumext_execute_after_env:
3670 }

```

12.39 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”.

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.

```

3671 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3672 {
3673     \bool_if:NF \l__enumext_store_active_bool
3674     {
3675         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3676     }
3677     \int_incr:N \l__enumext_keyans_level_int
3678     \bool_set_true:N \l__enumext_keyans_env_bool
3679     \__enumext_keyans_name_and_start:
3680     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3681     \bool_set_false:N \l__enumext_store_active_bool
3682     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3683     {
3684         \msg_error:nn { enumext } { keyans-nested }
3685     }
3686     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3687     {
3688         \msg_error:nn { enumext } { keyans-wrong-level }
3689     }
3690 }

```

(End of definition for __enumext_keyans_safe_exec:.)

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

```

3691 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3692 {
3693     \keys_set:nn { enumext / keyans } { #1 }
3694 }

```

(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:
3695 \cs_new_protected:Nn \__enumext_before_list_v:
3696 {
3697     \__enumext_vspace_above_v:
3698     \__enumext_before_args_exec:
3699     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3700     {
3701         \dim_set:Nn \l__enumext_minipage_left_v_dim
3702         {
3703             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3704         }
3705         \bool_set_true:N \l__enumext_minipage_active_v_bool

```

```

3706         \int_gincr:N \g__enumext_minipage_stat_int
3707         \__enumext_keyans_minipage_add_space:
3708         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3709     }
3710     \__enumext_keyans_multicols_start:
3711 }
3712 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3713 {
3714     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3715     {
3716         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3717         {
3718             \dim_set:Nn \l__enumext_columns_sep_v_dim
3719             {
3720                 (
3721                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3722                 ) / \l__enumext_columns_v_int
3723                 - \l__enumext_listoffset_v_dim
3724             }
3725         }
3726         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3727         \dim_zero:N \columnseprule % no rule here
3728         \bool_if:NF \l__enumext_minipage_active_v_bool
3729         {
3730             \skip_zero:N \multicolsep
3731             \__enumext_keyans_multi_addvspace:
3732         }
3733         \raggedcolumns
3734         \begin{multicols}{ \l__enumext_columns_v_int }
3735     }
3736 }
3737 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3738 {
3739     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3740     {
3741         \__enumext_stop_list:
3742         \end{multicols}
3743         \__enumext_unskip_unkern:
3744         \__enumext_unskip_unkern:
3745         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3746     }
3747     {
3748         \__enumext_stop_list:
3749     }
3750 }
3751 \cs_new_protected:Nn \__enumext_second_part_v:
3752 {
3753     \bool_if:NTF \l__enumext_minipage_active_v_bool
3754     {
3755         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3756         {
3757             \msg_warning:nn { enumext } { missing-miniright }
3758             \miniright
3759         }
3760         \int_gzero:N \g__enumext_minipage_stat_int
3761         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3762         \end__enumext_mini_page
3763         \par\addvspace{ \l__enumext_minipage_after_skip }
3764     }
3765     {
3766         \__enumext_keyans_multicols_stop:
3767     }
3768     \bool_set_false:N \l__enumext_keyans_env_bool
3769     \__enumext_after_stop_list_v:
3770     \__enumext_vspace_below_v:
3771 }

```

(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.

```

3772 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3773 {
3774   \dim_set:Nn \itemwidth { \linewidth }
3775   \dim_compare:nT
3776   {
3777     \l__enumext_listoffset_v_dim != \c_zero_dim
3778   }
3779   {
3780     \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3781   }
3782 }

```

(End of definition for __enumext_keyans_set_item_width:.)

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

```

3783 \NewDocumentEnvironment{keyans}{0}{}
3784 {
3785   \__enumext_keyans_safe_exec:
3786   \__enumext_keyans_parse_keys:n {#1}
3787   \__enumext_before_list_v:
3788   \__enumext_start_list:nn
3789   { \tl_use:N \l__enumext_label_v_tl }
3790   {
3791     \__enumext_list_arg_two_v:
3792     \__enumext_before_keys_exec_v:
3793   }
3794   \__enumext_keyans_set_item_width:
3795   \__enumext_after_args_exec_v:
3796 }
3797 {
3798   \__enumext_check_starred_cmd:n { item }
3799   \__enumext_second_part_v:
3800 }

```

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

12.40 Tagging PDF support for non-standart list environments

The ~~TeX~~ release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the **list** environment. Unfortunately non-standard *list environments* like **keyanspic** or the horizontal list environments **enumext*** and **keyans*** are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a **list** environment is a **list** environment, no matter what it looks like in the PDF file.

To maintain a correct **list** structure when \DocumentMetadata is active, it is necessary to do some things manually. 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](#).

12.40.1 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
3801 \socket_new:nn {taggsupport/enumext/starred}{1}
3802 \socket_new_plugin:nnn {taggsupport/enumext/starred} {start-list-tags}
3803 {
3804   \tag_resume:n {#1}
3805   \tag_struct_begin:n {tag=LI}
3806   \tag_struct_begin:n {tag=Lbl}
3807   \tag_mc_begin:n {tag=Lbl}
3808 }
3809 \socket_new_plugin:nnn {taggsupport/enumext/starred} {stop-start-tags}
3810 {
3811   \tag_mc_end:
3812   \tag_struct_end:n {tag=Lbl}
3813   \tag_struct_begin:n {tag=LBody}
3814   \tag_struct_begin:n {tag=text-unit}
3815   \tag_struct_begin:n {tag=text}
3816 }
3817 \socket_new_plugin:nnn {taggsupport/enumext/starred} {stop-list-tags}
3818 {
3819   \tag_struct_end:n {tag=text}
3820   \tag_struct_end:n {tag=text-unit}
3821   \tag_struct_end:n {tag=LBody}
3822   \tag_struct_end:n {tag=LI}

```

```

3823     \tag_suspend:n {#1}
3824 }

```

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

```

3825 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3826 {
3827     \IfDocumentMetadataTF
3828     {
3829         \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3830         \socket_use:n {tagsupport/enumext/starred} {#1}
3831     } {}
3832 }
3833 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3834 {
3835     \IfDocumentMetadataTF
3836     {
3837         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3838         \socket_use:nn {tagsupport/enumext/starred} { }
3839     } {}
3840 }
3841 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3842 {
3843     \IfDocumentMetadataTF
3844     {
3845         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3846         \socket_use:nn {tagsupport/enumext/starred} {#1}
3847     } {}
3848 }

```

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

12.40.2 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:
3849 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3850 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3851 {
3852     \tag_resume:n {keyanspic}
3853     \tag_struct_begin:n {tag=LI}
3854     \tag_struct_begin:n {tag=Lbl}
3855     \tag_mc_begin:n {tag=Lbl}
3856 }
3857 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3858 {
3859     \tag_mc_end:
3860     \tag_struct_end:n {tag=Lbl}
3861     \tag_struct_begin:n {tag=LBody}
3862     \tag_struct_begin:n {tag=text-unit}
3863     \tag_struct_begin:n {tag=text}
3864     \tag_mc_begin:n {tag=text}
3865 }
3866 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3867 {
3868     \tag_mc_end:
3869     \tag_struct_end:n {tag=text-unit}
3870     \tag_struct_end:n {tag=text}
3871     \tag_struct_end:n {tag=LBody}
3872     \tag_struct_end:n {tag=LI}
3873     \tag_suspend:n {keyanspic}
3874 }

```

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

```

3875 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3876 {
3877     \IfDocumentMetadataTF
3878     {
3879         \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3880         \socket_use:n {tagsupport/enumext/keyanspic}
3881     } {}
3882 }
3883 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3884 {
3885     \IfDocumentMetadataTF

```

```

3886     {
3887         \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3888         \socket_use:nn {tagsupport/enumext/keyanspic}
3889     } {}
3890 }
3891 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3892 {
3893     \IfDocumentMetadataTF
3894     {
3895         \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3896         \socket_use:nn {tagsupport/enumext/keyanspic}
3897     } {}
3898 }

```

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

12.41 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.

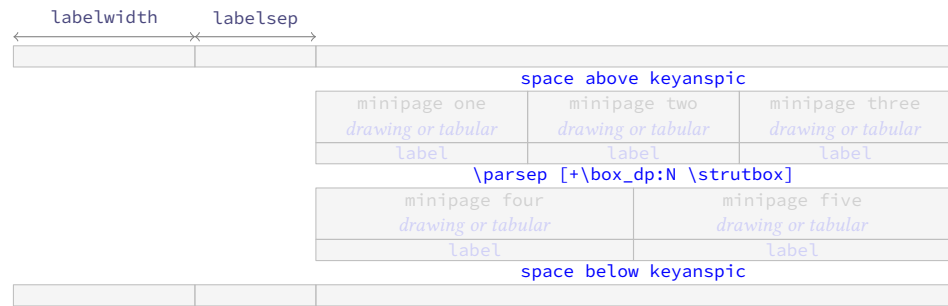


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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 `<label>`, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to `<label>` and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is `<label>`, is above #3 there are no problems with *tagged PDF*, but if #3 comes first the list created with *tagged PDF* will not be correct.

12.41.1 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* ‘`*`’.

```

3899 \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3900 {
3901     \int_incr:N \l__enumext_keyans_pic_level_int
3902     \int_compare:nNt { \l__enumext_keyans_pic_level_int } > { 1 }
3903     {
3904         \msg_error:nn { enumext } { keyanspic-nested }
3905     }
3906     \__enumext_keyans_name_and_start:
3907     \bool_if:nTF { #1 }
3908     {
3909         \bool_set_true:N \l__enumext_keyans_pic_star_bool
3910         \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
3911     }
3912     {
3913         \str_set:Nn \l__enumext_anspic_mini_pos_str { b }

```

```

3914     }
3915 }

```

(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`.

```

3916 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3917 {
3918   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3919   {
3920     \skip_set:Nn #1 { -#1 }
3921   }
3922 }

```

(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 *(keys)* `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.

```

3923 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3924 {
3925   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3926   \__enumext_list_arg_two_v:
3927   \__enumext_keyans_pic_skip_abs:N \parsep

```

Now we increment the `enumXv` counter of the `keyans` environment and save the *total height* of the *(label)* in `\l__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the *starred argument* “***” is NOT present.

```

3928   \bool_if:NF \l__enumext_keyans_pic_star_bool
3929   {
3930     \stepcounter { enumXv }
3931     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3932     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3933     {
3934       \box_ht_plus_dp:N \l__enumext_anspic_label_box
3935     }
3936     \skip_add:Nn \parsep
3937     {
3938       \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3939     }
3940     \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3941   }

```

Finally we adjust the value of `\leftmargin` and `\topsep` then set `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` to zero so that the *horizontal* and *vertical* space is not affected.

```

3942   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3943   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3944   \dim_zero:N \labelwidth
3945   \dim_zero:N \listparindent
3946   \dim_zero:N \labelsep
3947   \skip_zero:N \partopsep
3948   \skip_zero:N \itemsep
3949 }

```

(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`.

```

3950 \NewDocumentEnvironment{keyanspic}{s o }
3951 {
3952   \__enumext_keyans_pic_safe_exec:n { #1 }
3953   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3954   \IfDocumentMetadataTF
3955   {
3956     \tag_suspend:n {list}
3957   }{}
3958   \item[] \scan_stop:
3959   % paranoia

```

```

3960 \RenewDocumentCommand \item {}
3961 {
3962   \msg_error:nn { enumext } { keyanspic-item-cmd }
3963 }
3964 \IfDocumentMetadataTF
3965 {
3966   \tag_resume:n {keyanspic}
3967   \tag_tool:n {para/tagging=false}
3968   \tag_suspend:n {keyanspic}
3969 } { }
3970 }
3971 {
3972   \IfDocumentMetadataTF
3973   {
3974     \tag_resume:n {keyanspic}
3975     \tag_struct_begin:n {tag=L,attribute=enumerate}
3976   } { }

```

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.

```

3977 \tl_if_novalue:nTF { #2 }
3978 {
3979   \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
3980 }
3981 { \__enumext_anspic_print:n { #2 } }
3982 \IfDocumentMetadataTF
3983 {
3984   \tag_suspend:n {keyanspic}
3985 } { }
3986 \end{list}
3987 \IfDocumentMetadataTF
3988 {
3989   \tag_struct_end:
3990   \tag_struct_end:
3991 } { }

```

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

```

3992 \__enumext_check_starred_cmd:n { anspic }
3993 \setcounter { enumXvi } { 0 }
3994 \bool_if:NTF \l__enumext_keyans_pic_star_bool
3995 {
3996   \par\addvspace{ 0.5\box_dp:N \strutbox }
3997 }
3998 {
3999   \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
4000 }
4001 %\bool_set_false:N \l__enumext_store_active_bool
4002 }

```

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

12.41.2 The command `\anspic`

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

`\anspic` We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. The three arguments are handled by the function `__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

4003 \NewDocumentCommand \anspic { s o +m }
4004 {
4005   \bool_if:NF \l__enumext_store_active_bool
4006   {
4007     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4008   }
4009   \int_compare:nNt { \l__enumext_level_int } > { 1 }
4010   {
4011     \msg_error:nn { enumext } { keyanspic-wrong-level }
4012   }

```



```

4013     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4014     {
4015         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4016     }
4017     \seq_put_right:Nn \l__enumext_anspic_args_seq
4018     {
4019         \l__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4020     }
4021 }

```

(End of definition for \anspic. This function is documented on page 16.)

__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* ‘*’.

```

4022 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4023 {
4024     \bool_if:NF \l__enumext_keyans_pic_star_bool
4025     {
4026         \IfDocumentMetadataTF
4027         {
4028             \tag_suspend:n {keyanspic}
4029         } { }
4030         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4031         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4032         {
4033             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4034         }
4035         \IfDocumentMetadataTF
4036         {
4037             \tag_resume:n {keyanspic}
4038         } { }
4039     }
4040 }

```

(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.

```

4041 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4042 {
4043     \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4044     {
4045         \bool_if:nT { #1 }
4046         {
4047             \__enumext_keyans_addto_prop:n { #2 }
4048             \__enumext_keyans_store_ref:
4049             \__enumext_keyans_addto_seq:n { #2 }
4050             \int_gincr:N \g__enumext_check_starred_cmd_int
4051             \bool_lazy_or:nnT
4052             { \bool_if_p:N \l__enumext_show_answer_bool }
4053             { \bool_if_p:N \l__enumext_show_position_bool }
4054             {
4055                 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4056                 \__enumext_keyans_show_left:n { #2 }
4057                 \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4058             }
4059         }
4060         \tl_use:N \l__enumext_label_font_style_v_tl
4061         \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4062         \__enumext_keyans_show_item_opt:
4063     }
4064 }

```

(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 *label*, which will have the same configuration as the `keyans` environment.

```

4065 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3

```

```

4066 {
4067   \stepcounter { enumXvi }
4068   \__enumext_anspic_body_dim:n { #3 }
4069   \bool_if:NTF \__enumext_keyans_pic_star_bool
4070   {
4071     \__enumext_anspic_label:nn { #1 } { #2 }
4072   }
4073   {
4074     \raisebox
4075     {
4076       -\dim_eval:n
4077       {
4078         \l__enumext_anspic_label_htdp_dim
4079         + \l__enumext_anspic_body_htdp_dim
4080         + \box_dp:N \strutbox
4081       }
4082     }
4083     [ Opt ] [ Opt ]
4084     {
4085       \__enumext_anspic_label:nn { #1 } { #2 }
4086     }
4087   }
4088 }
4089 %

```

(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.

```

4090 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4091 {
4092   \__enumext_anspic_start_list_tag:
4093   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4094   \__enumext_anspic_stop_start_list_tag:
4095   \l #3
4096   \__enumext_anspic_stop_list_tag:
4097 }

```

(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.

```

4098 \cs_new_protected:Nn \__enumext_anspic_print:n
4099 {
4100   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4101 }
4102 \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.

```

4103 \cs_new_protected:Nn \__enumext_anspic_row:n
4104 {
4105   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4106   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4107   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4108   \int_step_inline:nnn
4109   { \l__enumext_anspic_above_int + 1 }
4110   { \l__enumext_anspic_below_int }
4111   {
4112     \IfDocumentMetadataTF
4113     {
4114       \tag_suspend:n {minipage}
4115     } { }
4116     \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4117       \centering
4118       \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4119     \end{minipage}
4120     \IfDocumentMetadataTF
4121     {

```

```

4122         \tag_resume:n {minipage}
4123     } { }
4124 }
4125 \par
4126 }

```

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

12.42 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.

12.42.1 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:
4127 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4128 {
4129     \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4130     {
4131         \dim_set:Nn \__enumext_columns_sep_vii_dim
4132         {
4133             ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4134             / \__enumext_columns_vii_int
4135         }
4136     }
4137     \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4138     \dim_set:Nn \__enumext_item_width_vii_dim
4139     {
4140         ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4141         / \__enumext_columns_vii_int
4142         - \__enumext_labelwidth_vii_dim
4143         - \__enumext_labelsep_vii_dim
4144     }

```

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

```

4145     \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4146     {
4147         \dim_sub:Nn \__enumext_item_width_vii_dim
4148         {
4149             ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4150             / \__enumext_columns_vii_int
4151         }
4152         \dim_add:Nn \__enumext_columns_sep_vii_dim
4153         {
4154             \__enumext_rightmargin_vii_dim
4155         }
4156     }
4157 }

```

Same implementation for the `keyans*` environment.

```

4158 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4159 {
4160   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4161   {
4162     \dim_set:Nn \__enumext_columns_sep_viii_dim
4163     {
4164       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
4165       / \__enumext_columns_viii_int
4166     }
4167   }
4168   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
4169   \dim_set:Nn \__enumext_item_width_viii_dim
4170   {
4171     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
4172     / \__enumext_columns_viii_int
4173     - \__enumext_labelwidth_viii_dim
4174     - \__enumext_labelsep_viii_dim
4175   }
4176   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4177   {
4178     \dim_sub:Nn \__enumext_item_width_viii_dim
4179     {
4180       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_viii_int )
4181       / \__enumext_columns_viii_int
4182     }
4183     \dim_add:Nn \__enumext_columns_sep_viii_dim
4184     {
4185       \__enumext_rightmargin_viii_dim
4186     }
4187   }
4188 }

```

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

12.42.2 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.

```

4189 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4190 {
4191   \int_set:Nn \__enumext_joined_item_vii_int {#1}
4192   \int_compare:nNnT { \__enumext_joined_item_vii_int } > { \__enumext_columns_vii_int }
4193   {
4194     \msg_warning:nnee { enumext } { item-joined }
4195     { \int_use:N \__enumext_joined_item_vii_int }
4196     { \int_use:N \__enumext_columns_vii_int }
4197     \int_set:Nn \__enumext_joined_item_vii_int
4198     {
4199       \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + 1
4200     }
4201   }
4202   \int_compare:nNnT
4203   { \__enumext_joined_item_vii_int }
4204   >
4205   { \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + 1 }
4206   {
4207     \msg_warning:nnee { enumext } { item-joined-columns }
4208     { \int_use:N \__enumext_joined_item_vii_int }
4209     {
4210       \int_eval:n
4211       { \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + 1 }
4212     }
4213     \int_set:Nn \__enumext_joined_item_vii_int
4214     {
4215       \__enumext_columns_vii_int - \__enumext_item_column_pos_vii_int + 1
4216     }
4217   }
4218   \int_compare:nNnTF { \__enumext_joined_item_vii_int } > { 1 }
4219   {
4220     \int_set_eq:NN \__enumext_joined_item_aux_vii_int \__enumext_joined_item_vii_int

```

```

4221     \int_decr:N \l__enumext_joined_item_aux_vii_int
4222     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4223     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4224     \dim_set:Nn \l__enumext_joined_width_vii_dim
4225     {
4226         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4227         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4228             + \l__enumext_columns_sep_vii_dim
4229             )*\l__enumext_joined_item_aux_vii_int
4230     }
4231     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4232 }
4233 {
4234     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4235     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4236 }
4237 }

```

Same implementation for the `keyans*` environment.

