

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-09-29^{*}

©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 \LaTeX 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	2	6	The storage system	12
1.1	Description and usage	3	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	4	6.1.2	Keys for wrap and display	13
1.3.1	Internal counters	4	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	4	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	4	6.2.1	Keys for <code>\anskey</code>	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	5	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	5	6.4	The environment <code>keyans</code>	15
2	The environments provided	6	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	6	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	6	6.5.1	The command <code>\anspic</code>	16
2.3	The command <code>\item*</code>	6	6.6	Printing stored content	17
2.3.1	Keys for <code>\item*</code>	7	6.6.1	The command <code>\getkeyans</code>	17
2.4	The command <code>\item</code> in <code>enumext*</code>	7	6.6.2	The command <code>\foreachkeyans</code>	17
3	The command <code>\setenumext</code>	7	6.6.3	The command <code>\printkeyans</code>	18
4	The command <code>\setenumextmeta</code>	7	7	Full examples	19
5	The <code>keyval</code> system	8	8	The way of non-enumerated lists	21
5.1	Keys for label and ref	8	9	References	23
5.2	Keys for spaces	9	10	Change history	24
5.2.1	Vertical spaces	9	11	Index of Documentation	25
5.2.2	Horizontal spaces	10	12	Implementation	27
5.3	Keys for add code	10	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 \LaTeX team for their great work and to the different members of the `TeX-SX` community who have provided great answers and ideas. Here a note of the main ones:

1. Answer given by Alan Munn in `\topsep`, `\itemsep`, `\partopsep`, `\parsep` - what do they each mean (and what about the bottom)?
2. Answer given by Enrico Gregorio in Understanding minipages - aligning at top
3. Answer given by Ulrich Diez in Different mechanics of hyperlink vs. hyperref
4. Answer given by Enrico Gregorio in Minipage and multicols, vertical alignment

^{*}This file describes a documentation for v1.0, last revised 2024-09-29.

[†]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) $\angle EDF = 45^\circ$
4. Question with image and label below:

A

A

A

D)

E)

5. Question with image on left side:

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



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

1. B) $x = 5$

2. D)

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

* 5. D) “other note”


*

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 \TeX , but “does not redefine” any internal commands associated with it such as `\list`, `\endlist` or `\item` outside of the “scope” in which they are defined.

- 

This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.
- This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in \TeX Live and $\text{MiK}\text{\TeX}$, use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktextlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “extra space” that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.

Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

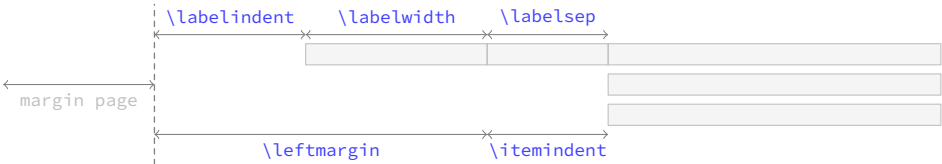


Figure 1: Representation of horizontal lengths in `enumitem`.

The `enumext` package does NOT provide a user interface to set the values for `\leftmargin` and `\itemindent`, instead it provides the keys `list-offset` and `list-indent` which internally set the values for `\leftmargin` and `\itemindent`. The concepts of `\leftmargin` and `\itemindent` are different in `enumext`. The figure 2 shows the visual representation of idea.

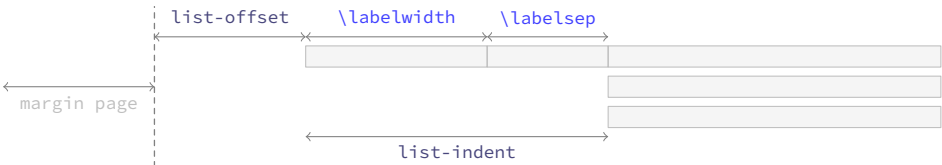


Figure 2: Representation of horizontal lengths concept in `enumext`.

In this way we reduce a *little* the amount of parameters we have to pass. With the default values of keys `list-offset`, `list-indent`, `labelwidth` and `labelsep` the lists will have the (usually) expected output for “simple worksheets”. The figure 3 shows the visual representation.