```

4238 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4239 {
4240     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4241     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4242     {
4243         \msg_warning:nnee { enumext } { item-joined }
4244         { \int_use:N \l__enumext_joined_item_viii_int }
4245         { \int_use:N \l__enumext_columns_viii_int }
4246         \int_set:Nn \l__enumext_joined_item_viii_int
4247         {
4248             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4249         }
4250     }
4251     \int_compare:nNnT
4252     { \l__enumext_joined_item_viii_int }
4253     >
4254     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4255     {
4256         \msg_warning:nnee { enumext } { item-joined-columns }
4257         { \int_use:N \l__enumext_joined_item_viii_int }
4258         {
4259             \int_eval:n
4260             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4261         }
4262         \int_set:Nn \l__enumext_joined_item_viii_int
4263         {
4264             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4265         }
4266     }
4267     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4268     {
4269         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4270         \int_decr:N \l__enumext_joined_item_aux_viii_int
4271         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4272         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4273         \dim_set:Nn \l__enumext_joined_width_viii_dim
4274         {
4275             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4276             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4277                 + \l__enumext_columns_sep_viii_dim
4278                 )*\l__enumext_joined_item_aux_viii_int
4279         }
4280         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4281     }
4282     {
4283         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4284         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4285     }
4286 }

```

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

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

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

```

4287 \cs_new_protected:Nn \__enumext_start_mini_vii:
4288 {
4289   \dim_compare:nNtT { \__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4290   {
4291     \dim_set:Nn \__enumext_minipage_left_vii_dim
4292     {
4293       \linewidth
4294       - \__enumext_minipage_right_vii_dim
4295       - \__enumext_minipage_hsep_vii_dim
4296     }
4297     \bool_set_true:N \__enumext_minipage_active_vii_bool
4298     \dim_gset_eq:NN
4299       \__enumext_minipage_right_vii_dim
4300       \__enumext_minipage_right_vii_dim
4301     \__enumext_mini_addvspace_vii:
4302     \nointerlineskip\noindent
4303     \__enumext_mini_page{ \__enumext_minipage_left_vii_dim }
4304   }
4305 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:nn` to execute the `minipage` on the “right side”. At this point we will execute the `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§12.43).

```

4306 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4307 {
4308   \bool_if:NTF \__enumext_minipage_active_vii_bool
4309   {
4310     \__enumext_stop_list:
4311     \__enumext_stop_store_level_vii:
4312     \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4313     \end__enumext_mini_page
4314     \hfill
4315     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4316   }
4317   {
4318     \__enumext_stop_list:
4319     \__enumext_stop_store_level_vii:
4320   }
4321 }

```

(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.

```

4322 \__enumext_after_env:nn {enumext*}
4323 {
4324   \bool_if:NT \g__enumext_minipage_active_vii_bool
4325   {
4326     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4327     \legacy_if_gset_false:n { @minipage }
4328     \skip_vertical:N \c_zero_skip
4329     \par\addvspace { \g__enumext_minipage_right_skip }
4330     \bool_if:NF \g__enumext_minipage_center_vii_bool
4331     {
4332       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4333       {
4334         \centering
4335       }
4336     }
4337     \vbox_set_top:Nn \__enumext_miniright_code_vii_box

```

```

4338         {
4339             \tl_use:N \g__enumext_miniright_code_vii_tl
4340         }
4341         \box_use_drop:N \l__enumext_miniright_code_vii_box
4342         \skip_vertical:N \c_zero_skip
4343         \__enumext_endminipage:
4344         \par\addvspace{ \g__enumext_minipage_after_skip }
4345     }
4346     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4347     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4348     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4349     \dim_gzero:N \g__enumext_minipage_right_vii_dim
4350     \bool_gset_false:N \g__enumext_starred_bool
4351 }

```

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:

```

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

```

4352 \cs_new_protected:Nn \__enumext_start_mini_viii:
4353 {
4354     \dim_compare:nNt { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4355     {
4356         \dim_set:Nn \l__enumext_minipage_left_viii_dim
4357         {
4358             \linewidth
4359             - \l__enumext_minipage_right_viii_dim
4360             - \l__enumext_minipage_hsep_viii_dim
4361         }
4362         \bool_set_true:N \l__enumext_minipage_active_viii_bool
4363         \dim_gset_eq:NN
4364             \g__enumext_minipage_right_viii_dim
4365             \l__enumext_minipage_right_viii_dim
4366         \__enumext_mini_addvspace_viii:
4367         \nointerlineskip\noindent
4368         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4369     }
4370 }
4371 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4372 {
4373     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4374     {
4375         \__enumext_stop_list:
4376         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { } { }
4377         \end__enumext_mini_page
4378         \hfill
4379         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4380     }
4381     {
4382         \__enumext_stop_list:
4383     }
4384 }
4385 \__enumext_after_env:nn {keyans*}
4386 {
4387     \bool_if:NT \g__enumext_minipage_active_viii_bool
4388     {
4389         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4390         \par\addvspace { \g__enumext_minipage_right_skip }
4391         \bool_if:NF \g__enumext_minipage_center_viii_bool
4392         {
4393             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4394             {
4395                 \centering
4396             }
4397         }
4398         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4399         {
4400             \tl_use:N \g__enumext_miniright_code_viii_tl
4401         }
4402         \box_use_drop:N \l__enumext_miniright_code_viii_box
4403         \end__enumext_mini_page
4404         \par\addvspace{ \g__enumext_minipage_after_skip }
4405     }

```



```

4406 \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4407 \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4408 \tl_gclear:N \g__enumext_miniright_code_viii_tl
4409 \dim_gzero:N \g__enumext_minipage_right_viii_dim
4410 }

```

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

12.42.4 Redefining \footnote command

__enumext_footnotetext:nn
 __enumext_renew_footnote:
 __enumext_print_footnote:

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](#).

```

4411 \cs_new_protected:Nn \__enumext_footnotetext:nn
4412 {
4413   \footnotetext[#1]{#2}
4414 }
4415 \cs_new_protected:Nn \__enumext_renew_footnote:
4416 {
4417   \seq_gclear:N \g__enumext_footnote_arg_seq
4418   \seq_gclear:N \g__enumext_footnote_int_seq
4419   \RenewDocumentCommand \footnote { o +m }
4420   {
4421     \tl_if_novalue:nTF {##1}
4422     {
4423       \stepcounter{footnote}
4424       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4425     }
4426     {
4427       \int_gset:Nn \g__enumext_footnote_int { ##1 }
4428     }
4429     \footnotemark [ \g__enumext_footnote_int ]
4430     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4431     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4432   }
4433 }
4434 \cs_new_protected:Nn \__enumext_print_footnote:
4435 {
4436   \seq_if_empty:NF \g__enumext_footnote_int_seq
4437   {
4438     \seq_map_pairwise_function:NNN
4439     \g__enumext_footnote_int_seq
4440     \g__enumext_footnote_arg_seq
4441     \__enumext_footnotetext:nn
4442   }
4443 }

```

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

12.43 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 pdfTeX and LuaTeX.

```

4444 \NewDocumentEnvironment{enumext*}{ o }
4445 {
4446   \__enumext_safe_exec_vii:
4447   \__enumext_parse_keys_vii:n {#1}
4448   \__enumext_before_list_vii:
4449   \__enumext_start_store_level_vii:
4450   \__enumext_start_list:nn { }
4451   {
4452     \__enumext_list_arg_two_vii:
4453     \__enumext_before_keys_exec_vii:
4454   }
4455   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4456   \__enumext_starred_columns_set_vii:
4457   \item[] \scan_stop:
4458   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:

```

```

4459     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4460     \ignorespaces
4461   }
4462   {
4463     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4464     \__enumext_stop_item_tmp_vii:
4465     \__enumext_remove_extra_parsep_vii:
4466     \__enumext_after_list_vii:
4467   }

```

(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.

```

4468 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4469   {
4470     \__enumext_internal_mini_page:
4471     \__enumext_is_not_nested:
4472     \int_incr:N \l__enumext_level_h_int
4473     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4474     {
4475       \msg_error:nn { enumext } { nested }
4476     }
4477     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4478     {
4479       \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4480     }
4481     \bool_set_true:N \l__enumext_starred_bool
4482     \bool_set_false:N \l__enumext_standar_bool
4483     \__enumext_is_on_first_level:
4484   }

```

(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.

```

4485 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4486   {
4487     \tl_if_novalue:nF {#1}
4488     {
4489       \str_clear:N \l__enumext_series_str
4490       \keys_set:nn { enumext / enumext* } {#1}
4491       \__enumext_parse_series:n {#1}
4492       \__enumext_store_active_keys_vii:n {#1}
4493     }
4494   }

```

(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*`.

```

4495 \cs_new_protected:Nn \__enumext_before_list_vii:
4496   {
4497     \__enumext_vspace_above_vii:
4498     \__enumext_check_ans_active:
4499     \__enumext_before_args_exec_vii:
4500     \__enumext_start_mini_vii:
4501   }

```

(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:` (§12.42.3) 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.

```

4502 \cs_new_protected:Nn \__enumext_after_list_vii:
4503 {
4504   \__enumext_stop_mini_vii:
4505   \__enumext_after_stop_list_vii:
4506   \__enumext_check_ans_key_hook:
4507   \__enumext_vspace_below_vii:
4508   \bool_set_false:N \__enumext_starred_bool
4509   \__enumext_resume_save_counter:
4510 }

```

(End of definition for `__enumext_after_list_vii:`.)

`__enumext_start_store_level_vii:` The `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:` functions activate the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment if `enumext*` are nested in `enumext`.

`__enumext_stop_store_level_vii:`

```

4511 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4512 {
4513   \bool_if:NT \__enumext_store_active_bool
4514   {
4515     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4516     {
4517       \__enumext_store_level_open_vii:
4518     }
4519   }
4520 }
4521 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4522 {
4523   \bool_if:NT \__enumext_store_active_bool
4524   {
4525     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4526     {
4527       \__enumext_store_level_close_vii:
4528     }
4529   }
4530 }

```

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

12.43.1 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.

```

4531 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4532 {
4533   \skip_horizontal:n { -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim }
4534 }

```

(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`.

```

4535 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4536 {
4537   \__enumext_stop_item_tmp_vii:
4538   \int_incr:N \__enumext_item_column_pos_vii_int
4539   \int_gincr:N \g__enumext_item_count_all_vii_int
4540   \__enumext_item_peek_args_vii:
4541 }

```

(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).

```
4542 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4543 {
4544   \peek_meaning:NTF (
4545     { \__enumext_joined_item_vii:w }
4546     { \__enumext_joined_item_vii:w (1) }
4547   }
```

(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`.

```
4548 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4549 {
4550   \__enumext_starred_joined_item_vii:n {#1}
4551   \peek_meaning_remove:NTF *
4552     { \__enumext_starred_item_vii:w }
4553     { \__enumext_standar_item_vii:w }
4554 }
```

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

```
4555 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4556 {
4557   \bool_set_false:N \l__enumext_item_starred_vii_bool
4558   \peek_meaning:NTF [
4559     {
4560       \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4561       \__enumext_start_item_vii:w
4562     }
4563     {
4564       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4565       \legacy_if_set_true:n { @noitemarg }
4566       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4567     }
4568 }
```

(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* [<symbol>]` and `\item* [<symbol>] [<offset>]`.

```
4569 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4570 {
4571   \bool_set_true:N \l__enumext_item_starred_vii_bool
4572   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4573   \peek_meaning:NTF [
4574     { \__enumext_starred_item_vii_aux_i:w }
4575     { \__enumext_starred_item_vii_aux_ii:w }
4576   }
4577   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4578   {
4579     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4580     \__enumext_starred_item_vii_aux_ii:w
4581   }
4582   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4583   {
4584     \peek_meaning:NTF [
4585       { \__enumext_starred_item_vii_aux_iii:w }
```

```

4586     {
4587       \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4588       \legacy_if_set_true:n { @noitemarg }
4589       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4590     }
4591   }
4592   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4593   {
4594     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4595     \legacy_if_set_true:n { @noitemarg }
4596     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4597   }

```

(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*` 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.

```

4598   \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4599   {
4600     \legacy_if:nT { @noitemarg }
4601     {
4602       \legacy_if_set_false:n { @noitemarg }
4603       \legacy_if:nT { @nmbrrlist }
4604       {
4605         \IfDocumentMetadataTF
4606         {
4607           \bool_if:NT \l__enumext_hyperref_bool
4608           {
4609             \legacy_if_set_true:n { @hyper@item }
4610           }
4611         } { }
4612         \refstepcounter{enumXvii}
4613         \bool_if:NT \l__enumext_check_answers_bool
4614         {
4615           \int_gincr:N \g__enumext_item_number_int
4616           \bool_set_true:N \l__enumext_item_number_bool
4617         }
4618       }
4619     }
4620     \bool_if:NT \l__enumext_item_starred_vii_bool
4621     {
4622       \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4623       {
4624         \tl_gset_eq:NN
4625         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4626       }
4627       \mode_leave_vertical:
4628       \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4629       \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4630       \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4631       \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4632     }
4633     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4634     {
4635       \tl_use:N \l__enumext_label_font_style_vii_tl
4636       \bool_if:NTF \l__enumext_wrap_label_vii_bool
4637       {
4638         \__enumext_wrapper_label_vii:n {#1}
4639       }
4640       { #1 }
4641     }
4642     \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces

```

```
4643 }
```

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

12.43.2 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
```

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 its `contents` in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
4644 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4645 {
4646   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4647   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4648   {
4649     \l__enumext_joined_width_vii_dim
4650     + \l__enumext_labelwidth_vii_dim
4651     + \l__enumext_labelsep_vii_dim
4652   }
```

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

```
4653   \IfDocumentMetadataTF { }
4654   {
4655     \bool_if:NF \l__enumext_footnotes_key_bool
4656     {
4657       \__enumext_renew_footnote:
4658     }
4659   }
```

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

```
4660   \__enumext_start_list_tag:n {enumext*}
4661   \__enumext_fake_make_label_vii:n {#1}
4662   \__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`.

```
4663   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4664   \tl_use:N \l__enumext_after_list_args_vii_tl
4665   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4666   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4667   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4668 }
```

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

```
\__enumext_stop_item_vii:
```

The `__enumext_stop_item_vii:` function will finish the fetching `\item` and its `content` by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```
4669 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4670 {
4671   \__enumext_endminipage:
4672   \__enumext_stop_list_tag:n {enumext*}
4673   \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print the `contents` of the *box* along with `\footnote`.

```
4674   \int_set:Nn \hbadness { 10000 }
4675   \box_use_drop:N \l__enumext_item_text_vii_box
4676   \IfDocumentMetadataTF { }
4677   {
4678     \bool_if:NF \l__enumext_footnotes_key_bool
4679     {
4680       \__enumext_print_footnote:
4681     }
4682   }
```

Finally set the *vertical* and *horizontal* spaces between rows and columns.

```

4683 \int_compare:nNnTF
4684 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4685 {
4686   \par\noindent
4687   \int_zero:N \l__enumext_item_column_pos_vii_int
4688 }
4689 {
4690   \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4691 }
4692 }

```

(End of definition for `__enumext_stop_item_vii:`)

`__enumext_remove_extra_parsep_vii:` Remove the *vertical space* equal to `\parsep=\itemsep` when the total number of items is divisible by the number of items in the last row of the environment. Here the use of `\unskip` or `\removeatlastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in *(vertical mode)*.

```

4693 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4694 {
4695   \int_compare:nNnTF
4696   {
4697     \int_mod:nn
4698     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4699   }
4700   =
4701   { 0 }
4702   {
4703     \para_end:
4704     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4705     \skip_vertical:N \c_zero_skip
4706     \int_gzero:N \g__enumext_item_count_all_vii_int
4707   }
4708 }

```

(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`.

```

4709 \__enumext_after_env:nn {enumext*}
4710 {
4711   \__enumext_execute_after_env:
4712 }

```

12.44 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.

```

4713 \NewDocumentEnvironment{keyans*}{o}
4714 {
4715   \__enumext_safe_exec_viii:
4716   \__enumext_parse_keys_viii:n {#1}
4717   \__enumext_before_list_viii:
4718   \__enumext_start_list:nn { }
4719   {
4720     \__enumext_list_arg_two_viii:
4721     \__enumext_before_keys_exec_viii:
4722   }
4723   \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { } { }
4724   \__enumext_starred_columns_set_viii:
4725   \item[] \scan_stop:
4726   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4727   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4728   \ignorespaces
4729 }
4730 {
4731   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { } { }
4732   \__enumext_stop_item_tmp_viii:

```



```

4733     \__enumext_remove_extra_parsep_viii:
4734     \__enumext_check_starred_cmd:n { item }
4735     \__enumext_after_list_viii:
4736 }

```

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

`__enumext_safe_exec_viii:` The `__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `\l__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```

4737 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4738 {
4739     \bool_if:NF \l__enumext_store_active_bool
4740     {
4741         \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4742     }
4743     \int_incr:N \l__enumext_keyans_level_h_int
4744     \int_compare:nNt { \l__enumext_keyans_level_h_int } > { 1 }
4745     {
4746         \msg_error:nn { enumext } { nested }
4747     }
4748     \__enumext_keyans_name_and_start:
4749     \bool_if:NT \l__enumext_starred_bool
4750     {
4751         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4752     }
4753     \bool_set_true:N \l__enumext_starred_bool
4754     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4755     \bool_set_false:N \l__enumext_store_active_bool
4756     \int_compare:nNt { \l__enumext_level_int } > { 1 }
4757     {
4758         \msg_error:nn { enumext } { keyans-wrong-level }
4759     }
4760 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

4761 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4762 {
4763     \tl_if_novalue:NF {#1}
4764     {
4765         \keys_set:nn { enumext / keyans* } {#1}
4766     }
4767 }

```

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

`__enumext_before_list_viii:` The function `__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{code}` defined by the `before*` key if it is active, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

4768 \cs_new_protected:Nn \__enumext_before_list_viii:
4769 {
4770     \__enumext_vspace_above_viii:
4771     \__enumext_before_args_exec_viii:
4772     \__enumext_start_mini_viii:
4773 }

```

(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.

```

4774 \cs_new_protected:Nn \__enumext_after_list_viii:
4775 {
4776     \__enumext_stop_mini_viii:

```

```

4777     \__enumext_after_stop_list_viii:
4778     \__enumext_vspace_below_viii:
4779 }

```

(End of definition for __enumext_after_list_viii:.)

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

```

\__enumext_first_item_tmp_viii: The \__enumext_first_item_tmp_viii: function will remove horizontal space equal to \labelwidth
plus \labelsep to the left of the first \item in the environment at the point of execution of this function, where
it is equal to the \__enumext_stop_item_tmp_viii: function inside the environment body definition.
4780 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4781 {
4782     \skip_horizontal:n { -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim }
4783 }

```

(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.

```

```

4784 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4785 {
4786     \__enumext_stop_item_tmp_viii:
4787     \int_incr:N \l__enumext_item_column_pos_viii_int
4788     \int_gincr:N \g__enumext_item_count_all_viii_int
4789     \__enumext_item_peek_args_viii:
4790 }

```

(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(\langle number \rangle). Look for the argu-
ment “(”, if it is present we will call the function \__enumext_joined_item_viii:w (\langle number \rangle), which is
in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```

```

4791 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4792 {
4793     \peek_meaning:NTF (
4794         { \__enumext_joined_item_viii:w }
4795         { \__enumext_joined_item_viii:w (1) }
4796     }

```

(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.

```

```

4797 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4798 {
4799     \__enumext_starred_joined_item_viii:n {#1}
4800     \peek_meaning_remove:NTF *
4801     { \__enumext_starred_item_viii:w }
4802     { \__enumext_standar_item_viii:w }
4803 }

```

(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].

```

4804 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4805 {
4806   \bool_set_false:N \l__enumext_item_starred_viii_bool
4807   \peek_meaning:NTF [
4808     {
4809       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4810       \__enumext_start_item_viii:w
4811     }
4812     {
4813       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4814       \legacy_if_set_true:n { @noitemarg }
4815       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4816     }
4817   }

```

(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⟩]`.

__enumext_starred_item_viii_aux_i:w

__enumext_starred_item_viii_aux_ii:w

```

4818 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4819 {
4820   \bool_set_true:N \l__enumext_item_starred_viii_bool
4821   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4822   \peek_meaning:NTF [
4823     { \__enumext_starred_item_viii_aux_i:w }
4824     { \__enumext_starred_item_viii_aux_ii:w }
4825   }

```

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.

```

4826 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4827 {
4828   \tl_clear:N \l__enumext_store_current_label_tl
4829   \tl_if_novalue:nF { #1 }
4830   {
4831     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4832     {
4833       \tl_put_right:Ne \l__enumext_store_current_label_tl
4834       {
4835         \l__enumext_store_keyans_item_opt_sep_tl
4836       }
4837       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4838     }
4839     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4840   }
4841   \__enumext_starred_item_viii_aux_ii:w
4842 }
4843 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4844 {
4845   \legacy_if_set_true:n { @noitemarg }
4846   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4847 }

```

(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.

```

4848 \cs_new_protected:Nn \__enumext_starred_item_exec:
4849 {

```

```

4850 \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4851 \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4852 \__enumext_keyans_store_ref:
4853 \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4854 \__enumext_keyans_addto_seq_link:
4855 \int_gincr:N \g__enumext_check_starred_cmd_int
4856 \bool_if:NT \l__enumext_show_answer_bool
4857 {
4858   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4859 }
4860 \bool_if:NT \l__enumext_show_position_bool
4861 {
4862   \tl_set:Ne \l__enumext_mark_answer_sym_tl
4863   {
4864     \group_begin:
4865     \exp_not:N \normalfont
4866     \exp_not:N \footnotesize [ \int_eval:n
4867       {
4868         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4869       }
4870     ]
4871     \group_end:
4872   }
4873   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4874 }
4875 }

```

(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.

```

4876 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4877 {
4878   \legacy_if:nT { @noitemarg }
4879   {
4880     \legacy_if_set_false:n { @noitemarg }
4881     \legacy_if:nT { @nmbrrlist }
4882     {
4883       \refstepcounter{enumXviii}
4884     }
4885   }
4886   \bool_if:NT \l__enumext_item_starred_viii_bool
4887   {
4888     \__enumext_starred_item_exec:
4889   }
4890   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4891   {
4892     \tl_use:N \l__enumext_label_font_style_viii_tl
4893     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4894     {
4895       \__enumext_wrapper_label_viii:n {#1}
4896     }
4897     { #1 }
4898   }
4899   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
4900 }

```

(End of definition for __enumext_fake_make_label_viii:n.)

12.44.2 Real definition of \item in keyans*

__enumext_start_item_viii:w

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

```

4901 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4902 {
4903   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4904   \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4905   {
4906     \l__enumext_joined_width_viii_dim
4907     + \l__enumext_labelwidth_viii_dim
4908     + \l__enumext_labelsep_viii_dim
4909   }
4910   \IfDocumentMetadataTF { }
4911   {

```

```

4912         \bool_if:NF \l__enumext_footnotes_key_bool
4913         {
4914             \__enumext_renew_footnote:
4915         }
4916     }
4917     \__enumext_start_list_tag:n {keyans*}
4918     \__enumext_fake_make_label_viii:n {#1}
4919     \__enumext_stop_start_list_tag:
4920     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4921     \tl_use:N \l__enumext_after_list_args_viii_tl
4922     \bool_if:NT \l__enumext_item_starred_viii_bool
4923     {
4924         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4925         \__enumext_keyans_show_item_opt:
4926         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4927     }
4928     {
4929         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4930     }
4931     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4932     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4933 }

```

(End of definition for __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The __enumext_stop_item_viii: function will finish the fetching \item and its *content* by closing the minipage environment and the horizontal box. Here we will reduce the warnings a bit by setting the value of \hbadness to 10000, print the *contents* of the box along with \footnote and finally set the vertical and horizontal spaces between rows and columns.

```

4934 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4935 {
4936     \__enumext_endminipage:
4937     \__enumext_stop_list_tag:n {keyans*}
4938     \hbox_set_end:
4939     \int_set:Nn \hbadness { 10000 }
4940     \box_use_drop:N \l__enumext_item_text_viii_box
4941     \IfDocumentMetadataTF { }
4942     {
4943         \bool_if:NF \l__enumext_footnotes_key_bool
4944         {
4945             \__enumext_print_footnote:
4946         }
4947     }
4948     \int_compare:nNnTF
4949     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4950     {
4951         \par\noindent
4952         \int_zero:N \l__enumext_item_column_pos_viii_int
4953     }
4954     {
4955         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4956     }
4957 }

```

(End of definition for __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii: Finally we will remove the vertical space equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```

4958 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4959 {
4960     \int_compare:nNnT
4961     {
4962         \int_mod:nn
4963         { \g__enumext_item_count_all_viii_int }
4964         { \l__enumext_columns_viii_int }
4965     }
4966     =
4967     { 0 }
4968     {
4969         \para_end:
4970         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }

```

```

4971         \skip_vertical:N \c_zero_skip
4972         \int_gzero:N \g__enumext_item_count_all_viii_int
4973     }
4974 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

12.45 The command \getkeyans

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form $\langle store\ name : position \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from *prop list* defined by *save-ans* key.

```

4975 \NewDocumentCommand \getkeyans { m }
4976 {
4977     \exp_args:Ne \__enumext_getkeyans_aux:n
4978     { \tl_to_str:e { \text_expand:n {#1} } }
4979 }

```

(End of definition for \getkeyans. This function is documented on page 17.)

`__enumext_getkeyans_aux:n`

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

```

4980 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4981 {
4982     \str_if_in:nnTF {#1} { : }
4983     {
4984         \use:e
4985         {
4986             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4987             { {##1} {##2} }
4988         }
4989         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4990     }
4991     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4992 }

```

(End of definition for __enumext_getkeyans_aux:n.)

`__enumext_getkeyans:nn`

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 $\langle argument \rangle$ from *prop list*.

```

4993 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4994 {
4995     \prop_if_exist:cTF { g__enumext_#1_prop }
4996     {
4997         \prop_item:cn { g__enumext_#1_prop }{#2}
4998     }
4999     {
5000         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5001     }
5002 }

```

(End of definition for __enumext_getkeyans:nn.)

12.46 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 $\langle filtered\ keys \rangle$ 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 $\langle keys \rangle$ 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⟩]`.

```

5003 \keys_define:nn { enumext / print }
5004 {
5005     print* .code:n      = \keys_precompile:neN { enumext / enumext* }
5006                     { \__enumext_filter_save_key:n {#1} }
5007                     \l__enumext_print_keyans_starred_tl, % starred cmd
5008     print* .initial:n   = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
5009     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }

```

```

5010         { \__enumext_filter_save_key:n {#1} }
5011         \__enumext_print_keyans_i_tl,
5012     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5013     print-2 .code:n    = \keys_precompile:neN { enumext / level-2 }
5014         { \__enumext_filter_save_key:n {#1} }
5015         \__enumext_print_keyans_ii_tl,
5016     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5017     print-3 .code:n    = \keys_precompile:neN { enumext / level-3 }
5018         { \__enumext_filter_save_key:n {#1} }
5019         \__enumext_print_keyans_iii_tl,
5020     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5021     print-4 .code:n    = \keys_precompile:neN { enumext / level-4 }
5022         { \__enumext_filter_save_key:n {#1} }
5023         \__enumext_print_keyans_iv_tl,
5024     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5025     print-* .code:n    = \keys_precompile:neN { enumext / enumext* }
5026         { \__enumext_filter_save_key:n {#1} }
5027         \__enumext_print_keyans_vii_tl, % starred nested
5028     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5029 }

```

• The reason for storing $\langle keys \rangle$ in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

`\printkeyans` 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`.

```

5030 \NewDocumentCommand \printkeyans { s O{ } m }
5031 {
5032     \group_begin:
5033     \tl_use:N \__enumext_print_keyans_i_tl
5034     \tl_use:N \__enumext_print_keyans_ii_tl
5035     \tl_use:N \__enumext_print_keyans_iii_tl
5036     \tl_use:N \__enumext_print_keyans_iv_tl
5037     \tl_use:N \__enumext_print_keyans_vii_tl
5038     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5039     \group_end:
5040 }

```

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

`__enumext_printkeyans:nnn` The internal function `__enumext_printkeyans:nnn` will check for the existence of the *sequence*, if it does not exist it will return an error message, then it will check if not empty.

```

5041 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5042 {
5043     \seq_if_exist:cTF { g__enumext_#3_seq }
5044     {
5045         \seq_if_empty:cF { g__enumext_#3_seq }
5046         {

```

If the *starred argument* `*` is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `__enumext_print_keyans_starred_tl` that contains the default $\langle keys \rangle$ for the environment `enumext*`, it will open the environment `enumext*` passing the *optional argument* to the “*first level*”, set the key *base-fix* and then will map the *sequence*.

```

5047         \bool_if:nTF {#1}
5048         {
5049             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5050             {
5051                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5052             }
5053             {
5054                 \tl_use:N \__enumext_print_keyans_starred_tl
5055                 \bool_set_true:N \__enumext_base_line_fix_bool
5056                 \bool_set_true:N \__enumext_print_keyans_star_bool
5057                 \begin{enumext*}[#2]
5058                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5059                     \end{enumext*}
5060                 \bool_set_false:N \__enumext_base_line_fix_bool
5061                 \bool_set_false:N \__enumext_print_keyans_star_bool
5062             }
5063         }

```


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

```

5064         {
5065             \begin{enumext}[#2]
5066                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5067             \end{enumext}
5068         }
5069     }
5070 }
5071 {
5072     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5073 }
5074 }

```

(End of definition for `__enumext_printkeyans:nnn`.)

12.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the $\langle keys \rangle$ 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 $\langle keys \rangle$ that complicate us.

The function `__enumext_filter_first_level:n` will be in charge of filtering the $\langle keys \rangle$ passed to the environment `enumext*` and “first level” of the environment `enumext`.

```

5075 \cs_new:Npn \__enumext_filter_first_level:n #1
5076 {
5077     \use:e
5078     {
5079         \keyval_parse:NNn
5080             \__enumext_filter_first_level_key:n
5081             \__enumext_filter_first_level_pair:nn {#1}
5082     }
5083 }

```

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

```

5084 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5085 {
5086     \str_case:nnF {#1}
5087     {
5088         { resume } {}
5089         { resume* } {}
5090     }
5091     { , { \exp_not:n {#1} } }
5092 }

```

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

```

5093 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5094 {
5095     \str_case:nnF {#1}
5096     {
5097         { series } {}
5098         { resume } {}
5099         { save-ans } {}
5100     }
5101     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5102 }

```

(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 $\langle keys \rangle$ to access from `\setenumext`.

```

5103 \keys_define:nn { enumext / meta-families }
5104 {
5105     enumext-1 .code:n =
5106         {
5107             \keys_set:ne { enumext / level-1 }
5108             {
5109                 \__enumext_filter_first_level:n {#1}
5110             }
5111         } ,
5112     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,

```

```

5113 enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5114 enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5115 keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5116 enumext* .code:n =
5117 {
5118     \keys_set:ne { enumext / enumext* }
5119     {
5120         \__enumext_filter_first_level:n {#1}
5121     }
5122 } ,
5123 keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5124 print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5125 print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5126 print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5127 print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5128 print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5129 print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5130 unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5131 }

```

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

```

5132 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5133 {
5134     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5135     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5136 }

```

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

```

5137 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5138 {
5139     \seq_clear:N \l__enumext_setkey_tmpa_seq
5140     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5141     \int_set:Nn \l__enumext_setkey_tmpa_int
5142     {
5143         \seq_count:N \l__enumext_setkey_tmpb_seq
5144     }
5145     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5146     {
5147         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5148         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5149         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5150         {
5151             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5152         }
5153     }
5154     {
5155         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5156     }
5157     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5158     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5159     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5160     {
5161         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5162     }
5163 }

```

(End of definition for `\setenumext`. This function is documented on page 6.)

`__enumext_set_parse:n`
`__enumext_set_error:n`

Internal functions used by the `\setenumext` command.

```

5164 \cs_new_protected:Npn \__enumext_set_parse:n #1
5165 {
5166     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5167     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5168     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5169     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5170     {
5171         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5172         { \tl_trim_spaces:n {#1} }
5173     }
5174     { \__enumext_set_error:nn {#1} { } }
5175 }

```

```

5176 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5177 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for __enumext_set_parse:n and __enumext_set_error:nn.)

12.48 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

```

```

5178 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5179 {
5180   {enumext,1} = level-1,
5181   {enumext,2} = level-2,
5182   {enumext,3} = level-3,
5183   {enumext,4} = level-4,
5184   {enumext*} = enumext*
5185 }

```

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

```

5186 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5187 {
5188   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5189   { \msg_error:nn { enumext } { prohibited-unknown } }
5190   {
5191     \bool_if:nTF {#1}
5192     {
5193       \int_step_inline:nn { 4 }
5194       { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5195       \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5196     }
5197     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5198   }
5199 }

```

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

```

5200 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5201 {
5202   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5203   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5204   \prop_get:NVNTF
5205   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5206   { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5207   {
5208     \msg_error:nnn { enumext } { unknown-set } {#1}
5209     \use_none:nn
5210   }
5211 }
5212 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5213 {
5214   \bool_lazy_or:nnTF
5215   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5216   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5217   { \msg_error:nnn { enumext } { already-defined } {#2} }
5218   {
5219     \keys_define:nn { enumext / #1 }
5220     {
5221       #2 .meta:n = {#3},
5222       #2 .value_forbidden:n = true
5223     }
5224   }
5225 }
5226 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 6.)

12.49 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](#).

`\foreachkeyans`

`__enumext_parse_foreach_keys:nn`

`__enumext_parse_foreach_keys:n`

`__enumext_foreach_keyans:nn`

`__enumext_foreach_add_body:n`

We define a set of *⟨keys⟩* for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```

5227 \keys_define:nn { enumext / foreach }
5228 {
5229   before .tl_set:N = \l__enumext_foreach_before_tl,
5230   before .value_required:n = true,
5231   after .tl_set:N = \l__enumext_foreach_after_tl,
5232   after .value_required:n = true,
5233   start .int_set:N = \l__enumext_foreach_start_int,
5234   start .value_required:n = true,
5235   stop .int_set:N = \l__enumext_foreach_stop_int,
5236   stop .value_required:n = true,
5237   step .int_set:N = \l__enumext_foreach_step_int,
5238   step .value_required:n = true,
5239   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5240   wrapper .value_required:n = true,
5241   sep .tl_set:N = \l__enumext_foreach_sep_tl,
5242   sep .value_required:n = true,
5243   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5244 }
5245 \keys_precompile:nnN { enumext / foreach }
5246 {
5247   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5248 }
5249 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown *⟨keys⟩*.

```

5250 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5251 {
5252   \tl_if_blank:nTF {#2}
5253   {
5254     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5255   }
5256   {
5257     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5258   }
5259 }
5260 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5261 {
5262   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5263 }

```

We create the command.

```

5264 \NewDocumentCommand \foreachkeyans { +0{} m }
5265 {
5266   \__enumext_foreach_keyans:nn {#1} {#2}
5267 }

```

Finally the internal functions `__enumext_foreach_keyans:nn` and `__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5268 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5269 {
5270   \tl_use:N \g__enumext_foreach_default_keys_tl
5271   \keys_set:nn { enumext / foreach } {#1}
5272   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5273   \prop_if_exist:cF { g__enumext_#2_prop }
5274   {
5275     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5276   }
5277   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5278   {
5279     \int_set:Nn \l__enumext_foreach_stop_int
5280     { \prop_count:c { g__enumext_#2_prop } }
5281   }
5282   \seq_clear:N \l__enumext_foreach_print_seq
5283   \int_step_function:nnnN

```

```

5284     { \l__enumext_foreach_start_int }
5285     { \l__enumext_foreach_step_int }
5286     { \l__enumext_foreach_stop_int }
5287     \__enumext_foreach_add_body:n
5288     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5289   }
5290 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5291 {
5292   \seq_put_right:Ne \l__enumext_foreach_print_seq
5293   {
5294     \exp_not:V \l__enumext_foreach_before_tl
5295     \__enumext_foreach_wrapper:n
5296     {
5297       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5298     }
5299     \exp_not:V \l__enumext_foreach_after_tl
5300   }
5301 }

```

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

12.50 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5302 \msg_new:nnn { enumext } { package-load }
5303 {
5304   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5305 }
5306 \msg_new:nnn { enumext } { package-not-load }
5307 {
5308   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5309 }
5310 \msg_new:nnn { enumext } { package-load-foot }
5311 {
5312   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5313 }

```

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

```

5314 \msg_new:nnn { enumext } { counters }
5315 {
5316   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5317   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5318 }

```

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

```

5319 \msg_new:nnn { enumext } { unknown-choice }
5320 {
5321   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5322 }

```

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

```

5323 \msg_new:nnnn { enumext } { anskey-env-error }
5324 {
5325   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
5326   'enumext' ~ package, ~ It~ is~ already~ defined.
5327 }
5328 {
5329   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5330   for ~ the ~ 'save-ans' ~ key.\\
5331 }

```

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

```

5332 \msg_new:nnn { enumext } { store-prop }
5333 {
5334   * ~ Package ~ enumext: ~ Creating ~
5335   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5336 }
5337 \msg_new:nnn { enumext } { store-seq }
5338 {
5339   * ~ Package ~ enumext: ~ Creating ~
5340   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5341 }
5342 \msg_new:nnn { enumext } { store-int }

```

```

5343 {
5344     * ~ Package ~ enumext: ~ Creating ~
5345     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5346 }
5347 \msg_new:nnn { enumext } { prop-seq-int-hook }
5348 {
5349     * ~ Package ~ enumext: ~ Elements ~ in ~
5350     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\
5351     * ~ Package ~ enumext: ~ Elements ~ in ~
5352     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\
5353     * ~ Package ~ enumext: ~ Value ~ off ~
5354     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5355 }
5356 \msg_new:nnn { enumext } { item-answer-hook }
5357 {
5358     * ~ Package ~ enumext: ~ Value ~ off ~
5359     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
5360     * ~ Package ~ enumext: ~ Value ~ off ~
5361     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
5362     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5363 }

```

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

```

5364 \msg_new:nnn { enumext } { invalid-key }
5365 {
5366     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5367 }
5368 \msg_new:nnn { enumext } { unknown-key-family }
5369 {
5370     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5371 }

```

Messages used in length calculation.

```

5372 \msg_new:nnn { enumext } { width-negative }
5373 {
5374     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5375     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5376 }
5377 \msg_new:nnn { enumext } { width-zero }
5378 {
5379     Invalid ~ '#1=#2' ~ \msg_line_context:.\
5380     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5381 }

```

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

```

5382 \msg_new:nnn { enumext } { list-lengths }
5383 {
5384     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
5385     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5386     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5387     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5388     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5389     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5390     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5391     \__enumext_show_length:nnn { skip } { topsep } {#1}
5392     \__enumext_show_length:nnn { skip } { parsep } {#1}
5393     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5394     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5395     ****
5396 }

```

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

```

5397 \msg_new:nnn { enumext } { list-lengths-not-nested }
5398 {
5399     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
5400     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5401     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5402     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5403     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5404     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5405     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5406     \__enumext_show_length:nnn { skip } { topsep } {#1}
5407     \__enumext_show_length:nnn { skip } { parsep } {#1}

```

```

5408     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5409     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5410     *****
5411 }

```

Messages used by `ref` key.

```

5412 \msg_new:nnn { enumext } { key-ref-empty }
5413 {
5414     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5415 }

```

Messages used by `save-ans` key.

```

5416 \msg_new:nnn { enumext } { save-ans-empty }
5417 {
5418     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5419 }
5420 \msg_new:nnn { enumext } { save-ans-log }
5421 {
5422     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5423 }
5424 \msg_new:nnn { enumext } { save-ans-log-hook }
5425 {
5426     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5427 }
5428 \msg_new:nnn { enumext } { save-ans-hook }
5429 {
5430     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5431 }

```

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

```

5432 \msg_new:nnn { enumext } { need-save-ans }
5433 {
5434     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5435 }
5436 \msg_new:nnn { enumext } { items-same-answer }
5437 {
5438     *****\
5439     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5440     for ~ \c_left_brace_str #2 \c_right_brace_str\
5441     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5442     'OK', ~ all ~ items ~ with ~ answer.\
5443     *****
5444 }
5445 \msg_new:nnn { enumext } { item-greater-answer }
5446 {
5447     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5448     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5449     Items ~ > ~ Answers.
5450 }
5451 \msg_new:nnn { enumext } { item-less-answer }
5452 {
5453     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5454     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5455     Items ~ < ~ Answers.
5456 }

```

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

```

5457 \msg_new:nnn { enumext } { missing-starred }
5458 {
5459     Missing ~ '\c_backslash_str #1*' ~ #2.
5460 }
5461 \msg_new:nnn { enumext } { many-starred }
5462 {
5463     Many ~ '\c_backslash_str #1*' ~ #2.
5464 }

```

Messages used by `\printkeyans*` command.

```

5465 \msg_new:nnn { enumext } { print-starred }
5466 {
5467     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5468     #2 ~ environment ~ \msg_line_context:.
5469 }

```


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

```
5470 \msg_new:nnn { enumext } { list-too-deep }
5471 {
5472   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5473   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5474 }
```

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

```
5475 \msg_new:nnn { enumext } { anskey-unnumber-item }
5476 {
5477   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5478 }
5479 \msg_new:nnn { enumext } { anskey-already-stored }
5480 {
5481   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5482 }
5483 \msg_new:nnn { enumext } { anskey-empty-arg }
5484 {
5485   Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5486 }
5487 \msg_new:nnn { enumext } { anskey-wrong-place }
5488 {
5489   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5490   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5491 }
5492 \msg_new:nnn { enumext } { anskey-nested }
5493 {
5494   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5495 }
5496 \msg_new:nnn { enumext } { anskey-math-mode }
5497 {
5498   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5499 }
5500 \msg_new:nnn { enumext } { anskey-env-wrong }
5501 {
5502   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5503 }
5504 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5505 {
5506   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5507   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5508 }
5509 \msg_new:nnn { enumext } { command-wrong-place }
5510 {
5511   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5512   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5513 }
5514 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5515 {
5516   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5517   'anskey*' ~ and ~ is ~ being ~ ignored.
5518 }
5519 {
5520   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5521   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5522 }
5523 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5524 {
5525   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5526   'anskey*' ~ and ~ is ~ being ~ ignored.
5527 }
5528 {
5529   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5530   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5531 }
5532 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5533 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5534 {
5535   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5536   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5537 }
```

```

5538 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5539 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.
5540 {
5541   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5542   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5543 }

```

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

```

5544 \msg_new:nnn { enumext } { keyans-nested }
5545 {
5546   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5547 }
5548 \msg_new:nnn { enumext } { keyans-wrong-level }
5549 {
5550   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5551   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5552 }
5553 \msg_new:nnn { enumext } { wrong-place }
5554 {
5555   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5556   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5557 }
5558 \msg_new:nnn { enumext } { keyanspic-nested }
5559 {
5560   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5561 }
5562 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5563 {
5564   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5565   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5566 }
5567 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5568 {
5569   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5570 }
5571 \msg_new:nnnn { enumext } { keyans-unknown-key }
5572 {
5573   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5574   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5575 }
5576 {
5577   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5578   ~ have ~ a ~ key ~ called ~ '#1'.\\
5579   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5580 }
5581 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5582 {
5583   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5584   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5585 }
5586 {
5587   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5588   ~ have ~ a ~ key ~ called ~ '#1'.\\
5589   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5590 }

```

Message used by unknown `<keys>` in `enumext*` environment.