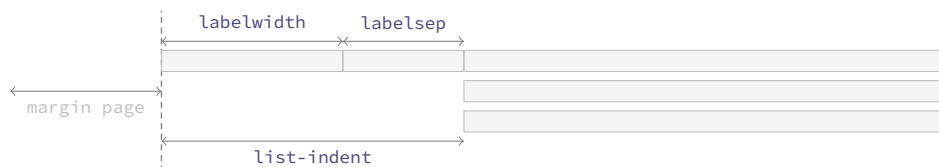


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to “store content” and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual stored content, `\printkeyans` to print all stored content, `\miniright` for `minipage` and `\setenumext` to config all `[<key = val>]` options.

1.3.1 Internal counters

The package `enumext` uses internally the `enumXi`, `enumXii`, `enumXiii`, `enumXiv` counters for the four nesting levels of the `enumext` environment, the `enumXv` counter for the `keyans` environment, the `enumXvi` counter for the `keyanspic` environment, the counter `enumXvii` for `enumext*` environment and the counter `enumXviii` for `keyans*` environment.

- If any package defines these counters or they are user-defined in the document, the package will return a fatal error and abort the load.

1.3.2 Public dimension

The package `enumext` only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. The dimension `\itemwidth` is *rigid length* and contains the “width of the content” of each `\item` regardless of `labelwidth` and `labelsep`.

- If any package defines `\itemwidth` or they are user-defined `\itemwidth` in the document, the package will overwrite it without warning.

1.3.3 Support for multicol

The package provides direct support for using the `multicol`[3] package. This allows to obtain directly a two-column output as shown in the figure 4.



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

The “non starred” version of the `multicols` environment is always used together with the `\raggedcolumns` command and is controlled by `columns` and `columns-sep` keys. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and can together with the `mini-env` key. If you need to force a start a new column `\columnbreak` must be used (see §5.5).

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

1.3.4 Support for minipage

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



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

The `minipage` environments on “left side” and “right side” is always used with “aligned on top” `[t]`. It can be used in all nesting levels of the environment `enumext` and the environment `keyans` and is controlled by `mini-env` and `mini-sep` keys. In order to switch from the “left” side `minipage` environment to the “right” side one must use the command `\miniright` (see §5.6).

1.3.5 The `\label` and `\ref` system

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

1.3.6 Support for `\footnote`

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

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

2.4 The command \item in enumext*

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

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

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

3 The command \setenumext

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

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

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

- The key `save-ans` that activate the “storage system” must NOT be passed through this command and must be passed directly in the *optional argument* of the “first level” of the environment in which they are executed.

4 The command \setenumextmeta

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

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

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

5 The keyval system

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

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

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

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

5.1 Keys for `label` and `ref`

`label = { $\langle \backslash alph* | \backslash Alph* | \backslash arabic* | \backslash roman* | \backslash Roman* \rangle$ }` default: *by levels*

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

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

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

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

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

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

5.2 Keys for spaces

`show-length` = { $\langle true | false \rangle$ } 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` = { $\langle rubber\ length | rigid\ length \rangle$ } 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` = { $\langle rubber\ length | rigid\ length \rangle$ } 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` = { $\langle rubber\ length | rigid\ length \rangle$ } 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 $\langle vertical\ mode \rangle$ and apply this value to the “top” and “bottom” the environment or nested level.

`itemsep` = { $\langle rubber\ length | rigid\ length \rangle$ } 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` $\langle value\ forbidden \rangle$ default: *not used*

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

`nosep` $\langle value\ forbidden \rangle$ default: *not used*

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

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

This is a “meta-key” that does not receive an argument available only for the *first level* of environment `enumext` and environment `enumext*`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` or vice versa 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 $\langle keys \rangle$ should be used with “caution”, they are intended to be used at the “top” and “bottom” of the environment when the `columns` or `mini-env` keys do not provide adequate *vertical spaces*. The values passed can be *rubber* or *rigid* lengths, the way they are applied is the way you differ, using the star ‘*’ $\langle keys \rangle$ applies `\vspace*` so that TeX does *not discard* this space at page break.