```

5591 \msg_new:nnnn { enumext } { starred-unknown-key }
5592 {
5593   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5594   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5595 }
5596 {
5597   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5598   ~ have ~ a ~ key ~ called ~ '#1'.\\
5599   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5600 }
5601 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5602 {
5603   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5604   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5605 }

```

```

5606 {
5607     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5608     ~ have ~ a ~ key ~ called ~ '#1'.\\
5609     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5610 }

```

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

```

5611 \msg_new:nnnn { enumext } { standar-unknown-key }
5612 {
5613     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5614     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5615 }
5616 {
5617     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5618     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5619     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5620 }
5621 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5622 {
5623     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5624     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5625 }
5626 {
5627     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5628     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5629     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5630 }

```

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

```

5631 \msg_new:nnnn { enumext } { for-key-unknown }
5632 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5633 {
5634     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5635     Check~that~you~have~spelled~the~key~name~correctly.
5636 }
5637 \msg_new:nnnn { enumext } { for-key-value-unknown }
5638 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5639 {
5640     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5641     Check~that~you~have~spelled~the~key~name~correctly.
5642 }

```

Messages used by \getkeyans command.

```

5643 \msg_new:nnn { enumext } { undefined-storage-anskey }
5644 {
5645     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5646 }

```

Messages used by \miniright command.

```

5647 \msg_new:nnn { enumext } { missing-miniright }
5648 {
5649     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5650     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5651 }
5652 \msg_new:nnn { enumext } { wrong-miniright-place }
5653 {
5654     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5655     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5656 }
5657 \msg_new:nnn { enumext } { wrong-miniright-use }
5658 {
5659     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5660     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5661 }
5662 \msg_new:nnn { enumext } { wrong-miniright-starred }
5663 {
5664     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5665 }
5666 \msg_new:nnn { enumext } { many-miniright-used }
5667 {
5668     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5669 }

```

Messages used by `\setenumextmeta` command.

```

5670 \msg_new:nnn { enumext } { unknown-set }
5671 {
5672   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5673 }
5674 \msg_new:nnn { enumext } { already-defined }
5675 {
5676   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5677 }
5678 \msg_new:nnn { enumext } { prohibited-unknown }
5679 {
5680   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5681 }

```

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

```

5682 \msg_new:nnn { enumext } { nested }
5683 {
5684   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5685 }
5686 \msg_new:nnn { enumext } { nested-horizontal }
5687 {
5688   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5689 }
5690 \msg_new:nnn { enumext } { item-joined }
5691 {
5692   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5693 }
5694 \msg_new:nnn { enumext } { item-joined-columns }
5695 {
5696   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5697 }

```

12.51 Finish package

Finish package implementation.

```

5698 \file_input_stop:
5699 </package>

```

13 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, 2775, 4095, 5316, 5325, 5330, 5350, 5352, 5359, 5361, 5374, 5379, 5384, 5399, 5438, 5440, 5442, 5447, 5448, 5453, 5454, 5472, 5489, 5506, 5511, 5520, 5529, 5535, 5541, 5550, 5555, 5564, 5578, 5588, 5598, 5608, 5618, 5628, 5634, 5640, 5649, 5654, 5659	
A	
above	<u>1589</u>
above*	<u>1589</u>
<code>\addvspace</code> 1159, 1187, 1230, 1233, 1401, 1404, 1501, 1507, 1542, 1548, 1569, 1575, 3584, 3745, 3763, 3996, 3999, 4329, 4344, 4390, 4404	
after	<u>989</u>
align	<u>536</u>
<code>\Alph</code>	38, <u>43</u>
<code>\Alph</code>	488, 606, 651, 719, 5024
<code>\alph</code>	38, <u>43</u>
<code>\alph</code>	489, 604, 5016
<code>\anskey</code>	13, 76, 78, <u>2593</u>
anskey*	14, <u>2703</u>
<code>\anspic</code>	16, 105, 107, <u>4003</u>
<code>\anspic*</code>	70
<code>\arabic</code>	32, 38
<code>\arabic</code>	487, 603, 650, 5008, 5012, 5028
B	
base-fix	<u>849</u>
<code>\baselineskip</code>	52
<code>\baselineskip</code>	865, 872
before	<u>989</u>
before*	<u>989</u>
below	<u>1589</u>
below*	<u>1589</u>
bool commands:	
<code>\bool_gset_false:N</code> 357, 358, 359, 2879, 2881, 4346, 4350, 4406	
<code>\bool_gset_true:N</code> 265, 275, 1092, 2082, 2088, 4315, 4347, 4379, 4407	
<code>\bool_if:NTF</code> . 427, 439, 456, 1523, 1611, 1625, 1638, 1649, 1660, 1671, 1682, 1693, 1742, 1759, 1764, 1772, 1799, 1837, 1842, 1849, 1853, 1875, 1880, 1888, 1895, 1926, 1934, 2027, 2225, 2235, 2314, 2338, 2345, 2369, 2467, 2489, 2529, 2543, 2547, 2597, 2616, 2640, 2694, 2705, 2794, 2831, 2895, 2930, 2945, 3020, 3031, 3035, 3054, 3067, 3109, 3143, 3186, 3205, 3348, 3363, 3425, 3435, 3468, 3473, 3539, 3565, 3615, 3673, 3728, 3753, 3928, 3994, 4005, 4024, 4069, 4308, 4324, 4330, 4373, 4387, 4391, 4513, 4523, 4607, 4613, 4620, 4636, 4655, 4678, 4739, 4749, 4856, 4860, 4886, 4893, 4912, 4922, 4943	
<code>\bool_if:nTF</code> 1549, 1576, 3165, 3322, 3383, 3907, 4045, 5047, 5191	
<code>\bool_if_p:N</code> 284, 299, 859, 860, 868, 869, 1906, 1907, 1915, 1916, 2040, 2066, 2079, 2080, 2085, 2086, 2402, 2412, 2424, 2439, 2440, 2474, 2515, 2516, 2817, 3007, 3008, 3045, 3046, 3512, 3514, 3525, 4052, 4053	
<code>\bool_lazy_all:nTF</code> 282, 297, 857, 2038, 2064, 2400, 2409, 2422, 2437, 3510, 3523	
<code>\bool_lazy_and:nnTF</code> 261, 271, 867, 1516, 1905, 1914, 2078, 2084, 2473, 2480, 2514, 2658, 2670, 2816, 2822, 3006	
<code>\bool_lazy_or:nnTF</code> . . 1967, 1974, 3044, 4051, 5214	
<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, 861, 2411, 2475, 2481, 2818, 2823, 3513, 3526	
<code>\bool_set_eq:NN</code> 3118, 3301, 4560, 4809	
<code>\bool_set_false:N</code> 436, 879, 2012, 2013, 2045, 2050, 2054, 2058, 2071, 2758, 3487, 3632, 3681, 3768, 3925, 4001, 4482, 4508, 4557, 4755, 4806, 5060, 5061	
<code>\bool_set_true:N</code> . 289, 290, 304, 305, 416, 420, 529, 894, 1595, 1600, 1862, 1984, 1985, 2257, 2265, 2759, 3112, 3114, 3146, 3148, 3297, 3309, 3448, 3486, 3519, 3532, 3605, 3678, 3705, 3909, 4297, 4362, 4481, 4564, 4571, 4572, 4616, 4753, 4813, 4820, 4821, 5055, 5056	
box commands:	
<code>\box_dp:N</code> . . 1447, 1448, 1451, 1458, 1471, 1479, 1485, 1493, 3938, 3943, 3996, 4080	
<code>\box_ht:N</code> . . 1230, 1233, 1244, 1245, 1256, 1258, 1273, 1276, 1284, 1285, 1296, 1298, 1313, 1316, 1323, 1324, 1335, 1337, 1352, 1355, 1401, 1404, 1412, 1413, 1421, 1422, 1434, 1436	
<code>\box_ht_plus_dp:N</code> 3934, 4033	
<code>\box_new:N</code> 70, 149, 150, 183, 189	
<code>\box_use_drop:N</code> 4341, 4402, 4675, 4940	
<code>\box_wd:N</code> 495	
C	
<code>\c</code>	228, 229, 756, 758, 770, 772
<code>\catcode</code>	2775
<code>\cB</code>	229
<code>\cE</code>	229
<code>\centering</code>	1551, 1578, 4117, 4334, 4395
check-ans	<u>2004</u>
Document class:	
article	45
clist commands:	
<code>\clist_const:Nn</code>	195
<code>\clist_map_function:nN</code>	4100
<code>\clist_map_inline:Nn</code> . . 535, 804, 988, 1003, 1084, 1605	
<code>\clist_map_inline:nn</code> . 49, 60, 78, 86, 99, 111, 141, 170, 194, 566, 586, 899, 920, 1098, 1711, 1951, 2018, 2204, 2222, 2254, 2397, 2939, 3226, 3238, 3278, 3412, 3415, 3443, 3455, 3458, 3478, 5167	
<code>\columnbreak</code>	77
<code>\columnbreak</code>	2477
columns	<u>1068</u>
columns-sep	<u>1068</u>
<code>\columnsep</code>	98
<code>\columnsep</code>	3560, 3726
<code>\columnseprule</code>	98
<code>\columnseprule</code>	3563, 3727

Commands provide by **enumext**:

`\anskey` 30, 67, 68, 72–76, 78, 79, 85, 87, 97, 98, 117, 127, 128, 136
`\anspic*` 30, 31, 70, 73, 86, 107, 127, 128
`\anspic` 30, 74, 105, 107, 136
`\foreachkeyans` 132, 138
`\getkeyans` 73, 127, 138
`\item*` 30, 31, 70, 73, 74, 86, 89, 92, 118, 119, 124, 127, 128
`\item` 89, 92, 93, 111, 117, 118, 120, 123, 124
`\miniright` 29, 49, 57, 58, 99, 100, 138
`\printkeyans*` 127
`\printkeyans` 30, 74, 127, 128
`\setenumextmeta` 131, 139
`\setenumext` 30, 128–130, 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, 762, 778, 2306, 2311, 2387, 2711, 3402, 4102, 5226
`\cs_if_exist:NTF` 467
`\cs_if_free:NTF` 2662, 2674
`\cs_new:Nn` 214
`\cs_new:Npn` 232, 1712, 1721, 1729, 2269, 2278, 2286, 5075, 5084, 5093
`\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, 627, 690, 742, 855, 1004, 1008, 1012, 1016, 1020, 1024, 1028, 1032, 1036, 1040, 1044, 1048, 1052, 1056, 1060, 1064, 1099, 1111, 1144, 1161, 1172, 1189, 1215, 1236, 1361, 1387, 1407, 1440, 1462, 1497, 1503, 1606, 1620, 1634, 1645, 1656, 1667, 1678, 1689, 1770, 1873, 1886, 1903, 1924, 1952, 1957, 1982, 2023, 2033, 2076, 2091, 2098, 2107, 2112, 2117, 2122, 2131, 2136, 2141, 2312, 2336, 2343, 2367, 2374, 2388, 2614, 2633, 2649, 2712, 2748, 2779, 2814, 2856, 2877, 2885, 2928, 2943, 2971, 3004, 3040, 3052, 3065, 3151, 3161, 3172, 3180, 3196, 3318, 3334, 3342, 3356, 3479, 3508, 3537, 3544, 3574, 3591, 3613, 3635, 3671, 3695, 3712, 3737, 3751, 3772, 3923, 4090, 4098, 4103, 4127, 4158, 4287, 4306, 4352, 4371, 4411, 4415, 4434, 4468, 4495, 4502, 4511, 4521, 4542, 4693, 4737, 4768, 4774, 4791, 4848, 4958
`\cs_new_protected:Npn` 202, 206, 210, 449, 465, 482, 492, 498, 607, 652, 724, 749, 763, 1533, 1562, 1738, 1757, 1827, 1860, 1962, 2146, 2223, 2233, 2255, 2263, 2298, 2307, 2463, 2526, 2541, 2579, 2583, 2703, 2734, 2738, 2769, 2905, 2981, 3025, 3105, 3124, 3239, 3243, 3257, 3261, 3279, 3283, 3293, 3305, 3371, 3405, 3446, 3490, 3691, 3899, 3916, 4022, 4041, 4065, 4189, 4238, 4485, 4548, 4555, 4569, 4577, 4582, 4592, 4761, 4797, 4804, 4818, 4826, 4843, 4980, 4993, 5041, 5164, 5176, 5200, 5212, 5250, 5260, 5268, 5290
`\cs_new_protected_nopar:Nn` 3833, 3875, 3883, 3891, 4531, 4535, 4669, 4780, 4784, 4934
`\cs_new_protected_nopar:Npn` 3825, 3841, 4598, 4644, 4876, 4901
`\cs_set:Npn` 2398, 2435, 4986
`\cs_set_eq:NN` 4458, 4459, 4646, 4726, 4727, 4903

`\cs_set_protected:Nn` 925, 941, 954, 967
`\cs_set_protected:Npn` 45, 54, 71, 79, 94, 100, 134, 166, 174, 514, 536, 571, 587, 634, 779, 805, 881, 904, 980, 989, 1068, 1085, 1589, 1700, 1943, 2004, 2163, 2205, 2241, 2390, 2932, 3215, 3231, 3271, 3403, 3444
`\cs_to_str:N` 484, 507
`\cs_undefine:N` 2651, 2652, 2653, 2654

D

`\d` 220
`\DeclareDocumentEnvironment` 397
dim commands:
`\dim_abs:n` 3376, 3381
`\dim_add:Nn` 3942, 4152, 4183
`\dim_compare:nNnTF` 927, 943, 956, 969, 1248, 1260, 1288, 1300, 1327, 1339, 1416, 1424, 1535, 1564, 3373, 3378, 3384, 3390, 3392, 3394, 3549, 3596, 3699, 3716, 3918, 4129, 4145, 4160, 4176, 4289, 4354
`\dim_compare:nTF` 2499, 2844, 3638, 3775
`\dim_eval:n` 865, 4076
`\dim_gset_eq:NN` 4298, 4363
`\dim_gzero:N` 2883, 4349, 4409
`\dim_new:N` 67, 74, 75, 76, 96, 143, 151, 152, 182, 184, 185, 191
`\dim_set:Nn` 495, 895, 3141, 3376, 3381, 3383, 3386, 3387, 3391, 3393, 3396, 3397, 3399, 3552, 3599, 3637, 3701, 3718, 3774, 3932, 4031, 4105, 4131, 4138, 4162, 4169, 4224, 4273, 4291, 4356, 4594
`\dim_set_eq:NN` 594, 641, 712, 716, 3056, 3057, 3069, 3070, 3136, 3414, 3457, 3560, 3726, 4231, 4234, 4235, 4280, 4283, 4284, 4587, 4666, 4931
`\dim_sub:Nn` 3643, 3780, 4147, 4178
`\dim_use:N` 928, 936, 1536, 1546, 2377, 2380, 2385, 3156, 3158, 3201, 3550, 3554, 3555, 3557, 3597, 3602, 3603, 3609, 3640, 3645
`\dim_zero:N` 3449, 3563, 3727, 3944, 3945, 3946
`\dim_zero_new:N` 464
`\c_zero_dim` 930, 944, 957, 970, 1536, 1564, 2501, 2846, 3373, 3378, 3384, 3391, 3550, 3597, 3640, 3699, 3716, 3777, 3918, 4129, 4145, 4160, 4176, 4289, 4354
`\dimeval` 2170

E

`\end` 2340, 2371, 3581, 3742, 3986, 4119, 5049, 5059, 5067
end internal commands:
`\end__enumext__mini_page` 1544, 1571, 3624, 3762, 4313, 4377, 4403
`\endgroup` 2775
`\endlist` 385
`\endminipage` 391
enumext 5, 3649
enumext internal commands:
`\l__enumext__ref_the_count_tl` 40
`\l__enumext__resume_name_tl` 63
`__enumext_add_meta_key:nnn` 131, 5178, 5194, 5195, 5197, 5200
`__enumext_add_pre_parsep:` 50, 1109, 1111, 1111
`__enumext_after_args_exec:` 48, 1004, 1016, 3662
`__enumext_after_args_exec_v:` 1020, 1032, 3795
`__enumext_after_args_exec_vii:` 1036, 1060
`__enumext_after_args_exec_viii:` 1064
`__enumext_after_env:nn` 82, 83, 85, 101, 113, 121, 206, 206, 2789, 3667, 4322, 4385, 4709
`__enumext_after_hyperref:` 36, 409, 411, 411
`\l__enumext_after_list_args_v_tl` 1034
`\l__enumext_after_list_args_vii_tl` 1062, 4664