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

`list-indent` = { $\langle rigid\ length \rangle$ } default: *labelwidth + labelsep*

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

If `list-indent=0pt` is set in the environment `enumext` the $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

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

5.3 Keys for add code

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

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

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

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

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

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

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

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

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

Execute $\{\langle code \rangle\}$ “*after*” finishing the environment. The $\{\langle code \rangle\}$ must be passed between braces.

5.4 Keys for start, series and resume

`start` = { $\langle integer \mid integer\ expression \rangle$ } default: *1*

Sets the *start value* of the numbering on the current level. The $\{\langle integer\ expression \rangle\}$ 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*` = { $\langle integer \mid string \rangle$ } default: *not used*

Sets the *start value* of the numbering on the current level. Internally $\langle string \rangle$ 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 $\langle keys \rangle$ are “*only*” available for the `enumext*` environment and the “*first level*” of the `enumext` environment and are ignored if set when nested within each other.

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

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

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

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

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

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

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

5.5 Keys for multicol

`columns = {⟨integer⟩}` default: **1**

Set the *number of columns* to be used by the `multicol` environment within the 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 `multicol` 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 `multicol` 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*` = { $\langle symbol \rangle$ } default: $\$ \backslash star \$$
 Sets the *symbol* for `\item*` when using the key `item-star` and stores { $\langle content \rangle$ } in the *sequence* { $\langle store name \rangle$ } of the form `\item*[\langle symbol \rangle] \langle content \rangle`. The *symbol* can be in text or math mode, for example `item-sym*={ $\$ \backslash ast \$$ }` stores `\item*[\mathast] \langle content \rangle`.

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

Example

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

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

6.3 The environment `anskey*`

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

The environment `anskey*` takes a mandatory { $\langle body content \rangle$ } and “stores” it in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “*label and ref*” system provided by 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 [$\langle key = val \rangle$] on the same line are NOT supported, the package `scontents` will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskey*}` on the same line the package `scontents` will return a “warning” message.

6.3.1 Keys for `anskey*`

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

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

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

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

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

Example

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

- | | |
|---|--|
| <p>★ 5. Text containing our instructions or questions.</p> <p>[5] <input type="text" value="First answer with verbatim"/></p> <p>6. Text containing our instructions or questions.</p> <p>(a) Question.</p> <p>[6] <input type="text" value="second answer"/></p> | <p>7. Text containing our instructions or questions.</p> <p>[7] <input type="text" value="third answer"/></p> <p>8. Text containing our instructions or questions.</p> <p>[8] <input type="text" value="fourth answer"/></p> |
|---|--|

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.

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

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

6.4.1 The \item* in keyans and keyans*

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

1. Text containing a question.

A) Choice

* B) Correct choice

C) Choice

D) Choice

E) Choice

2. Text containing a question and image.


A) Choice

B) Choice

C) Choice

D) Choice

* E) [note] Correct choice



Some text

6.5 The environment keyanspic

keyanspic

```
\begin{keyanspic}*[\textit{n}^{\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.

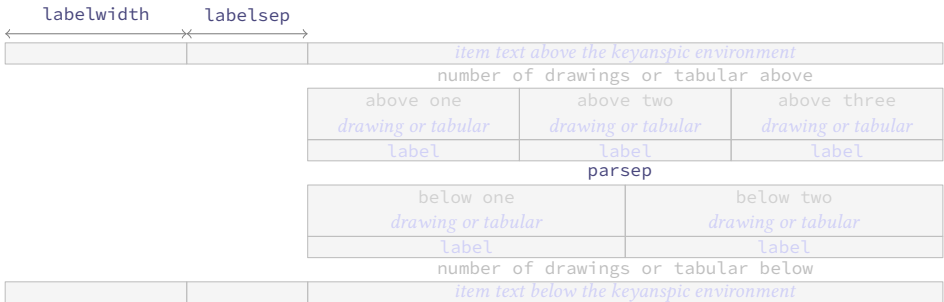


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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

6.5.1 The command \anspic

\anspic

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

The `\anspic` command take three arguments, the *starred argument* ‘*’ store the current `\label` next to the *optional argument* `\textit{content}` in *sequence* and *prop list* `{\textit{store name}}` set by `save-ans` key.

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

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
  \item Question with images.
  \begin{keyanspic}[3,2]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
  \end{keyanspic}
\end{enumext}
```