```

\l__enumext_after_list_args_viii_tl .. 1066,
    4921
\__enumext_after_list_vii: 113, 117, 4466, 4502,
    4502
\__enumext_after_list_viii: ... 122, 4735, 4774,
    4774
\__enumext_after_stop_list: 48, 100, 1004, 1012,
    3629
\__enumext_after_stop_list_v: 1020, 1028, 3769
\l__enumext_after_stop_list_v_tl ..... 1030
\__enumext_after_stop_list_vii: .. 117, 1036,
    1052, 4505
\l__enumext_after_stop_list_vii_tl ... 1054
\__enumext_after_stop_list_viii: . 1056, 4777
\l__enumext_after_stop_list_viii_tl ... 1058
\l__enumext_align_label_pos_v_str .... 3360
\l__enumext_align_label_pos_X_str ..... 79
\l__enumext_align_label_vii_str ..... 4633
\l__enumext_align_label_viii_str ..... 4890
\l__enumext_align_label_X_str ..... 174
\c__enumext_all_envs_clist .. 195, 535, 804, 988,
    1003, 1084, 1605
\c__enumext_all_families_seq .. 130, 5132, 5158
\l__enumext_anskey_env_bool 33, 81, 34, 290, 305,
    2705
\__enumext_anskey_env_clean_vars: . 84, 2810,
    2814, 2877
\__enumext_anskey_env_define_keys: 81, 2703,
    2712, 2783
\__enumext_anskey_env_exec: 83, 2708, 2779, 2779
\__enumext_anskey_env_make:n 67, 81, 1987, 2703,
    2703, 2711
\__enumext_anskey_env_reset_keys: 82, 83, 2748,
    2811
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2703
\__enumext_anskey_env_save_keys: .. 83, 2791,
    2814, 2814
\__enumext_anskey_env_store: .. 84, 2807, 2814,
    2856
\__enumext_anskey_env_unknown:n 82, 2731, 2734
\__enumext_anskey_env_unknown:nn . 2736, 2738
\l__enumext_anskey_level_int .. 28, 2635, 2636
\__enumext_anskey_safe_inner: . 80, 2608, 2614,
    2633
\__enumext_anskey_safe_inner:n ..... 79
\__enumext_anskey_safe_outer: . 79, 2595, 2614,
    2614
\__enumext_anskey_show_wrap_arg:n . 78, 2526,
    2526, 2545, 2560
\__enumext_anskey_show_wrap_left:n 78, 2471,
    2541, 2541
\__enumext_anskey_unknown:n 79, 2563, 2577, 2579
\__enumext_anskey_unknown:nn . 2563, 2581, 2583
\__enumext_anskey_wrapper:n ..... 2167, 2539
\l__enumext_anspic_above_int . 142, 4106, 4107,
    4109
\__enumext_anspic_args:nnn 107, 109, 4019, 4090,
    4090
\l__enumext_anspic_args_seq 107, 109, 142, 3979,
    4017, 4118
\l__enumext_anspic_below_int . 142, 4106, 4107,
    4110
\l__enumext_anspic_body_box ... 142, 4030, 4033
\__enumext_anspic_body_dim:n .. 108, 4022, 4022,
    4068
\l__enumext_anspic_body_htdp_dim .. 108, 142,
    4031, 4079
\__enumext_anspic_label:nn 108, 4041, 4041, 4071,
    4085
\l__enumext_anspic_label_box .. 142, 3931, 3934
\l__enumext_anspic_label_htdp_dim . 106, 142,
    3932, 3938, 4078
\__enumext_anspic_label_pos:nnn .. 108, 4065,
    4065, 4093
\l__enumext_anspic_mini_pos_str 105, 142, 3910,
    3913, 4116
\l__enumext_anspic_mini_width_dim 142, 4043,
    4105, 4116
\__enumext_anspic_print:n 109, 3979, 3981, 4098,
    4098, 4102
\__enumext_anspic_row:n .. 109, 4098, 4100, 4103
\__enumext_anspic_start_list_tag: 3849, 3875,
    4092
\__enumext_anspic_stop_list_tag: . 3849, 3891,
    4096
\__enumext_anspic_stop_start_list_tag: 3849,
    3883, 4094
\__enumext_at_begin_document:n .. 35, 202, 202,
    382, 388
\l__enumext_base_line_fix_bool . 851, 860, 879,
    5055, 5060
\__enumext_before_args_exec: . 48, 99, 116, 1004,
    1004, 3594
\__enumext_before_args_exec_v: 1020, 1020, 3698
\__enumext_before_args_exec_vii: . 1036, 1036,
    4499
\__enumext_before_args_exec_viii: 1040, 4771
\__enumext_before_env:nn 81, 206, 210, 2656, 2668,
    2680, 2781
\__enumext_before_keys_exec: .. 48, 1004, 1008,
    3659
\__enumext_before_keys_exec_v: 1020, 1024, 3792
\__enumext_before_keys_exec_vii ..... 1036
\__enumext_before_keys_exec_vii: . 1044, 4453
\__enumext_before_keys_exec_viii: 1048, 4721
\__enumext_before_list: ... 99, 3591, 3591, 3653
\__enumext_before_list_v: ... 3695, 3695, 3787
\__enumext_before_list_vii: ... 116, 4448, 4495,
    4495
\__enumext_before_list_viii: .. 122, 4717, 4768,
    4768
\l__enumext_before_no_starred_key_v_tl 1026
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1046
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1050
\l__enumext_before_starred_key_v_tl ... 1022
\l__enumext_before_starred_key_vii_tl . 1038
\l__enumext_before_starred_key_viii_tl 1042
\__enumext_calc_hspace:NNNNNNN 94, 3371, 3371,
    3402, 3407, 3450
\__enumext_check_ans_active: . 68, 99, 116, 2023,
    2023, 3595, 4498
\g__enumext_check_ans_item_tl ..... 87
\g__enumext_check_ans_key_bool 69, 70, 153, 357,
    2082, 2088, 2895
\l__enumext_check_ans_key_bool 69, 2008, 2013,
    2079, 2085

```


__enumext_check_ans_key_hook: . . 69, 100, 117, 2076, 2076, 3630, 4506
 __enumext_check_ans_level: 68, 2023, 2029, 2033
 __enumext_check_ans_log: 69, 70, 85, 2122, 2122, 2899
 __enumext_check_ans_log_msg_greater: 2122, 2128, 2141
 __enumext_check_ans_log_msg_less: 2122, 2126, 2131
 __enumext_check_ans_log_msg_same_ok: 2122, 2127, 2136
 __enumext_check_ans_msg_greater: 2098, 2104, 2117
 __enumext_check_ans_msg_less: 2098, 2102, 2107
 __enumext_check_ans_msg_same_ok: 2098, 2103, 2112
 __enumext_check_ans_show: . . 69, 84, 2098, 2098, 2897
 \l__enumext_check_answers_bool . 67, 68, 79, 89, 153, 1985, 2012, 2027, 2314, 2338, 2345, 2369, 2597, 2794, 3020, 3109, 3143, 4613
 __enumext_check_starred_cmd:n 34, 70, 87, 121, 2146, 2146, 3798, 3992, 4734
 \g__enumext_check_starred_cmd_int . . 93, 153, 2149, 2155, 2160, 3316, 4050, 4855
 \l__enumext_check_start_line_env_tl . 34, 153, 320, 328, 336, 2152, 2158, 2161
 \l__enumext_columns_sep_v_dim 3716, 3718, 3726
 \l__enumext_columns_sep_vii_dim . . 4129, 4131, 4140, 4152, 4228, 4690
 \l__enumext_columns_sep_viii_dim . 4160, 4162, 4171, 4183, 4277, 4955
 \l__enumext_columns_v_int 1381, 1399, 1567, 3714, 3722, 3734, 3739
 \l__enumext_columns_vii_int . . 4134, 4137, 4141, 4150, 4192, 4196, 4199, 4205, 4211, 4215, 4684, 4698
 \l__enumext_columns_viii_int . 4165, 4168, 4172, 4181, 4241, 4245, 4248, 4254, 4260, 4264, 4949, 4964
 \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, 732
 \l__enumext_counter_vi_tl 45, 479
 \l__enumext_counter_vii_tl 45, 480, 662
 \l__enumext_counter_viii_tl 45, 481, 679
 \l__enumext_current_widest_dim 28, 67, 509, 595, 642, 713, 717
 __enumext_def_meta_key:nnn . . . 131, 5178, 5206, 5212, 5226
 __enumext_default_item:n . . . 3105, 3105, 3169
 __enumext_define_counters:Nn 28, 465, 465, 474, 475, 476, 477, 478, 479, 480, 481
 __enumext_endminipage: . 36, 382, 391, 405, 4343, 4671, 4936
 \g__enumext_envir_name_tl 33, 34, 291, 306, 365, 1955, 1960, 1970, 2110, 2115, 2120, 2134, 2139, 2144
 \l__enumext_envir_name_tl . 33, 34, 34, 260, 270, 319, 327, 335, 5574, 5577, 5584, 5587, 5594, 5597, 5604, 5607, 5613, 5617, 5623, 5627, 5684, 5688
 __enumext_execute_after_env: 35, 66, 69, 70, 80, 84, 2885, 2885, 3669, 4711
 __enumext_fake_item_indent: . . 925, 925, 3434
 \l__enumext_fake_item_indent_v_dim 944, 949
 \l__enumext_fake_item_indent_v_tl 946, 3298, 3302, 3310
 __enumext_fake_item_indent_vii: . . 925, 954, 3467
 \l__enumext_fake_item_indent_vii_dim 957, 962
 \l__enumext_fake_item_indent_vii_tl 959, 4665
 __enumext_fake_item_indent_viii: . 925, 967, 3472
 \l__enumext_fake_item_indent_viii_dim . 970, 975, 4926
 \l__enumext_fake_item_indent_viii_tl . . 972, 4924, 4929
 \l__enumext_fake_item_indent_X_tl 100
 __enumext_fake_make_label_vii:n . 119, 4598, 4598, 4661
 __enumext_fake_make_label_viii:n 4876, 4876, 4918
 __enumext_filter_first_level:n . . 129, 5075, 5075, 5109, 5120
 __enumext_filter_first_level_key:n 129, 5075, 5080, 5084
 __enumext_filter_first_level_pair:nn . 129, 5075, 5081, 5093
 __enumext_filter_save_key:n . . 73, 2230, 2238, 2261, 2267, 2269, 2269, 5006, 5010, 5014, 5018, 5022, 5026
 __enumext_filter_save_key_key:n . . 73, 2269, 2274, 2278
 __enumext_filter_save_key_pair:nn 73, 2269, 2275, 2286
 __enumext_filter_series:n 61, 1712, 1712, 1750, 1762, 1767
 __enumext_filter_series_key:n 62, 1712, 1717, 1721
 __enumext_filter_series_pair:nn . . 62, 1712, 1718, 1729
 __enumext_first_item_tmp_vii: 115, 117, 4458, 4531, 4531
 __enumext_first_item_tmp_viii: 121, 123, 4726, 4780, 4780
 \g__enumext_footnote_arg_seq . 171, 4417, 4430, 4440
 \g__enumext_footnote_int . 171, 4424, 4427, 4429, 4431
 \g__enumext_footnote_int_seq . 171, 4418, 4431, 4436, 4439
 __enumext_footnotes_key_bool 36
 \l__enumext_footnotes_key_bool 31, 37, 120, 161, 420, 427, 436, 4655, 4678, 4912, 4943
 __enumext_footnotetext:nn . . . 4411, 4411, 4441
 __enumext_foreach_add_body:n . 132, 5227, 5287, 5290
 \l__enumext_foreach_after_tl 5231, 5299
 \l__enumext_foreach_before_tl 5229, 5294
 \g__enumext_foreach_default_keys_tl 132, 126, 5249, 5270
 __enumext_foreach_keyans:nn . . 132, 5227, 5266, 5268
 \l__enumext_foreach_name_prop_tl . 126, 5272, 5297
 \l__enumext_foreach_print_seq 126, 5282, 5288, 5292

<code>\l__enumext_foreach_sep_tl</code>	5241, 5288	<code>\l__enumext_item_text_X_box</code>	174
<code>\l__enumext_foreach_start_int</code>	5233, 5284	<code>\l__enumext_item_width_vii_dim</code> . . .	4138, 4147, 4226, 4234, 4235
<code>\l__enumext_foreach_step_int</code>	5237, 5285	<code>\l__enumext_item_width_viii_dim</code> . .	4169, 4178, 4275, 4283, 4284
<code>\l__enumext_foreach_stop_int</code> .	5235, 5277, 5279, 5286	<code>\l__enumext_item_width_X_dim</code>	174
<code>__enumext_foreach_wrapper:n</code>	5239, 5295	<code>\l__enumext_itemindent_X_dim</code>	71
<code>__enumext_getkeyans:nn</code> . .	127, 4989, 4993, 4993	<code>\l__enumext_itemsep_i_skip</code> . . .	1242, 1249, 1252, 1254, 1261, 1265, 1268, 1270, 1410, 1417, 1419, 1420, 1425, 1429, 1431, 1432
<code>__enumext_getkeyans_aux:n</code>	127, 4977, 4980, 4980	<code>\l__enumext_itemsep_ii_skip</code> . .	1282, 1289, 1292, 1294, 1301, 1305, 1308, 1310
<code>\l__enumext_hyperref_bool</code> .	31, 36, 37, 161, 416, 439, 456, 2516, 3008, 4607	<code>\l__enumext_itemsep_iii_skip</code> .	1321, 1328, 1331, 1333, 1340, 1344, 1347, 1349
<code>__enumext_hypertarget:nn</code>	37, 411, 441, 445, 461	<code>\l__enumext_itemsep_vii_skip</code>	4704
<code>__enumext_if_is_int:n</code>	218	<code>\l__enumext_itemsep_viii_skip</code>	4970
<code>__enumext_if_is_int:nTF</code>	218, 751, 765	<code>\l__enumext_joined_item_aux_vii_int</code> . .	4220, 4221, 4222, 4223, 4229
<code>__enumext_internal_mini_page:</code>	36, 97, 116, 393, 393, 3481, 4470	<code>\l__enumext_joined_item_aux_viii_int</code> .	4269, 4270, 4271, 4272, 4278
<code>__enumext_is_not_nested:</code>	28, 33, 97, 116, 254, 254, 3482, 4471	<code>\l__enumext_joined_item_aux_X_int</code>	174
<code>__enumext_is_on_first_level:</code> .	28, 33, 97, 116, 254, 280, 3488, 4483	<code>__enumext_joined_item_vii:w</code> . .	118, 4545, 4546, 4548, 4548
<code>\g__enumext_item_anskey_int</code>	79, 87, 153, 352, 379, 380, 2095, 2465, 3022	<code>\l__enumext_joined_item_vii_int</code> . .	4191, 4192, 4195, 4197, 4203, 4208, 4213, 4218, 4220, 4226
<code>__enumext_item_answer_diff:</code> . .	69, 70, 84, 2091, 2091, 2892	<code>__enumext_joined_item_viii:w</code> .	123, 4794, 4795, 4797, 4797
<code>\g__enumext_item_answer_diff_int</code> .	69, 70, 153, 353, 2093, 2100, 2124	<code>\l__enumext_joined_item_viii_int</code> .	4240, 4241, 4244, 4246, 4252, 4257, 4262, 4267, 4269, 4275
<code>\l__enumext_item_column_pos_vii_int</code>	117, 4199, 4205, 4211, 4215, 4222, 4538, 4684, 4687	<code>\l__enumext_joined_item_X_int</code>	174
<code>\l__enumext_item_column_pos_viii_int</code> . .	123, 4248, 4254, 4260, 4264, 4271, 4787, 4949, 4952	<code>\l__enumext_joined_width_vii_dim</code> .	4224, 4231, 4234, 4649, 4663
<code>\l__enumext_item_column_pos_X_int</code>	174	<code>\l__enumext_joined_width_viii_dim</code>	4273, 4280, 4283, 4906, 4920
<code>\g__enumext_item_count_all_vii_int</code>	117, 4223, 4539, 4698, 4706	<code>\l__enumext_joined_width_X_dim</code>	174
<code>\g__enumext_item_count_all_viii_int</code>	123, 4272, 4788, 4963, 4972	<code>__enumext_keyans_addto_prop:n</code>	85, 2905, 2905, 3313, 4047
<code>\g__enumext_item_count_all_X_int</code>	174	<code>__enumext_keyans_addto_seq:n</code> .	86, 2981, 2981, 3315, 4049
<code>\g__enumext_item_number_bool</code>	153	<code>__enumext_keyans_addto_seq_link:</code>	2981, 3002, 3004, 4854
<code>\l__enumext_item_number_bool</code>	68, 159, 2045, 2050, 2054, 2058, 2071, 2640, 2694, 3112, 3146, 4616	<code>__enumext_keyans_default_item:n</code> . .	93, 3293, 3293, 3330
<code>\g__enumext_item_number_int</code>	68, 69, 153, 351, 378, 380, 2044, 2049, 2053, 2057, 2070, 2095, 3111, 3145, 4615	<code>\l__enumext_keyans_env_bool</code>	34, 3513, 3526, 3678, 3768
<code>__enumext_item_peek_args_vii:</code>	117, 118, 4540, 4542, 4542	<code>__enumext_keyans_fake_item_indent:</code>	925, 941, 3424
<code>__enumext_item_peek_args_viii:</code> . .	123, 4789, 4791, 4791	<code>\l__enumext_keyans_level_h_int</code> . .	122, 28, 672, 699, 2624, 2686, 2959, 4477, 4743, 4744
<code>__enumext_item_star_exec:</code>	90, 3124, 3151, 3188, 3207	<code>\l__enumext_keyans_level_int</code> . .	28, 1527, 2620, 2682, 2954, 3677, 3682, 4013
<code>\l__enumext_item_starred_vii_bool</code>	4557, 4571, 4620	<code>__enumext_keyans_make_label:</code>	39, 93, 3334, 3334, 3422
<code>\l__enumext_item_starred_viii_bool</code>	4806, 4820, 4886, 4922	<code>__enumext_keyans_make_label_box:</code>	3334, 3338, 3356
<code>\l__enumext_item_starred_X_bool</code>	174	<code>__enumext_keyans_make_label_std:</code>	3334, 3340, 3342
<code>__enumext_item_std:w</code> .	36, 89, 93, 382, 386, 3115, 3121, 3149, 3298, 3302, 3310	<code>__enumext_keyans_mini_right_cmd:n</code>	58, 1529, 1562, 1562
<code>\g__enumext_item_symbol_aux_tl</code> .	89, 130, 3129, 3132, 3157, 3193, 3211	<code>__enumext_keyans_mini_set_vskip:</code>	55
<code>\g__enumext_item_symbol_aux_vii_tl</code>	4579, 4622, 4625, 4629, 4631	<code>__enumext_keyans_minipage_add_space:</code>	1361, 1387, 3707
<code>\g__enumext_item_symbol_aux_X_tl</code>	174	<code>__enumext_keyans_minipage_set_skip:</code> .	1361, 1361, 1389
<code>\l__enumext_item_symbol_sep_vii_dim</code> . .	4587, 4594, 4628, 4630		
<code>\l__enumext_item_symbol_vii_tl</code>	4625		
<code>\l__enumext_item_text_vii_box</code>	4647, 4675		
<code>\l__enumext_item_text_viii_box</code> . . .	4904, 4940		

```

\__enumext_keyans_multi_addvspace: 1161, 1172,
    3731
\__enumext_keyans_multi_set_vskip:   51, 1161,
    1161, 1174
\__enumext_keyans_multicols_start: 3695, 3710,
    3712
\__enumext_keyans_multicols_stop:   1566, 3695,
    3737, 3766
\__enumext_keyans_name_and_start:   28, 34, 122,
    313, 313, 3679, 3906, 4748
\__enumext_keyans_parse_keys:n      3691, 3691, 3786
\__enumext_keyans_pic_arg_two:      106, 3923, 3923,
    3953
\l__enumext_keyans_pic_level_int .. 28, 1511,
    2628, 2690, 2908, 2949, 2984, 3072, 3901, 3902
\g__enumext_keyans_pic_parsep_skip 142, 3940,
    3999
\__enumext_keyans_pic_safe_exec:n   105, 3899,
    3899, 3952
\__enumext_keyans_pic_skip_abs:N .. 106, 3916,
    3916, 3927
\l__enumext_keyans_pic_star_bool .. 105, 142,
    3909, 3928, 3994, 4024, 4069
\__enumext_keyans_pre_itemsep_skip: .. 1361,
    1380, 1407
\__enumext_keyans_redefine_item: .. 93, 3318,
    3318, 3421
\__enumext_keyans_ref: .. 43, 724, 742, 3423
\__enumext_keyans_ref:n .. 43, 721, 724, 724
\__enumext_keyans_safe_exec: . 3671, 3671, 3785
\__enumext_keyans_set_item_width:   102, 3772,
    3772, 3794
\__enumext_keyans_show_ans: .. 3025, 3033, 3052
\__enumext_keyans_show_item_opt: .. 93, 3025,
    3040, 3311, 4062, 4925
\__enumext_keyans_show_left:n . 93, 3025, 3025,
    3308, 4056
\__enumext_keyans_show_pos: .. 3025, 3037, 3065
\__enumext_keyans_starred_item:n .. 93, 3305,
    3305, 3326
\__enumext_keyans_store_ref: .. 86, 2928, 2928,
    3314, 4048, 4852
\__enumext_keyans_store_ref_aux_i:  86, 2928,
    2940, 2943
\__enumext_keyans_store_ref_aux_ii: 86, 2928,
    2969, 2971
\__enumext_keyans_unknown_keys:n . 3231, 3235,
    3239
\__enumext_keyans_unknown_keys:nn 3231, 3241,
    3243
\__enumext_keyans_wrapper_opt:n .. 2173, 3048
\l__enumext_label_copy_i_tl .. 2431, 2947, 2952,
    2957, 2962
\l__enumext_label_copy_v_tl .. 2957
\l__enumext_label_copy_vi_tl .. 2952
\l__enumext_label_copy_vii_tl 2407, 2418, 2447,
    2947
\l__enumext_label_copy_viii_tl .. 2962
\l__enumext_label_copy_X_tl .. 163
\l__enumext_label_fill_left_v_tl .. 3346
\l__enumext_label_fill_left_X_tl .. 100
\l__enumext_label_fill_right_v_tl .. 3353
\l__enumext_label_fill_right_X_tl .. 100
\l__enumext_label_font_style_v_tl 3347, 3362,
    4060
\l__enumext_label_font_style_vii_tl ... 4635
\l__enumext_label_font_style_viii_tl .. 4892
\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, 639, 710, 714
\l__enumext_label_v_tl 86, 707, 2913, 2989, 3059,
    3099, 3307, 3312, 3789, 3931, 4055, 4057
\l__enumext_label_vi_tl 86, 707, 2910, 2986, 4055,
    4057, 4061
\l__enumext_label_vii_tl . 634, 4566, 4589, 4596
\l__enumext_label_viii_tl 634, 4815, 4846, 4850
\l__enumext_label_width_by_box .. 67, 494, 495
\__enumext_label_width_by_box:Nn 38, 492, 492,
    497, 509, 775
\l__enumext_labelsep_i_dim ... 3057, 3062, 3070,
    3102, 4858, 4873
\l__enumext_labelsep_v_dim .. 3721
\l__enumext_labelsep_vii_dim . 2532, 3057, 3070,
    4133, 4143, 4227, 4533, 4587, 4642, 4651
\l__enumext_labelsep_viii_dim 4164, 4174, 4276,
    4782, 4899, 4908, 4926
\l__enumext_labelwidth_i_dim . 3056, 3062, 3069,
    3102, 4858, 4873
\l__enumext_labelwidth_v_dim .. 3360, 3721
\l__enumext_labelwidth_vii_dim ... 2532, 3056,
    3069, 4133, 4142, 4227, 4533, 4633, 4650
\l__enumext_labelwidth_viii_dim .. 4164, 4173,
    4276, 4782, 4890, 4907
\l__enumext_leftmargin_tmp_v_bool . 106, 3925
\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, 629, 631,
    928, 932, 936, 1006, 1010, 1014, 1018, 1101, 1103,
    1105, 1107, 1149, 1151, 1153, 1155, 1159, 1193, 1199,
    1204, 1206, 1209, 1212, 1225, 1228, 1536, 1540, 1546,
    1609, 1611, 1613, 1616, 1623, 1625, 1627, 1630, 2225,
    2227, 2229, 2257, 2258, 2260, 2316, 2324, 2328, 2332,
    2536, 2537, 3114, 3115, 3119, 3120, 3121, 3129, 3137,
    3138, 3141, 3148, 3149, 3153, 3156, 3158, 3184, 3185,
    3186, 3189, 3192, 3201, 3202, 3204, 3205, 3208, 3519,
    3532, 3539, 3547, 3550, 3552, 3554, 3555, 3556, 3557,
    3560, 3565, 3571, 3577, 3584, 3597, 3599, 3602, 3603,
    3605, 3609, 3615, 3640, 3645, 3656, 3658
\l__enumext_level_h_int 116, 28, 263, 286, 300, 655,
    692, 1518, 2041, 2061, 2426, 2660, 2672, 3527, 4472,
    4473
\l__enumext_level_int . 97, 28, 216, 273, 285, 301,
    395, 1113, 1238, 1517, 2035, 2067, 2403, 2413, 2419,
    2425, 2432, 2441, 2446, 2659, 2671, 2887, 3438, 3483,
    3484, 3495, 3503, 3517, 3530, 3561, 3686, 4009, 4515,
    4525, 4756, 5614, 5618, 5624, 5628
\__enumext_list_arg_two_i: .. 3403
\__enumext_list_arg_two_ii: .. 3403
\__enumext_list_arg_two_iii: .. 3403
\__enumext_list_arg_two_iv: .. 3403
\__enumext_list_arg_two_v: . 93, 3403, 3791, 3926
\__enumext_list_arg_two_vii: .. 3444, 4452
\__enumext_list_arg_two_viii: .. 3444, 4720
\l__enumext_listoffset_v_dim . 3723, 3777, 3780
\l__enumext_listparindent_vii_dim .. 4666

```

```

\l__enumext_listparindent_viii_dim ... 4931
\__enumext_log_answer_vars: . 35, 367, 375, 2894
\__enumext_log_global_vars: . 35, 367, 375, 2893
\__enumext_make_label: . 39, 90, 3172, 3172, 3432
\__enumext_make_label_box: ... 3172, 3176, 3196
\__enumext_make_label_std: ... 3172, 3178, 3180
\l__enumext_mark_answer_sym_tl 75, 2179, 2382,
2549, 3074, 3087, 4862
\l__enumext_mark_position_str 130, 2183, 2184,
2210, 2211, 2380
\l__enumext_mark_ref_sym_tl .. 2196, 2521, 3016
\l__enumext_meta_path_tl . 126, 5202, 5203, 5205,
5206
\c__enumext_meta_paths_prop ..... 131, 5178
\__enumext_mini_addvspace_vii: 57, 1497, 1497,
4301
\__enumext_mini_addvspace_viii: 57, 1497, 1503,
4366
__enumext_mini_env* ..... 393
\__enumext_mini_page 1546, 1573, 3609, 3708, 4303,
4368, 4389
\__enumext_mini_right_cmd:n 58, 1531, 1533, 1533
\__enumext_mini_set_vskip_vii: 56, 1440, 1440,
1499
\__enumext_mini_set_vskip_viii: 56, 1440, 1462,
1505
\__enumext_minipage:w 36, 382, 390, 399, 4326, 4663,
4920
\l__enumext_minipage_active_v_bool 3705, 3728,
3753
\g__enumext_minipage_active_vii_bool .. 113,
4315, 4324, 4346
\l__enumext_minipage_active_vii_bool . 4297,
4308
\g__enumext_minipage_active_viii_bool 4379,
4387, 4406
\l__enumext_minipage_active_viii_bool 4362,
4373
\g__enumext_minipage_active_X_bool ... 174
\l__enumext_minipage_active_X_bool .... 87
\__enumext_minipage_add_space: .. 53, 99, 1189,
1215, 3607
\g__enumext_minipage_after_skip 87, 1444, 1456,
4344, 4404
\l__enumext_minipage_after_skip .. 52, 100, 87,
1202, 1242, 1244, 1249, 1252, 1256, 1261, 1265, 1268,
1272, 1284, 1289, 1292, 1296, 1301, 1305, 1308, 1312,
1323, 1328, 1331, 1335, 1340, 1344, 1347, 1351, 1363,
1377, 1410, 1412, 1417, 1419, 1421, 1425, 1429, 1431,
1433, 1464, 1477, 1491, 1542, 1569, 3763
\g__enumext_minipage_center_vii_bool . 4330,
4347
\g__enumext_minipage_center_viii_bool 4391,
4407
\g__enumext_minipage_center_X_bool ... 174
\l__enumext_minipage_hsep_v_dim ..... 3703
\l__enumext_minipage_hsep_vii_dim .... 4295
\l__enumext_minipage_hsep_viii_dim ... 4360
\l__enumext_minipage_left_skip 87, 1364, 1442,
1447, 1451, 1465, 1469, 1483, 1501, 1507
\l__enumext_minipage_left_v_dim .. 3701, 3708
\l__enumext_minipage_left_vii_dim 4291, 4303
\l__enumext_minipage_left_viii_dim 4356, 4368
\l__enumext_minipage_left_X_dim ..... 87
\g__enumext_minipage_right_skip 87, 1443, 1448,
1452, 4329, 4390
\l__enumext_minipage_right_skip . 52, 87, 1191,
1197, 1202, 1204, 1206, 1365, 1366, 1372, 1377, 1378,
1379, 1384, 1466, 1473, 1487, 1548, 1575
\l__enumext_minipage_right_v_dim . 1564, 1573,
3699, 3703
\g__enumext_minipage_right_vii_dim 113, 4299,
4326, 4349
\l__enumext_minipage_right_vii_dim 113, 4289,
4294, 4300
\g__enumext_minipage_right_viii_dim .. 4364,
4389, 4409
\l__enumext_minipage_right_viii_dim .. 4354,
4359, 4365
\g__enumext_minipage_right_X_dim ..... 174
\g__enumext_minipage_right_X_skip .... 174
\__enumext_minipage_set_skip: . 52, 1189, 1189,
1217
\g__enumext_minipage_stat_int 99, 87, 1553, 1580,
3606, 3617, 3622, 3706, 3755, 3760
\l__enumext_minipage_temp_skip 87, 1263, 1273,
1276, 1303, 1313, 1316, 1342, 1352, 1355, 1427, 1434,
1436
\l__enumext_miniright_code_vii_box 4337, 4341
\g__enumext_miniright_code_vii_tl 113, 4332,
4339, 4348
\l__enumext_miniright_code_viii_box .. 4398,
4402
\g__enumext_miniright_code_viii_tl 4393, 4400,
4408
\l__enumext_miniright_code_X_box ..... 174
\__enumext_multi_addvspace: . 51, 99, 1144, 1144,
3568
\__enumext_multi_set_vskip: 50, 1099, 1099, 1146
\l__enumext_multicols_above_ii_skip ... 1118
\l__enumext_multicols_above_iii_skip .. 1127
\l__enumext_multicols_above_iv_skip ... 1136
\l__enumext_multicols_above_v_skip 1163, 1177,
1187, 1378
\l__enumext_multicols_above_X_skip .... 79
\l__enumext_multicols_below_ii_skip .. 1245,
1254, 1258, 1270, 1275
\l__enumext_multicols_below_iii_skip . 1285,
1294, 1298, 1310, 1315
\l__enumext_multicols_below_iv_skip .. 1324,
1333, 1337, 1349, 1354
\l__enumext_multicols_below_v_skip 1167, 1181,
1379, 1413, 1420, 1422, 1432, 1435, 3745
\l__enumext_multicols_below_X_skip .... 79
\g__enumext_multicols_right_X_skip .... 79
\__enumext_multicols_start: . 98, 99, 3544, 3544,
3611
\__enumext_multicols_stop: 99, 1538, 3574, 3574,
3627
\__enumext_nested_base_line_fix: . 45, 97, 855,
855, 3499
\__enumext_newlabel:nn 31, 37, 76, 449, 449, 2457,
2975
\l__enumext_newlabel_arg_one_tl 31, 37, 76, 86,
163, 2450, 2458, 2520, 2964, 2976, 3014
\l__enumext_newlabel_arg_two_tl 31, 37, 75, 163,
2406, 2416, 2429, 2444, 2459, 2951, 2956, 2961, 2977
\__enumext_parse_foreach_keys:n .. 5227, 5243,
5260

```


__enumext_parse_foreach_keys:nn . 5227, 5250, 5262
 __enumext_parse_keys:n 45, 62, 3490, 3490, 3652
 __enumext_parse_keys_vii:n 62, 4447, 4485, 4485
 __enumext_parse_keys_viii:n . 4716, 4761, 4761
 __enumext_parse_save_key:n 72, 2250, 2255, 2255
 __enumext_parse_save_key_vii:n 72, 2245, 2255, 2263
 __enumext_parse_series:n 62, 97, 116, 1738, 1738, 3498, 4491
 __enumext_parse_store_keys:n 97
 \l__enumext_parsep_i_skip 1116, 1120
 \l__enumext_parsep_ii_skip 1125, 1129
 \l__enumext_parsep_iii_skip 1134, 1138
 \l__enumext_parsep_vii_skip 4667
 \l__enumext_parsep_viii_skip 4932
 \l__enumext_partopsep_v_skip . 1179, 1183, 1374, 1397
 \l__enumext_partopsep_viii_skip 1475
 __enumext_phantomsection: 37, 411, 442, 446, 462
 __enumext_pre_itemsep_skip: 52, 53, 1207, 1236, 1236
 __enumext_print_footnote: . . . 4411, 4434, 4680, 4945
 __enumext_print_keyans_box:NN 75, 2374, 2374, 2387, 2531, 2535, 3061, 3101, 4858, 4873
 \l__enumext_print_keyans_i_tl 5011, 5033
 \l__enumext_print_keyans_ii_tl . . . 5015, 5034
 \l__enumext_print_keyans_iii_tl . . 5019, 5035
 \l__enumext_print_keyans_iv_tl . . . 5023, 5036
 \l__enumext_print_keyans_star_bool 130, 861, 869, 5056, 5061
 \l__enumext_print_keyans_starred_tl 127, 128, 130, 5007, 5054
 \l__enumext_print_keyans_vii_tl 127, 5027, 5037
 \l__enumext_print_keyans_X_tl 130
 __enumext_printkeyans:nnn 128, 5038, 5041, 5041
 __enumext_redefine_item: . 90, 3161, 3161, 3431
 \l__enumext_ref_key_arg_tl . 40, 41, 50, 229, 609, 610, 623, 654, 657, 668, 674, 685, 726, 727, 738
 \l__enumext_ref_the_count_tl . 41, 50, 616, 619, 622, 662, 664, 667, 679, 681, 684, 732, 734, 737
 __enumext_regex_counter_style: . . 32, 40, 224, 224, 617, 663, 680, 733
 __enumext_register_counter_style:Nn . . 482, 482, 487, 488, 489, 490, 491
 __enumext_remove_extra_parsep_vii: . . 4465, 4693, 4693
 __enumext_remove_extra_parsep_viii: . 4733, 4958, 4958
 __enumext_renew_footnote: . . . 4411, 4415, 4657, 4914
 \l__enumext_renew_the_count_v_tl 735, 744, 746
 \l__enumext_renew_the_count_vii_tl 665, 694, 696
 \l__enumext_renew_the_count_viii_tl 682, 701, 703
 \l__enumext_renew_the_count_X_tl 50
 __enumext_rescan_anskey_env:n . . 82, 84, 2769, 2864, 2872
 __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, 2902
 \l__enumext_resume_active_bool 62, 64, 61, 1742, 1862
 __enumext_resume_counter: . 64, 1860, 1866, 1873
 __enumext_resume_counter:n . 62, 64, 1831, 1836, 1860, 1860, 1930, 1938
 __enumext_resume_counter_save_ans: . . 64, 65, 1860, 1871, 1903
 __enumext_resume_counter_series: 64, 65, 1860, 1869, 1886
 \g__enumext_resume_int . . . 61, 1783, 1877, 1878
 __enumext_resume_last:n . . 62, 1738, 1744, 1757
 \l__enumext_resume_name_tl 61, 1779, 1787, 1790, 1806, 1814, 1817, 1863, 1864, 1892, 1899
 __enumext_resume_save_counter: . 63, 100, 117, 1770, 1770, 3633, 4509
 __enumext_resume_series:n . 64, 1706, 1827, 1827
 __enumext_resume_starred: . 65, 1707, 1924, 1924
 \g__enumext_resume_vii_int 61, 1810, 1882, 1883
 \l__enumext_rightmargin_vii_dim . . 4145, 4149, 4154
 \l__enumext_rightmargin_viii_dim . 4176, 4180, 4185
 __enumext_safe_exec: . . 36, 97, 3479, 3479, 3651
 __enumext_safe_exec_vii: . 36, 4446, 4468, 4468
 __enumext_safe_exec_viii: 122, 4715, 4737, 4737
 __enumext_second_part: . . 100, 3613, 3613, 3665
 __enumext_second_part_v: . . . 3695, 3751, 3799
 \l__enumext_series_name_tl 64
 \l__enumext_series_str . . 63, 97, 116, 1704, 1740, 1748, 1749, 1751, 1753, 1774, 1777, 1781, 1801, 1804, 1808, 3494, 4489
 __enumext_set_error:nn 5164, 5174, 5176
 __enumext_set_item_width: 100, 3635, 3635, 3661
 __enumext_set_parse:n 5148, 5164, 5164
 \l__enumext_setkey_tmpa_int . . . 121, 5141, 5145
 \l__enumext_setkey_tmpa_seq . . 121, 5139, 5149, 5155, 5157, 5159, 5171
 \l__enumext_setkey_tmpa_tl 121, 5147, 5151
 \l__enumext_setkey_tmpb_seq . . 121, 5140, 5143, 5147, 5148
 \l__enumext_setkey_tmpb_tl 121, 5166, 5168, 5169
 \l__enumext_show_answer_bool . 2190, 2214, 2543, 3031, 3045, 4052, 4856
 __enumext_show_length:nnn . . 47, 232, 232, 5385, 5386, 5387, 5388, 5389, 5390, 5391, 5392, 5393, 5394, 5400, 5401, 5402, 5403, 5404, 5405, 5406, 5407, 5408, 5409
 \l__enumext_show_position_bool . . . 2193, 2217, 2547, 3035, 3046, 4053, 4860
 \g__enumext_standar_bool 33, 97, 34, 262, 265, 284, 358, 1772, 1837, 1849, 1875, 1888, 1926, 2066, 2080, 2411, 2424, 2439, 3514
 \l__enumext_standar_bool 97, 100, 34, 2412, 3486, 3632, 4482
 \l__enumext_standar_first_bool 33, 97, 34, 289, 1759, 1906, 1968, 1975
 __enumext_standar_item_vii:w . 118, 4553, 4555, 4555
 __enumext_standar_item_viii:w 123, 124, 4802, 4804, 4804
 __enumext_standar_ref: 41, 607, 627, 3433
 __enumext_standar_ref:n 40, 599, 607, 607
 \g__enumext_standar_series_tl . 61, 1761, 1762,