1. Question with images.



A)



B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

- `sep = {<code>}` default: *empty*
 Establishes the *separation* between “each” `{<content>}` stored in *prop list* `{<store name>}`. For example, you can use `sep={\[\[10pt]}` for vertical separation of stored contents.
- `step = {<integer>}` default: **1**
 Sets the *step* (increment) applied to the value set by key `start` for each `{<content>}` stored in *prop list* `{<store name>}`. The value must be a *positive integer*.
- `start = {<integer>}` default: **1**
 Sets the *position* of the *prop list* `{<store name>}` from which execution will start. The value must be a *positive integer*.
- `stop = {<integer>}` default: **0**
 Sets the *position* of the *prop list* `{<store name>}` from which execution it will finish executing. The value must be a *positive integer*.
- `before = {<code>}` default: *empty*
 Sets the `{<code>}` that will be executed *before* each `{<content>}` stored in *prop list* `{<store name>}`. The `{<code>}` must be passed between braces.
- `after = {<code>}` default: *empty*
 Sets the `{<code>}` that will be executed *after* each `{<content>}` stored in *prop list* `{<store name>}`. The `{<code>}` must be passed between braces.
- `wrapper = {<code> {#1} more code}` default: *empty*
 Wraps the `{<content>}` stored in *prop list* `{<store name>}` referenced by `{#1}`. The `{<code>}` must be passed between braces. For example `\foreachkeyans[wrapper={\makebox[1em][l]{#1}}]{<store name>}`.

6.6.3 The command `\printkeyans`

```
\printkeyans \printkeyans{<store name>}
\printkeyans[<keys>]{<store name>}
\printkeyans*[<keys>]{<store name>}
```

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

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

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

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

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

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

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

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

Example

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

$3(x + y + z)$

2. True False

(a)

TeX is cool?

[2]

Very True!

3. Related to Linux

(a) You use linux?
- [3]

Yes

(b) Rate the following package and class

i.

xsim

[4]

very good

ii.

exsheets

[5]

obsolete

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 arara¹ 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.

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

©2024 by Pablo González L

19 / 155

- C

27,8 km/h.
- D

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

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

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

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

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}.$
1. B
2. A
3. B
4. A
- *
- *
- *
- *

Example 3

A “simple multiple choice” test 📄.

1. First type of questions
- A

value
- B

correct
- C

value
- D

value
2. Second type of questions
- I. $2\alpha + 2\delta = 90^\circ$
- II. $\alpha = \delta$
- III. $\angle EDF = 45^\circ$
- A

I only
- B

II only
- C

I and II only
- D

I and III only
- E

I, II, and III
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$
- (2) $\angle EDF = 45^\circ$
- A

value
- B

value
- C

value
- D

value
- E

value
4. Question with image and label below:

A

A

D

B

B

A

E

B

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 :) 📄.

- ca. 1.

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

Factor $3x + 3y + 3z$
- The following questions need to be cuaqtified :)
- ca. 3.

True False
- (a) $\alpha > \delta$

(b) \LaTeX 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

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

1. A)

2. C)

3. B)

4. D)

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

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

LoNg

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 \LaTeX ₃ Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The \LaTeX _{2_ε} 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 2024-09-29 – First public release.