```

    1928, 1931
\__enumext_standar_unknown_keys:n 3271, 3275,
    3279
\__enumext_standar_unknown_keys:nn 3271, 3281,
    3283
\g__enumext_starred_bool 33, 116, 34, 272, 275, 299,
    359, 1799, 1842, 1853, 1880, 1895, 1934, 2040, 2086,
    2402, 2945, 4350
\l__enumext_starred_bool 116, 117, 122, 34, 1523,
    2440, 2475, 2481, 2529, 2818, 2823, 3054, 3067, 3487,
    4481, 4508, 4749, 4753
\__enumext_starred_columns_set_vii: . 4127,
    4127, 4456
\__enumext_starred_columns_set_viii: . 4127,
    4158, 4724
\l__enumext_starred_first_bool 33, 116, 34, 304,
    859, 868, 1764, 1915, 1968, 1975
\__enumext_starred_item:nn . . . 3124, 3124, 3167
\__enumext_starred_item_exec: 124, 4848, 4848,
    4888
\__enumext_starred_item_vii:w . 118, 4552, 4569,
    4569
\__enumext_starred_item_vii_aux_i:w . 4569,
    4574, 4577
\__enumext_starred_item_vii_aux_ii:w . 4569,
    4575, 4580, 4582
\__enumext_starred_item_vii_aux_iii:w 4569,
    4585, 4592
\__enumext_starred_item_viii:w 123, 124, 4801,
    4818, 4818
\__enumext_starred_item_viii_aux_i:w . 124,
    4818, 4823, 4826
\__enumext_starred_item_viii_aux_ii:w . 124,
    4818, 4824, 4841, 4843
\__enumext_starred_joined_item_vii:n 111, 118,
    4189, 4189, 4550
\__enumext_starred_joined_item_viii:n . 111,
    123, 4189, 4238, 4799
\__enumext_starred_ref: . . . . 42, 652, 690, 3464
\__enumext_starred_ref:n . . . . 41, 646, 652, 652
\g__enumext_starred_series_tl . 61, 1766, 1767,
    1936, 1939
\__enumext_starred_unknown_keys:n 3253, 3255,
    3257
\__enumext_starred_unknown_keys:nn 3253, 3259,
    3261
\__enumext_start_from:NNn 43, 749, 749, 762, 784,
    790
\l__enumext_start_i_int . . . . 1878, 1890, 1909
\__enumext_start_item_tmp_vii: 115, 4459, 4535,
    4535
\__enumext_start_item_tmp_viii: . 121, 4727,
    4784, 4784
\__enumext_start_item_vii:w 118, 120, 4561, 4566,
    4589, 4596, 4644, 4644
\__enumext_start_item_viii:w . 124, 4810, 4815,
    4846, 4901, 4901
\g__enumext_start_line_tl 33, 34, 292, 307, 364,
    2110, 2115, 2120, 2134, 2139, 2144
\__enumext_start_list:nn . 36, 94, 382, 384, 3655,
    3788, 4450, 4718
\__enumext_start_list_tag:n . 3801, 3825, 4660,
    4917
\__enumext_start_mini_vii: 116, 4287, 4287, 4500
\__enumext_start_mini_viii: . . 122, 4352, 4352,
    4772
\__enumext_start_save_ans_msg: 66, 1952, 1952,
    1977
\__enumext_start_store_level: . 97, 3508, 3508,
    3654
\__enumext_start_store_level_vii: 117, 4449,
    4511, 4511
\l__enumext_start_vii_int . . . 1883, 1897, 1918
\l__enumext_start_X_int . . . . . 100
\__enumext_stop_item_tmp_vii: . 115, 117, 120,
    4458, 4464, 4537, 4646
\__enumext_stop_item_tmp_viii: 121, 123, 4726,
    4732, 4786, 4903
\__enumext_stop_item_vii: 120, 4646, 4669, 4669
\__enumext_stop_item_viii: 126, 4903, 4934, 4934
\__enumext_stop_list: 36, 113, 117, 382, 385, 3579,
    3587, 3741, 3748, 4310, 4318, 4375, 4382
\__enumext_stop_list_tag:n . . 3801, 3841, 4672,
    4937
\__enumext_stop_mini_vii: 113, 117, 4287, 4306,
    4504
\__enumext_stop_mini_viii: 122, 4352, 4371, 4776
\__enumext_stop_save_ans_msg: . 66, 1952, 1957,
    2891
\__enumext_stop_start_list_tag: . 3801, 3833,
    4662, 4919
\__enumext_stop_store_level: 98, 99, 3537, 3537,
    3580, 3588
\__enumext_stop_store_level_vii: . 113, 117,
    4311, 4319, 4511, 4521
\l__enumext_store_active_bool 30, 67, 112, 1907,
    1916, 1984, 2616, 3512, 3525, 3673, 3681, 4001, 4005,
    4513, 4523, 4739, 4755
\__enumext_store_active_keys:n . 72, 97, 2223,
    2223, 3505
\__enumext_store_active_keys_vii:n . 72, 116,
    2223, 2233, 4492
\__enumext_store_addto_prop:n 73, 85, 2298, 2298,
    2306, 2466, 2926, 4851
\__enumext_store_addto_seq:n 74, 87, 2307, 2307,
    2311, 2318, 2332, 2340, 2349, 2363, 2371, 2524, 3019
\l__enumext_store_anskey_arg_tl . 30, 77, 112,
    2472, 2477, 2479, 2484, 2491, 2494, 2504, 2509, 2512,
    2518, 2524
\__enumext_store_anskey_code:n 76, 79, 84, 2463,
    2463, 2609, 2862, 2870
\l__enumext_store_anskey_env_tl . 30, 83, 112,
    2792, 2796, 2802, 2864, 2872
\l__enumext_store_anskey_opt_tl 30, 83, 84, 112,
    2793, 2820, 2826, 2833, 2839, 2849, 2859, 2868
\__enumext_store_anskey_safe_outer: . . . . 79
\g__enumext_store_columns_break_bool . 2716,
    2817, 2879
\l__enumext_store_columns_break_bool . 2474,
    2565
\l__enumext_store_current_label_tl 30, 85, 87,
    124, 112, 2907, 2910, 2913, 2919, 2924, 2926, 2983,
    2986, 2989, 2995, 3000, 3010, 3019, 4828, 4833, 4837,
    4850, 4851, 4853
\l__enumext_store_current_label_tmp_tl . 30,
    112, 3307, 3312
\l__enumext_store_current_opt_arg_tl 30, 124,
    112, 3029, 3042, 3048, 4839
\__enumext_store_internal_ref: . 75, 76, 2388,

```

[2388](#), [2469](#)
[\g__enumext_store_item_join_int](#) .. [2719](#), [2824](#),
[2828](#), [2880](#)
[\l__enumext_store_item_join_int](#) .. [2482](#), [2486](#),
[2568](#)
[\g__enumext_store_item_star_bool](#) .. [2721](#), [2831](#),
[2881](#)
[\l__enumext_store_item_star_bool](#) .. [2489](#), [2570](#)
[\g__enumext_store_item_symbol_sep_dim](#) [2726](#),
[2846](#), [2851](#), [2883](#)
[\l__enumext_store_item_symbol_sep_dim](#) [2501](#),
[2506](#), [2575](#)
[\g__enumext_store_item_symbol_tl](#) .. [2724](#), [2837](#),
[2841](#), [2882](#)
[\l__enumext_store_item_symbol_tl](#) .. [2492](#), [2496](#),
[2573](#)
[\l__enumext_store_keyans_item_opt_sep_-](#)
[tl](#) [2176](#), [2917](#), [2921](#), [2993](#), [2997](#), [4831](#), [4835](#)
[__enumext_store_level_close:](#) .. [74](#), [2312](#), [2336](#),
[3541](#)
[__enumext_store_level_close_vii:](#) .. [74](#), [2343](#),
[2367](#), [4527](#)
[__enumext_store_level_open:](#) [74](#), [98](#), [2312](#), [2312](#),
[3520](#), [3533](#)
[__enumext_store_level_open_vii:](#) .. [74](#), [2343](#),
[2343](#), [4517](#)
[\g__enumext_store_name_tl](#) [30](#), [67](#), [112](#), [363](#), [370](#),
[371](#), [372](#), [373](#), [1960](#), [1986](#), [2109](#), [2114](#), [2119](#), [2133](#),
[2138](#), [2143](#), [2889](#)
[\l__enumext_store_name_tl](#) [30](#), [66](#), [68](#), [112](#), [1793](#),
[1796](#), [1820](#), [1823](#), [1911](#), [1920](#), [1955](#), [1964](#), [1965](#), [1986](#),
[1987](#), [1988](#), [1990](#), [1991](#), [1993](#), [1995](#), [1996](#), [1998](#), [2000](#),
[2001](#), [2025](#), [2300](#), [2302](#), [2309](#), [2452](#), [2453](#), [2555](#), [2798](#),
[2966](#), [2967](#), [3080](#), [3093](#), [4868](#)
[\l__enumext_store_ref_key_bool](#) [76](#), [2199](#), [2467](#),
[2515](#), [2930](#), [3007](#)
[\l__enumext_store_save_key_vii_bool](#) .. [2235](#),
[2265](#)
[\l__enumext_store_save_key_vii_tl](#) [2237](#), [2238](#),
[2266](#), [2267](#), [2347](#), [2355](#), [2359](#), [2363](#)
[\l__enumext_store_save_key_X_bool](#) .. [72](#), [130](#)
[\l__enumext_store_save_key_X_tl](#) [72](#), [130](#)
[\l__enumext_store_upper_level_X_bool](#) .. [130](#)
[__enumext_storing_exec:](#) .. [66](#), [67](#), [81](#), [1962](#), [1978](#),
[1982](#)
[__enumext_storing_set:n](#) .. [66](#), [1947](#), [1962](#), [1962](#)
[\l__enumext_the_counter_v_tl](#) [734](#)
[\l__enumext_the_counter_vii_tl](#) [664](#)
[\l__enumext_the_counter_viii_tl](#) [681](#)
[\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](#), [1700](#), [1711](#),
[1943](#), [1951](#), [2004](#), [2022](#), [2163](#), [2204](#), [2205](#), [2222](#), [2241](#),
[2254](#), [2390](#), [2397](#), [2398](#), [2419](#), [2432](#), [2435](#), [2446](#), [2932](#),
[2939](#), [3231](#), [3238](#), [3271](#), [3278](#), [3403](#), [3443](#), [3444](#), [3478](#)
[__enumext_tmp:nn](#) [514](#), [535](#), [536](#), [570](#), [571](#), [586](#), [779](#),
[804](#), [881](#), [903](#), [904](#), [924](#), [980](#), [988](#), [989](#), [1003](#), [1068](#),
[1084](#), [1085](#), [1098](#), [1589](#), [1605](#), [3215](#), [3230](#)
[__enumext_tmp:nnn](#) [587](#), [603](#), [604](#), [605](#), [606](#), [634](#), [650](#),
[651](#)
[__enumext_tmp:nnnnnn](#) [805](#), [830](#), [833](#), [836](#), [838](#), [840](#),
[843](#), [846](#)
[__enumext_tmp:w](#) [4986](#), [4989](#)
[\l__enumext_tmpa_vii_int](#) [4137](#), [4140](#), [4149](#), [4180](#)
[\l__enumext_tmpa_viii_int](#) [4168](#), [4171](#)
[\l__enumext_tmpa_X_dim](#) [174](#)
[\l__enumext_tmpa_X_int](#) [174](#)
[\l__enumext_topsep_v_skip](#) ... [1165](#), [1169](#), [1368](#)
[\l__enumext_topsep_vii_skip](#) .. [1445](#), [1454](#), [1458](#)
[\l__enumext_topsep_viii_skip](#) .. [1467](#), [1489](#), [1493](#)
[__enumext_undefine_anskey_env:](#) .. [80](#), [85](#), [2649](#),
[2649](#), [2900](#)
[__enumext_unskip_unkern:](#) .. [33](#), [238](#), [238](#), [1218](#),
[1390](#), [3582](#), [3583](#), [3623](#), [3743](#), [3744](#), [3761](#)
[\l__enumext_vspace_a_star_v_bool](#) [1638](#)
[\l__enumext_vspace_a_star_vii_bool](#) ... [1660](#)
[\l__enumext_vspace_a_star_viii_bool](#) ... [1671](#)
[\l__enumext_vspace_a_star_X_bool](#) [100](#)
[__enumext_vspace_above:](#) [59](#), [99](#), [1606](#), [1606](#), [3593](#)
[__enumext_vspace_above_v:](#) .. [60](#), [1634](#), [1634](#), [3697](#)
[\l__enumext_vspace_above_v_skip](#) .. [1636](#), [1640](#),
[1642](#)
[__enumext_vspace_above_vii:](#) [60](#), [116](#), [1656](#), [1656](#),
[4497](#)
[\l__enumext_vspace_above_vii_skip](#) [1658](#), [1662](#),
[1664](#)
[__enumext_vspace_above_viii:](#) .. [60](#), [1656](#), [1667](#),
[4770](#)
[\l__enumext_vspace_above_viii_skip](#) [1669](#), [1673](#),
[1675](#)
[\l__enumext_vspace_b_star_v_bool](#) [1649](#)
[\l__enumext_vspace_b_star_vii_bool](#) ... [1682](#)
[\l__enumext_vspace_b_star_viii_bool](#) ... [1693](#)
[\l__enumext_vspace_b_star_X_bool](#) [100](#)
[__enumext_vspace_below:](#) [60](#), [100](#), [1620](#), [1620](#), [3631](#)
[__enumext_vspace_below_v:](#) .. [60](#), [1645](#), [1645](#), [3770](#)
[\l__enumext_vspace_below_v_skip](#) .. [1647](#), [1651](#),
[1653](#)
[__enumext_vspace_below_vii:](#) [61](#), [117](#), [1678](#), [1678](#),
[4507](#)
[\l__enumext_vspace_below_vii_skip](#) [1680](#), [1684](#),
[1686](#)
[__enumext_vspace_below_viii:](#) .. [61](#), [1678](#), [1689](#),
[4778](#)
[\l__enumext_vspace_below_viii_skip](#) [1691](#), [1695](#),
[1697](#)
[__enumext_widest_from:nnnn](#) .. [43](#), [763](#), [763](#), [778](#),
[797](#)
[\g__enumext_widest_label_tl](#) [28](#), [38](#), [67](#), [502](#), [506](#),
[510](#)
[\l__enumext_wrap_label_opt_v_bool](#) [3301](#)
[\l__enumext_wrap_label_opt_vii_bool](#) [118](#), [4560](#)
[\l__enumext_wrap_label_opt_viii_bool](#) .. [124](#),
[4809](#)
[\l__enumext_wrap_label_opt_X_bool](#) [100](#)
[\l__enumext_wrap_label_v_bool](#) [3297](#), [3301](#), [3309](#),
[3348](#), [3363](#)
[\l__enumext_wrap_label_vii_bool](#) .. [118](#), [4560](#),
[4564](#), [4572](#), [4636](#)
[\l__enumext_wrap_label_viii_bool](#) .. [124](#), [4809](#),
[4813](#), [4821](#), [4893](#)
[\l__enumext_wrap_label_X_bool](#) [100](#)
[__enumext_wrapper_label_v:n](#) .. [3350](#), [3365](#), [4061](#)
[__enumext_wrapper_label_vii:n](#) [4638](#)
[__enumext_wrapper_label_viii:n](#) [4895](#)
[\l__enumext_write_aux_file_tl](#) .. [31](#), [76](#), [86](#), [163](#),
[2455](#), [2461](#), [2973](#), [2979](#)
[enumext*](#) [5](#), [4444](#)
[enumXi](#) [474](#)
[enumXii](#) [474](#)

enumXiii	474
enumXiv	474
enumXv	474
enumXvi	474
enumXvii	474
enumXviii	474

Environments provide by **enumext**:

anskey*	30, 67, 72, 75, 76, 78, 80, 81, 83, 85, 97, 98, 117, 128, 133, 136
enumext*	27, 28, 31–33, 36, 38, 41, 42, 44, 46–49, 56, 57, 60–66, 68, 69, 71–80, 83, 85, 86, 91, 92, 96–98, 103, 110, 111, 113–115, 117, 119–123, 125, 127–129, 131, 134, 137, 139
enumext	27, 28, 32, 33, 36, 38–41, 43–52, 55, 57–66, 68, 69, 71–80, 83, 85, 86, 89–92, 94, 95, 98, 100, 101, 105, 110, 113, 116, 117, 119, 122, 127, 129, 131, 134, 136, 138
keyans*	27, 28, 30–34, 38, 41–44, 46–49, 56, 57, 60, 61, 67, 68, 70, 71, 73, 81, 85, 91, 96, 103, 111, 112, 115, 122, 134, 137, 139
keyanspic	27, 28, 30, 31, 34, 38, 39, 42, 67, 68, 70, 73, 74, 81, 85–87, 103–109, 137
keyans	27, 28, 30, 31, 33, 34, 38, 39, 42, 44, 46–49, 51, 55, 57–60, 67, 68, 70, 71, 73, 74, 81, 85–87, 91–95, 101, 103, 105, 106, 108, 113, 123, 134, 137

Environments:

center	110
description	110
enumerate	110
flushleft	110
flushright	110
itemize	110
list	32, 35, 36, 78, 94, 99, 100, 103, 105, 106, 110, 113
lrbox	120
minipage	32, 35, 36, 49, 52, 53, 105, 109, 110, 113, 120, 126
multicols	50–53, 58, 98–100
quotation	110
quote	110
scontents	81, 83
tabbing	110
trivlist	110
verbatim	110
verse	110

exp commands:

\exp_after:wN	4989
\exp_args:Ne	2861, 2869, 3502, 4977
\exp_args:NV	2581, 2736, 3241, 3259, 3281, 5262
\exp_not:N	58, 505, 622, 667, 684, 737, 934, 948, 949, 961, 962, 974, 975, 2520, 2552, 2553, 3012, 3077, 3078, 3090, 3091, 4865, 4866, 4986
\exp_not:n	294, 309, 322, 330, 338, 561, 581, 622, 623, 667, 668, 684, 685, 737, 738, 935, 1727, 1736, 2187, 2284, 2296, 2458, 2486, 2496, 2506, 2520, 2521, 2828, 2841, 2851, 2976, 3014, 3016, 5091, 5101, 5294, 5299

F

\fbox	2170
\fboxrule	2170
\fboxsep	2170
file commands:	
\file_input_stop:	5698
first	989
font	514
\footnote	115
\footnote	115, 4419
\footnotemark	4429

\footnotesize	2553, 3078, 3091, 4866
\footnotetext	4413
\foreachkeyans	17, 132, 5227

G

\getkeyans	17, 127, 4975
group commands:	
\group_begin:	2551, 2596, 2771, 2858, 3076, 3089, 4864, 5032
\group_end:	2558, 2612, 2875, 3083, 3096, 4871, 5039

H

\hbadness	4674, 4939
hbox commands:	
\hbox_overlap_left:n	3157, 4629
\hbox_set:Nn	494, 3931
\hbox_set_end:	4673, 4938
\hbox_set_to_wd:Nnw	4647, 4904
\hfill	544, 549, 555, 556, 1545, 1572, 2520, 3012, 4314, 4378
hook commands:	
\hook_gput_code:nnn	9, 204, 208, 212, 409
\hook_gremove_code:nn	83, 2787
\hook_gset_rule:nnnn	410
\hook_if_empty:nTF	2785
\hyperlink	77, 87
\hyperlink	2520, 3012
\hypertarget	37
\hypertarget	441

I

\IfDocumentMetadataTF	3174, 3336, 3827, 3835, 3843, 3877, 3885, 3893, 3954, 3964, 3972, 3982, 3987, 4026, 4035, 4112, 4120, 4312, 4376, 4455, 4463, 4605, 4653, 4676, 4723, 4731, 4910, 4941
\IfHyperBoolean	417
\IfPackageLoadedTF	11, 19, 413, 429
\ignorespaces	937, 950, 963, 976, 4460, 4642, 4728, 4899
\inputlineno	294, 309, 322, 330, 338
int commands:	
\int_add:Nn	4222, 4271
\int_case:nn	1113, 1238, 2035, 2061, 2100, 2124
\int_case:nnTF	240
\int_compare:nNnTF	395, 655, 672, 692, 699, 1208, 1227, 1381, 1399, 1511, 1527, 1539, 1567, 2148, 2154, 2620, 2624, 2628, 2636, 2682, 2686, 2690, 2887, 2908, 2949, 2954, 2959, 2984, 3072, 3484, 3495, 3517, 3530, 3546, 3561, 3576, 3617, 3682, 3686, 3714, 3739, 3755, 3902, 4009, 4013, 4192, 4202, 4218, 4241, 4251, 4267, 4473, 4477, 4515, 4525, 4683, 4695, 4744, 4756, 4948, 4960, 5145, 5277
\int_compare_p:nNn	263, 273, 285, 286, 300, 301, 1517, 1518, 2041, 2067, 2403, 2413, 2425, 2426, 2441, 2482, 2659, 2660, 2671, 2672, 2824, 3527
\int_decr:N	4221, 4270
\int_eval:n	380, 792, 2302, 2453, 2553, 2967, 3078, 3091, 3418, 3463, 4210, 4259, 4866
\int_from_alph:n	757, 771
\int_from_roman:n	759, 773
\int_gadd:Nn	4223, 4272
\int_gdecr:N	2044, 2049, 2053, 2057, 2070
\int_gincr:N	1877, 1882, 2465, 3022, 3111, 3145, 3316, 3606, 3706, 4050, 4539, 4615, 4788, 4855
\int_gset:Nn	2093, 4427
\int_gset_eq:NN	1776, 1783, 1789, 1795, 1803, 1810, 1816, 1822, 4424

`\int_gzero:N` . 351, 352, 353, 1553, 1580, 2160, 2880, 3622, 3760, 4706, 4972

`\int_if_exist:NTF` 1751, 1787, 1793, 1814, 1820, 1998

`\int_incr:N` 2635, 3483, 3677, 3901, 4472, 4538, 4743, 4787

`\int_mod:nn` 4697, 4962

`\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, 1753, 2001

`\int_set:Nn` 753, 757, 759, 1890, 1897, 1909, 1918, 2772, 4106, 4107, 4137, 4168, 4191, 4197, 4213, 4240, 4246, 4262, 4674, 4939, 5141, 5279

`\int_set_eq:NN` 1878, 1883, 4220, 4269

`\int_sign:n` 2095

`\int_step_function:nnN` 2419, 2432, 2446

`\int_step_function:nnnN` 5283

`\int_step_inline:nn` 5193

`\int_step_inline:nnn` 4108

`\int_to_roman:n` 216, 2399, 2436

`\int_use:N` 373, 378, 379, 1209, 1228, 1540, 1892, 1899, 1911, 1920, 3418, 3438, 3463, 3503, 3547, 3556, 3571, 3577, 4195, 4196, 4208, 4244, 4245, 4257, 5614, 5618, 5624, 5628

`\int_zero:N` 4687, 4952