11 Index of Documentation

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

C		I	
Document class:		\itemsep	9
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*	6-8, 12, 13, 15, 16	step	17
\item	6-8, 10, 11, 13, 15	stop	17
\miniright	11	wrapper	17
\printkeyans	7, 12, 18	Keys for anskey* provide by enumext:	
\setenumextmeta	7	break-col	13
\setenumext	6-8, 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*	9
enumXv	4	above	9
E		after	10, 11
Environments provide by enumext:		align	8, 22
anskey*	12-14	base-fix	9
enumext*	4-15, 18	before*	10
enumext	4-10, 12-15, 18, 21	before	10
keyans*	4-15	below*	10
keyanspic	4, 8, 9, 12-14, 16, 21	below	9
keyans	4-10, 12-16, 21	check-ans	13
Environments:		columns-sep	4, 11
Verbatim	14	columns	4, 9, 11
center	6	first	10
description	6	font	8
enumerate	1, 3, 6, 23	item-pos*	6, 7
figure	6	item-sym*	6, 7
flushleft	6	itemindent	10
flushright	6	itemsep	9
itemize	6	labelsep	4, 6-8, 10, 11, 13, 22
list	3, 6, 10, 23	labelwidth	4, 7, 8, 10, 11, 13, 22
minipage	3-6, 11, 23	labelwith	6
multicols	3, 4, 11	label	8, 10, 15, 21, 22
quotation	6	list-indent	3, 4, 10
quote	6	list-offset	3, 4, 10, 22
tabbing	6	listparindent	10
table	6	mark-ans	13
task	6	mark-pos	13
trivlist	6	mark-ref	12
verbatim	6	mini-env	4, 5, 9, 11, 12
verse	6	mini-right*	8, 11, 12
F		mini-right	8, 11, 12
\footnote	6	mini-sep	4, 11, 12
		no-store	12-14

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

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   {2024-09-29}
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 `{⟨store name⟩}` 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 `{⟨store name⟩}` 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 `⟨body⟩` and the `⟨keys⟩` 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_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 “*storage level system*”.

```

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

```

(End of definition for `\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).

```

141 \seq_new:N \l__enumext_anspic_args_seq
142 \dim_new:N \l__enumext_anspic_mini_width_dim
143 \int_new:N \l__enumext_anspic_above_int
144 \int_new:N \l__enumext_anspic_below_int
145 \bool_new:N \l__enumext_keyans_pic_star_bool
146 \str_new:N \l__enumext_anspic_mini_pos_str
147 \skip_new:N \g__enumext_keyans_pic_parsep_skip
148 \box_new:N \l__enumext_anspic_label_box
149 \box_new:N \l__enumext_anspic_body_box
150 \dim_new:N \l__enumext_anspic_label_htdp_dim
151 \dim_new:N \l__enumext_anspic_body_htdp_dim

```

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

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

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

```
152 \bool_new:N \l__enumext_check_answers_bool
153 \bool_new:N \g__enumext_check_ans_key_bool
154 \tl_new:N \l__enumext_check_start_line_env_tl
155 \int_new:N \g__enumext_check_starred_cmd_int
156 \int_new:N \g__enumext_item_anskey_int
157 \int_new:N \g__enumext_item_number_int
158 \bool_new:N \l__enumext_item_number_bool
159 \int_new:N \g__enumext_item_answer_diff_int
```

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

```
\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool
```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§12.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
160 \bool_new:N \l__enumext_hyperref_bool
161 \bool_new:N \l__enumext_footnotes_key_bool
```

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

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

Internal variables used by `save-ref` key (§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.

```
162 \tl_new:N \l__enumext_newlabel_arg_one_tl
163 \tl_new:N \l__enumext_newlabel_arg_two_tl
164 \tl_new:N \l__enumext_write_aux_file_tl
165 \cs_set_protected:Npn \__enumext_tmp:n #1
166 {
167   \tl_new:c { \l__enumext_label_copy_#1_tl }
168 }
169 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq
```

Internal variables used for redefinition of `\footnote` (§12.42.4).

```
170 \int_new:N \g__enumext_footnote_int
171 \seq_new:N \g__enumext_footnote_arg_seq
172 \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`.)

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

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

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

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

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

(End of definition for `\c__enumext_all_envs_clist`.)

12.5 Some utility functions

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

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

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

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

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

(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`

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

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

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

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

(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`

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

(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.

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


(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.

```

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

```

(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.

```

237 \cs_new_protected:Nn \__enumext_unskip_unkern:
238 {
239     \int_case:nnT { \lastnodetype }
240     {
241         { 11 }
242         {
243             % \typeout{SKIP} \typeout{\the\lastskip}
244             \unskip
245         }
246         { 12 }
247         {
248             % \typeout{KERN} \typeout{\the\lastkern}
249             \unkern
250         }
251     }
252 }
253 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
254 \cs_new_protected:Nn \__enumext_unskip_unkern_new:
255 {
256     \int_case:nnT { \lastnodetype }
257     {
258         { 11 }
259         {
260             \typeout{SKIPPPPPPPPPPPPPPPPP} \typeout{\the\lastskip}
261             \unskip
262         }
263         { 12 }
264         {
265             \typeout{KERNNNNNNNNNNNNNNNNN} \typeout{\the\lastkern}
266             \unkern
267         }
268         { 13 }
269         {
270             \typeout{penaltyyyyyyyyyyyyyyyyy}
271         }
272     }
273 }

```

(End of definition for `__enumext_unskip_unkern:`.)

12.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`

`__enumext_is_on_first_level:`

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

```

274 \cs_new_protected:Nn \__enumext_is_not_nested:
275 {
276     \str_case:en { \@currenvir }
277     {
278         {enumext}
279         {
280             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
281             \bool_lazy_and:nnT
282             { \bool_not_p:n { \g__enumext_standar_bool } }
283             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
284             {
285                 \bool_gset_true:N \g__enumext_standar_bool

```

```

286         }
287     }
288     {enumext*}
289     {
290         \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
291         \bool_lazy_and:nnT
292         { \bool_not_p:n { \g__enumext_starred_bool } }
293         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
294         {
295             \bool_gset_true:N \g__enumext_starred_bool
296         }
297     }
298 }
299 }

```

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.

```

300 \cs_new_protected:Nn \__enumext_is_on_first_level:
301 {
302     \bool_lazy_all:nT
303     {
304         { \bool_if_p:N \g__enumext_standar_bool }
305         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
306         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
307     }
308     {
309         \bool_set_true:N \l__enumext_standar_first_bool
310         \bool_set_true:N \l__enumext_anskey_env_bool
311         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
312         \tl_gset:Ne \g__enumext_start_line_tl
313         {
314             on ~ line ~ \exp_not:V \inputlineno
315         }
316     }
317     \bool_lazy_all:nT
318     {
319         { \bool_if_p:N \g__enumext_starred_bool }
320         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
321         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
322     }
323     {
324         \bool_set_true:N \l__enumext_starred_first_bool
325         \bool_set_true:N \l__enumext_anskey_env_bool
326         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
327         \tl_gset:Ne \g__enumext_start_line_tl
328         {
329             on ~ line ~ \exp_not:V \inputlineno
330         }
331     }
332 }

```

(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.

```

333 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
334 {
335     \str_case:en { \@currenvir }
336     {
337         {keyans}
338         {
339             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
340             \tl_set:Ne \l__enumext_check_start_line_env_tl
341             {
342                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
343             }
344         }
345     }

```

```

344     }
345     {keyans*}
346     {
347         \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
348         \tl_set:Ne \l__enumext_check_start_line_env_tl
349         {
350             in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
351         }
352     }
353     {keyanspic}
354     {
355         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
356         \tl_set:Ne \l__enumext_check_start_line_env_tl
357         {
358             in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
359         }
360     }
361 }
362 }

```

(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.

```

\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
363 \cs_new_protected:Nn \__enumext_reset_global_vars:
364 {
365     \__enumext_reset_global_int:
366     \__enumext_reset_global_bool:
367     \__enumext_reset_global_tl:
368 }
369 \cs_new_protected:Nn \__enumext_reset_global_int:
370 {
371     \int_gzero:N \g__enumext_item_number_int
372     \int_gzero:N \g__enumext_item_anskey_int
373     \int_gzero:N \g__enumext_item_answer_diff_int
374 }
375 \cs_new_protected:Nn \__enumext_reset_global_bool:
376 {
377     \bool_gset_false:N \g__enumext_check_ans_key_bool
378     \bool_gset_false:N \g__enumext_standar_bool
379     \bool_gset_false:N \g__enumext_starred_bool
380 }
381 \cs_new_protected:Nn \__enumext_reset_global_tl:
382 {
383     \tl_gclear:N \g__enumext_store_name_tl
384     \tl_gclear:N \g__enumext_start_line_tl
385     \tl_gclear:N \g__enumext_envir_name_tl
386 }