`\item` . 89, 92, 117, 120, 123, 125, 386, 2320, 2326, 2351, 2357, 2479, 2986, 2989, 3163, 3320, 3958, 3960, 4457, 4459, 4725, 4727, 4853

`\item*` 5, 15, 70, 3318

`item-pos*` 3215

`item-sym*` 3215

`\itemindent` 95

`\itemindent` 94

`itemindent` 881

`\itemsep` 3948

`\itemwidth` . 464, 2170, 3637, 3643, 3774, 3780, 4231, 4235, 4280, 4284

K

`keyans` 15, 3783

`keyans*` 15, 4713

`keyanspic` 16, 3950

Keys for `\anskey` provide by `enumext`:

`break-col` 77, 78, 81–83

`item-join` 77, 78, 81–83

`item-pos*` 77, 78, 81, 82, 84

`item-star` 77, 78, 81, 82, 84

`item-sym*` 77, 78, 81, 82, 84

Keys for `anskey*` provide by `enumext`:

`break-col` 77, 78, 81–83

`item-join` 77, 78, 81–83

`item-pos*` 77, 78, 81, 82, 84

`item-star` 77, 78, 81, 82, 84

`item-sym*` 77, 78, 81, 82, 84

Keys for environments provide by `enumext`:

`above*` 29, 59, 60, 99, 116

`above` 29, 59, 60, 99, 116, 122

`after` 48, 100, 117, 122

`align` 29, 39, 40, 90, 93, 119, 133

`base-fix` 45, 62, 73, 97, 128

`before*` 48, 99, 116, 122

`before` 48

`below*` 29, 59–61, 100, 117

`below` 29, 59–61, 100, 117, 122

`check-ans` . 31–33, 66–70, 73, 84, 87, 100, 101, 117, 121, 135

`columns-sep` 49, 98

`columns` 29, 49, 59, 98

`first` 48, 120

`font` 39, 90, 93, 108, 119

`item-pos*` 89, 91

`item-sym*` 30, 89, 91

`itemindent` 29, 46, 89, 93, 120

`itemsep` 44, 96

`labelsep` 39, 95, 119

`labelwidth` 38–43, 95, 119

`label` 28, 38, 40, 43, 106, 110

`lisparindent` 96

`list-indent` 29, 46, 106

`list-offset` 46, 100, 102

`listparindent` 46, 120

`mark-ans` 71, 73, 78

`mark-pos` 71, 133

`mark-ref` 71, 73, 75, 77

`mini-env` 29, 36, 49, 58, 59, 73, 99, 110, 113, 114, 116, 117, 122

`mini-right*` 29, 32, 49, 73, 113, 114, 116, 117

`mini-right` 29, 32, 49, 57, 73, 113, 114, 116, 117

`mini-sep` 29, 49, 73, 99

`no-store` 31, 66–68, 73, 79, 89

`noitemsep` 44

`nosep` 44

`parindent` 96

`parsep` 44, 96, 120

`partopsep` 44

`ref` 28, 32, 40–42, 135

`resume*` 28, 61, 62, 65–67, 73, 100, 117, 129

`resume` 28, 35, 61–67, 73, 100, 117, 129

`rightmargin` 46, 110

`save-ans` 30, 35, 62–66, 68, 69, 72–74, 79–81, 84, 85, 87, 92, 101, 107, 122–124, 127, 129, 135

`save-key` 30, 62, 72, 97, 116

`save-pos` 73

`save-ref` 31, 37, 71, 73, 75–77, 86, 87, 93, 124

`save-sep` 71, 73, 85, 124

`series` 28, 61–65, 73, 97, 100, 116, 117, 129

`show-ans` 71, 73, 75, 76, 78, 93, 108, 124

`show-length` 33, 47, 134

`show-pos` 30, 71, 75, 76, 78, 87, 93, 108, 124

`start*` 29, 43, 44, 62

`start` 29, 32, 43, 44, 62

`store-key` 72

`topsep` 44

`widest` 28, 32, 43, 44

`wrap-ans` 37, 71, 73, 75, 78

`wrap-label*` 29, 39, 89, 90, 93, 118, 119, 124

`wrap-label` . . 29, 39, 89, 90, 93, 106, 108, 118, 119, 124

`wrap-opt` 71, 73, 93, 108

keys commands:

`\keys_define:nn` 516, 538, 573, 589, 636, 707, 781, 807, 849, 883, 906, 982, 991, 1070, 1087, 1591, 1702, 1945, 2006, 2165, 2207, 2243, 2248, 2563, 2714, 2750, 3217, 3233, 3253, 3273, 5003, 5103, 5219, 5227

`\keys_if_exist_p:nn` 5215, 5216

`\l_keys_key_str` 79, 82, 2581, 2736, 3241, 3259, 3281, 5262, 5370

`\keys_precompile:nnN` . . 128, 200, 200, 5005, 5009, 5013, 5017, 5021, 5025, 5245

`\keys_set:nn` . 530, 875, 1093, 1596, 1601, 1839, 1844, 1931, 1939, 2601, 3497, 3502, 3693, 4490, 4765, 5107, 5112, 5113, 5114, 5115, 5118, 5123, 5124, 5125, 5126,

5127, 5128, 5129, 5161, 5271	
\keys_set_known:nn	2868
keyval commands:	
\keyval_parse:NNn	1716, 2273, 5079
L	
label	587, 634, 707
Labels provide by enumext:	
\Alph*	38
\Roman*	38
\alph*	38
\arabic*	32, 38
\roman*	38
\labelsep	3942, 3946
labelsep	514
\labelwidth	38
\labelwidth	3942, 3944
labelwidth	514
\lastkern	249
\lastnodetype	240
\lastskip	244
\leftmargin	95
\leftmargin	94, 3942
legacy commands:	
\legacy_if:nTF	4600, 4603, 4878, 4881
\legacy_if_gset_false:n	400, 4327
\legacy_if_set_false:n	4602, 4880
\legacy_if_set_true:n	4565, 4588, 4595, 4609, 4814, 4845
\linewidth	99
\linewidth	3601, 3637, 3703, 3774, 4105, 4140, 4171, 4293, 4358
\list	384
list-indent	881
list-offset	881
\listparindent	3945
listparindent	881
M	
\makebox	110
\makebox	2378, 2380, 3200, 3360, 4043, 4633, 4890
\makelabel	89, 90, 93, 110
\makelabel	89, 92, 3182, 3198, 3344, 3358
\makesavenoteenv	435
mark-ans	2163
mark-pos	2163, 2205
mark-ref	2163
mini-env	1068
mini-sep	1068
\minipage	390
\miniright	11, 57, 1509, 1557, 1584, 3620, 3758
mode commands:	
\mode_if_math:TF	2644, 2698
\mode_if_vertical:TF	1147, 1175, 1195, 1219, 1370, 1391
\mode_leave_vertical:	864, 871, 934, 948, 961, 974, 2376, 3155, 4627
msg commands:	
\msg_error:nn	1559, 1586, 2605, 2638, 2642, 2696, 2804, 3684, 3688, 3904, 3962, 4011, 4475, 4746, 4758, 5130, 5189
\msg_error:nnn	612, 659, 676, 729, 1513, 1520, 1525, 1555, 1582, 1851, 1855, 1970, 2587, 2646, 2664, 2676, 2684, 2688, 2692, 2700, 2742, 3247, 3265, 3287, 4479, 4751, 4991, 5000, 5072, 5177, 5208, 5217, 5254, 5275

\msg_error:nnnn	2590, 2618, 2622, 2626, 2630, 2745, 3250, 3268, 3290, 3675, 4007, 4015, 4741, 5051, 5257
\msg_error:nnnnn	560, 580, 2186
\msg_fatal:nn	3485
\msg_fatal:nnn	468
\msg_info:nnn	13, 16, 21, 24, 415, 431
\msg_line_context:	5335, 5340, 5345, 5374, 5379, 5384, 5399, 5414, 5418, 5422, 5426, 5430, 5434, 5441, 5448, 5454, 5468, 5472, 5477, 5481, 5485, 5489, 5494, 5498, 5502, 5506, 5511, 5546, 5550, 5555, 5560, 5564, 5569, 5645, 5649, 5654, 5659, 5664, 5668, 5672, 5676, 5680, 5684, 5688, 5692, 5696
\msg_log:nnn	1990, 1995, 2000
\msg_log:nnnnn	377, 2133, 2138, 2143
\msg_log:nnnnnn	369
\msg_new:nnn	5302, 5306, 5310, 5314, 5319, 5332, 5337, 5342, 5347, 5356, 5364, 5368, 5372, 5377, 5382, 5397, 5412, 5416, 5420, 5424, 5428, 5432, 5436, 5445, 5451, 5457, 5461, 5465, 5470, 5475, 5479, 5483, 5487, 5492, 5496, 5500, 5504, 5509, 5544, 5548, 5553, 5558, 5562, 5567, 5643, 5647, 5652, 5657, 5662, 5666, 5670, 5674, 5678, 5682, 5686, 5690, 5694
\msg_new:nnnn	5323, 5514, 5523, 5532, 5538, 5571, 5581, 5591, 5601, 5611, 5621, 5631, 5637
\msg_term:nnnn	1954, 1959, 3427, 3437, 3469, 3474
\msg_term:nnnnn	2114
\msg_warning:nn	3619, 3757
\msg_warning:nnnn	2151, 2157, 3375, 3380, 4194, 4207, 4243, 4256
\msg_warning:nnnnn	2109, 2119
\multicolsep	98
\multicolsep	1212, 1384, 3567, 3730

N

\NeedsTeXFormat	3
\NewCommandCopy	386
\newcounter	471
\NewDocumentCommand	1509, 2593, 4003, 4975, 5030, 5137, 5186, 5264
\NewDocumentEnvironment	3649, 3783, 3950, 4444, 4713
\newenvsc	2707
\newlabel	37
\newlabel	453
no-store	2004
\noindent	3608, 4302, 4367, 4686, 4951
\nointerlineskip	1221, 1224, 1393, 1396, 1547, 1574, 4302, 4367
noitemsep	805
\nopagebreak	1158, 1186, 1221, 1224, 1393, 1396, 1500, 1506
\normalfont	2552, 3077, 3090, 4865
nosep	805

P

Packages:	
caption	113
enumext	27, 37, 40, 66, 95, 105, 133
enumitem	38
expl3	110
footnotehyper	37
hyperref	31, 32, 36, 37, 77, 87, 119, 133
ltxcmd	35
lua-visual-debug	52
multicol	27, 133
scontents	27, 80, 81
shortlst	110, 115, 120

`\par` .. 1158, 1186, 1224, 1396, 1500, 1506, 1542, 1547, 1569, 1574, 2528, 3584, 3745, 3763, 3996, 3999, 4125, 4329, 4344, 4390, 4404, 4686, 4951

para commands:

`\para_end:` 4703, 4969

`\parbox` 2170

`\parindent` 4666, 4931

`\parsep` 50, 106

`\parsep` 865, 3460, 3927, 3936, 3940

`parsep` 805

`\parskip` 4667, 4932

`\partopsep` 3461, 3761, 3947

`partopsep` 805

peek commands:

`\peek_meaning:N`TF 4544, 4558, 4573, 4584, 4793, 4807, 4822

`\peek_meaning_remove:N`TF 4551, 4800

`\peek_remove_spaces:n` 3324

`\phantomsection` 37

`\phantomsection` 442

prg commands:

`\prg_do_nothing:` 446

`\prg_new_protected_conditional:N`ppn ... 218

`\prg_replicate:nn` 235

`\prg_return_false:` 222

`\prg_return_true:` 221

`\printkeyans` 17, 127, 5030

prop commands:

`\prop_const_from_keyval:N`n 5178

`\prop_count:N` 371, 2302, 2453, 2555, 2967, 3080, 3093, 4868, 5280

`\prop_get:N`nNTF 5204

`\prop_gput_if_not_in:N`nn 2300

`\prop_if_exist:N`TF 1988, 4995, 5273

`\prop_item:N`n 4997, 5297

`\prop_new:N` 1991

`\ProvidesExplPackage` 4

R

`\raggedcolumns` 3570, 3733

`\raisebox` 4074

`\ref` 75, 85

`ref` 587, 634, 707

`\refstepcounter` 4612, 4883

regex commands:

`\regex_match:nn`TF .. 220, 756, 758, 770, 772, 2800

`\regex_replace_once:nn`N 228

`\renewcommand` 622, 667, 684, 737

`\RenewDocumentCommand` 1557, 1584, 3163, 3182, 3198, 3320, 3344, 3358, 3960, 4419

`\RequirePackage` 17, 25

`resume` 1700

`resume*` 1700

`rightmargin` 881

`\Roman` 38, 43

`\Roman` 490

`\roman` 38, 43

`\roman` 491, 605, 5020

S

`\s` 2801

`save-ans` 1943

`save-key` 2241

`save-ref` 2163

`save-sep` 2163

scan commands:

`\scan_stop:` 3958, 4457, 4725, 4986, 4989

scontents internal commands:

`\l__scontents_fname_out_tl` 2760

`__scontents_parse_environment_keys:n` . 2766

`__scontents_rescan_tokens:n` 2773

`\l__scontents_storing_bool` 2758

`\l__scontents_writing_bool` 2759

seq commands:

`\seq_clear:N` 5139, 5282

`\seq_const_from_clist:N`n 5132

`\seq_count:N` 372, 3979, 5143

`\seq_gclear:N` 4417, 4418

`\seq_gput_right:N`n 2309, 4430, 4431

`\seq_if_empty:N`TF 4436, 5045, 5157

`\seq_if_exist:N`TF 1993, 5043

`\seq_if_in:N`nNTF 5049

`\seq_item:N`n 2798, 4118

`\seq_map_function:NN` 5148

`\seq_map_inline:N`n 5058, 5066, 5158, 5159

`\seq_map_pairwise_function:NNN` 4438

`\seq_new:N` 124, 125, 127, 142, 172, 173, 1996

`\seq_pop_left:NN` 5147

`\seq_put_right:N`n 4017, 5155, 5171, 5292

`\seq_set_from_clist:N`n 5140

`\seq_set_map_e:NN`n 5149

`\seq_use:N`n 200, 201, 5288

series 1700

`\setcounter` 767, 771, 773, 3418, 3463, 3993

`\setenumext` 6, 129, 5137

`\setenumextmeta` 6, 131, 5178

`show-ans` 2163, 2205

`show-length` 980

`show-pos` 2205

skip commands:

`\skip_add:N`n 1118, 1127, 1136, 1149, 1153, 1177, 1181, 1197, 1255, 1257, 1271, 1274, 1295, 1297, 1311, 1314, 1334, 1336, 1350, 1353, 1372, 1421, 1422, 1433, 1435, 3936, 3943

`\skip_gset:N`n 1448, 1452, 1456

`\skip_gset_eq:NN` 3940

`\skip_gzero_new:N` 1443, 1444

`\skip_horizontal:N` 949, 962, 975, 4630, 4642, 4690, 4899, 4955

`\skip_horizontal:n` ... 935, 2377, 2385, 3156, 3158, 4533, 4628, 4782, 4926

`\skip_if_eq:nn`TF 1116, 1125, 1134, 1241, 1281, 1321, 1409, 1445, 1467, 1608, 1622, 1636, 1647, 1658, 1669, 1680, 1691

`\skip_new:N` 81, 82, 83, 88, 89, 90, 91, 92, 93, 148, 192

`\skip_set:N`n 1101, 1105, 1163, 1167, 1191, 1244, 1245, 1263, 1284, 1285, 1303, 1323, 1324, 1342, 1366, 1412, 1413, 1427, 1447, 1451, 1469, 1473, 1477, 1483, 1487, 1491, 3920

`\skip_set_eq:NN` 1202, 1203, 1205, 1212, 1377, 1378, 1379, 1384, 3416, 3459, 3460, 4667, 4932

`\skip_sub:N`n 1251, 1253, 1267, 1269, 1291, 1293, 1307, 1309, 1330, 1332, 1346, 1348, 1419, 1420, 1431, 1432

`\skip_use:N` 1103, 1107, 1151, 1155, 1159, 1179, 1183, 1193, 1199, 1609, 1613, 1616, 1623, 1627, 1630, 3584

`\skip_vertical:N` . 401, 404, 873, 4328, 4342, 4705, 4971

`\skip_vertical:n` 872, 4704, 4970

`\skip_zero:N` 1211, 1225, 1363, 1364, 1365, 1383, 1397, 3461, 3567, 3730, 3947, 3948

`\skip_zero_new:N` 1442, 1464, 1465, 1466
`\c_zero_skip` . 401, 404, 873, 1116, 1125, 1134, 1282,
 1321, 1445, 1467, 1609, 1623, 1636, 1647, 1658, 1669,
 1680, 1691, 4328, 4342, 4705, 4971
`\small` 5008, 5012, 5016, 5020, 5024, 5028
 socket commands:
`\socket_assign_plug:nn` .. 3829, 3837, 3845, 3879,
 3887, 3895
`\socket_new:nn` 3801, 3849
`\socket_new_plug:nnn` 3802, 3809, 3817, 3850, 3857,
 3866
`\socket_use:n` 3830, 3880
`\socket_use:nn` 3838, 3846, 3888, 3896
`\star` 3221
`start` 779
`start*` 779
`start-list-tags` 3801, 3849
`\stepcounter` 3930, 4067, 4423
`stop-list-tags` 3801, 3849
`stop-start-tags` 3801, 3849
 str commands:
`\c_backslash_str` 2646, 5335, 5340, 5345, 5350, 5352,
 5354, 5359, 5361, 5459, 5463, 5467, 5477, 5481, 5489,
 5490, 5494, 5506, 5507, 5511, 5512, 5533, 5535, 5539,
 5541, 5569, 5632, 5634, 5638, 5640, 5649, 5650, 5654,
 5659, 5660, 5664, 5668, 5672
`\c_colon_str` 2452, 2966, 4986
`\c_left_brace_str` 5440, 5447, 5453
`\c_right_brace_str` 5440, 5447, 5453
`\str_case:nn` 256, 315
`\str_case:nnTF` . 1723, 1731, 2280, 2288, 5086, 5095
`\str_clear:N` 3494, 4489
`\str_count:n` 235
`\str_if_empty:N` 1740, 1781, 1808
`\str_if_eq:nnTF` 3419, 3465, 5188
`\str_if_in:nnTF` 4982
`\str_new:N` 84, 132, 147, 187
`\str_set:Nn` . 545, 551, 557, 576, 577, 578, 2183, 2184,
 2210, 2211, 3910, 3913
`\str_use:N` 3202
`\string` 435
`\strutbox` . 1230, 1233, 1244, 1245, 1256, 1258, 1273, 1276,
 1284, 1285, 1296, 1298, 1313, 1316, 1323, 1324, 1335,
 1337, 1352, 1355, 1401, 1404, 1412, 1413, 1421, 1422,
 1434, 1436, 1447, 1448, 1451, 1458, 1471, 1479, 1485,
 1493, 3938, 3943, 3996, 4080

T

tag commands:

`\tag_mc_begin:n` 3807, 3855, 3864
`\tag_mc_end:` 3811, 3859, 3868
`\tag_resume:n` .. 3804, 3852, 3966, 3974, 4037, 4122,
 4312, 4376
`\tag_struct_begin:n` . 3805, 3806, 3813, 3814, 3815,
 3853, 3854, 3861, 3862, 3863, 3975
`\tag_struct_end:` 3989, 3990
`\tag_struct_end:n` 3812, 3819, 3820, 3821, 3822, 3860,
 3869, 3870, 3871, 3872, 4463, 4731
`\tag_suspend:n` . 3823, 3873, 3956, 3968, 3984, 4028,
 4114, 4455, 4723
`\tag_tool:n` 3967
 T_EX and L^AT_EX 2_ε commands:
`\@auxout` 451
`\@currenvir` 256, 315
`\protected@write` 451

tex commands:

`\tex_newlinechar:D` 2772

text commands:

`\text_expand:n` 4978
`\textasteriskcentered` 2180, 2197
`\the` 244, 249
`\thepage` 457

tl commands:

`\c_space_tl` 3048, 5384, 5399, 5422, 5426, 5613, 5614,
 5623, 5624, 5684, 5688
`\tl_clear:N` .. 543, 550, 2161, 2227, 2237, 2258, 2266,
 2472, 2792, 2793, 2907, 2983, 4828
`\tl_clear_new:N` 500
`\tl_const:Nn` 50, 484
`\tl_gclear:N` . 363, 364, 365, 1761, 1766, 2882, 3193,
 3211, 4348, 4408, 4631
`\tl_gclear_new:N` 1748
`\tl_gput_right:Nn` 485
`\tl_greplace_all:Nnn` 506
`\tl_gset:Nn` 291, 292, 306, 307, 1749, 1762, 1767, 1986,
 2796, 3132, 4579
`\tl_gset_eq:NN` 502, 3128, 4624
`\tl_if_blank:nTF` 2585, 2603, 2740, 3245, 3263, 3285,
 4622, 5252
`\tl_if_empty:N`TF . 610, 629, 657, 674, 694, 701, 727,
 744, 1774, 1779, 1801, 1806, 1864, 1928, 1936, 1965,
 2025, 2316, 2347, 2492, 2837, 2859, 2889, 2917, 2993,
 3042, 3153, 4831, 5169
`\tl_if_empty:nTF` 1829
`\tl_if_exist:N`TF 1834
`\tl_if_novalue:nTF` .. 2599, 2915, 2991, 3027, 3107,
 3126, 3134, 3295, 3492, 3977, 4421, 4487, 4763, 4829
`\tl_map_inline:Nn` 226, 503
`\tl_new:N` 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
`\tl_put_left::Ne` 2826
`\tl_put_left:Nn` 2324, 2355, 2477, 2820, 2833, 2839,
 2849, 3059, 3099, 4332, 4393, 4850, 4853
`\tl_put_right:Nn` 501, 620, 665, 682, 735, 2328, 2359,
 2406, 2416, 2429, 2444, 2450, 2455, 2479, 2484, 2491,
 2494, 2504, 2509, 2512, 2518, 2910, 2913, 2919, 2924,
 2951, 2956, 2961, 2964, 2973, 2986, 2989, 2995, 3000,
 3010, 4833, 4837
`\tl_remove_all:Nn` 5168
`\tl_remove_once:Nn` 2394, 2936
`\tl_replace_all:Nnn` 505, 5203
`\tl_reverse:N` 2393, 2395, 2935, 2937
`\tl_set:Nn` . 58, 260, 270, 319, 320, 327, 328, 335, 336,
 470, 544, 549, 555, 556, 609, 654, 726, 932, 946, 959,
 972, 1863, 1964, 2228, 2238, 2259, 2267, 2549, 2760,
 3029, 3074, 3087, 4839, 4862, 5166, 5202, 5272
`\tl_set_eq:NN` 511, 615, 618, 662, 664, 679, 681, 732,
 734, 2392, 2934, 2947, 3307, 3312, 4055, 4057
`\tl_to_str:n` 1834, 1840, 1845, 4978
`\tl_trim_spaces:n` ... 501, 5155, 5166, 5172, 5188
`\tl_use:N` 507, 510, 631, 696, 703, 746, 1006, 1010, 1014,
 1018, 1022, 1026, 1030, 1034, 1038, 1042, 1046, 1050,
 1054, 1058, 1062, 1066, 2382, 2399, 2407, 2418, 2431,
 2436, 2447, 3115, 3121, 3149, 3184, 3185, 3192, 3204,
 3298, 3302, 3310, 3346, 3347, 3353, 3362, 3656, 3789,
 4060, 4339, 4400, 4635, 4664, 4665, 4892, 4921, 4924,
 4929, 5033, 5034, 5035, 5036, 5037, 5054, 5151, 5270

token commands:

\token_to_str:N	453
\topsep	3761, 3943
topsep	805
\topskip	1211, 1383
\typeout	244, 249, 419, 423, 434, 435

U

\u	229, 2801
\unkern	250
unknown	3231, 3253, 3271
\unskip	245
use commands:	
\use:N	236, 3189, 3208, 3658
\use:n	1714, 2271, 4984, 5077
\use_none:nn	445, 5209
\usecounter	3417, 3462

V

\value	1777, 1783, 1790, 1796, 1804, 1810, 1817, 1823
vbox commands:	
\ vbox_set:Nn	4030
\ vbox_set_top:Nn	4337, 4398
\vspace	865, 1613, 1616, 1627, 1630, 1640, 1642, 1651, 1653, 1662, 1664, 1673, 1675, 1684, 1686, 1695, 1697

W

widest	779
wrap-ans	2163
wrap-label	514
wrap-label*	514
wrap-opt	2163

Z

\z	2801
----	------