```

(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.

```

387 \cs_new_protected:Nn \__enumext_log_global_vars:
388 {
389     \msg_log:nneeee { enumext } { prop-seq-int-hook }
390     { \g__enumext_store_name_tl }
391     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
392     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
393     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
394 }

```

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.

```

395 \cs_new_protected:Nn \__enumext_log_answer_vars:
396 {
397     \msg_log:nneee { enumext } { item-answer-hook }
398     { \int_use:N \g__enumext_item_number_int }

```

```

399     { \int_use:N \g__enumext_item_anskey_int }
400     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
401   }

```

(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 \TeX has the following plain form:

```

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

```

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

```

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

```

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

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

```

\__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.

```

402 \__enumext_at_begin_document:n
403 {
404   \cs_new_eq:NN \__enumext_start_list:nn \list
405   \cs_new_eq:NN \__enumext_stop_list: \endlist
406   \NewCommandCopy \__enumext_item_std:w \item
407 }

```

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

```

408 \__enumext_at_begin_document:n
409 {
410   \cs_new_eq:NN \__enumext_minipage:w \minipage
411   \cs_new_eq:NN \__enumext_endminipage: \endminipage
412 }

```

(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 a internal `__enumext_mini_page` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on “top” in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces “below”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§12.38) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§12.43)

```

413 \cs_new_protected:Nn \__enumext_internal_mini_page:
414 {
415   \int_compare:nNt { \l__enumext_level_int } = { 0 }
416   {
417     \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
418     {
419       \__enumext_minipage:w [ t ] { ##1 }
420       \legacy_if_gset_false:n { @minipage }
421       \skip_vertical:N \c_zero_skip
422     }
423     {
424       \skip_vertical:N \c_zero_skip
425       \__enumext_endminipage:
426     }
427   }
428 }

```

(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.

```
429 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
430 \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”.

```
431 \cs_new_protected:Nn \__enumext_after_hyperref:
432 {
433   \IfPackageLoadedTF { hyperref }
434   {
435     \msg_info:nnn { enumext } { package-load } { hyperref }
436     \bool_set_true:N \l__enumext_hyperref_bool
437     \IfHyperBoolean{hyperfootnotes}
438     {
439       % \typeout{hyperfootnotes=true}
440       \bool_set_true:N \l__enumext_footnotes_key_bool
441     }
442     {
443       % \typeout{hyperfootnotes=false}
444     }
445   }
446 }
```

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`.

```
447 \bool_if:NT \l__enumext_footnotes_key_bool
448 {
449   \IfPackageLoadedTF { footnotehyper }
450   {
451     \msg_info:nnn { enumext } { package-load } { footnotehyper }
452   }
453   {
454     % \typeout{No ~ footnotehyper ~ load}
455     % \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
456     \bool_set_false:N \l__enumext_footnotes_key_bool
457   }
458 }
```

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.

```
459 \bool_if:NTF \l__enumext_hyperref_bool
460 {
461   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
462   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
463 }
464 {
465   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
466   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
467 }
468 }
```

(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.

```
469 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
470 {
471   \protected@write \@auxout { }
472 }
```

```

473     \token_to_str:N \newlabel {#1}
474     {
475         {#2}
476         \bool_if:NT \l__enumext_hyperref_bool
477         { { \thepage } {#2} {#1} }
478         { }
479     }
480 }
481 \__enumext_hypertarget:nn {#1} { }
482 \__enumext_phantomsection:
483 }

```

(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.

```

484 \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 `\l__enumext_counter_X_tl` for “store” the counter’s name.

#2 : The counter’s name.

```

485 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
486 {
487     \cs_if_exist:cTF { c@ #2 }
488     { \msg_fatal:nnn { enumext } { counters } { #2 } }
489     {
490         \tl_set:Nn #1 { #2 }
491         \newcounter { #2 }
492     }
493 }

```

(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      494 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
enumXii     495 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
enumXiii    496 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXvii    497 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
enumXviii   498 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
enumXvii    499 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
enumXviii   500 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
enumXviii   501 \__enumext_define_counters:Nn \l__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.

```

502 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
503 {
504     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
505     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
506 }
507 \__enumext_register_counter_style:Nn \arabic { 0 }
508 \__enumext_register_counter_style:Nn \Alph { M }
509 \__enumext_register_counter_style:Nn \alph { m }
510 \__enumext_register_counter_style:Nn \Roman { VIII }
511 \__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.

```

512 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
513 {
514   \hbox_set:Nn \__enumext_label_width_by_box {#2}
515   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
516 }
517 \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.

```

518 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
519 {
520   \tl_clear_new:N #1
521   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
522   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
523   \tl_map_inline:Nn \g__enumext_counter_styles_tl
524   {
525     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
526     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
527     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
528   }
529   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
530   { \tl_use:N \g__enumext_widest_label_tl }
531   \tl_set_eq:cN { the #2 } #1
532 }
533 \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`
`labelsep`
`labelwidth`
`wrap-label`
`wrap-label*`

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

```

534 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
535 {
536   \keys_define:nn { enumext / #1 }
537   {
538     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
539     font      .value_required:n = true,
540     labelsep  .dim_set:c  = { l__enumext_labelsep_#2_dim },
541     labelsep  .initial:n   = {0.3333em},
542     labelsep  .value_required:n = true,
543     labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
544     labelwidth .value_required:n = true,
545     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
546     wrap-label .initial:n   = {##1},
547     wrap-label .value_required:n = true,
548     wrap-label* .code:n = {
549       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
550       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
551     },
552     wrap-label* .value_required:n = true,
553   }
554 }
555 \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.

```

556 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
557 {
558   \keys_define:nn { enumext / #1 }
559   {
560     align .choice:,
561     align / left .code:n =
562       {
563         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
564         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
565         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
566       },
567     align / right .code:n =
568       {
569         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
570         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
571         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
572       },
573     align / center .code:n =
574       {
575         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
576         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
577         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
578       },
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
587 {
588   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
589 }
590 { \__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 `\makelabel` is redefined and the only way to get the `align` key to work correctly is by using `\makebox`.

```

591 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
592 {
593   \keys_define:nn { enumext / #1 }
594   {
595     align .choice:,
596     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
597     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
598     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
599     align / unknown .code:n =
600       \msg_error:nneee { enumext } { unknown-choice }
601       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
602     align .initial:n = left,
603     align .value_required:n = true,
604   }
605 }
606 \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 $\langle label \rangle$, 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

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
607 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
608 {
609   \keys_define:nn { enumext / #1 }
610   {

```

```

611     label .code:n = {
612         \__enumext_label_style:cvn { l__enumext_label_#2_tl }
613         { l__enumext_counter_#2_tl } {##1}
614         \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
615         \l__enumext_current_widest_dim
616     },
617     label .initial:n = #3,
618     label .value_required:n = true,
619     ref .code:n = \__enumext_standar_ref:n {##1},
620     ref .value_required:n = true,
621 }
622 }
623 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
624 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
625 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
626 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

__enumext_standar_ref:n
 __enumext_standar_ref:

The `__enumext_standar_ref:n` first we will pass the key argument to `__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 `__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will return the modified `__enumext_ref_key_arg_tl` and we make the value of `__enumext_ref_the_count_tl` the same as that `__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `__enumext_renew_the_count_X_tl` with the renewed command.

```

627 \cs_new_protected:Npn \__enumext_standar_ref:n #1
628 {
629     \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
630     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
631     {
632         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
633     }
634     {
635         \tl_set_eq:Nc
636         \__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
637         \__enumext_regex_counter_style:
638         \tl_set_eq:Nc
639         \__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
640         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
641         {
642             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
643             { \exp_not:V \__enumext_ref_key_arg_tl }
644         }
645     }
646 }

```

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

```

647 \cs_new_protected:Nn \__enumext_standar_ref:
648 {
649     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
650     {
651         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
652     }
653 }

```

(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
654 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
655 {
656     \keys_define:nn { enumext / #1 }
657     {
658         label .code:n = {
659             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
660             { l__enumext_counter_#2_tl } {##1}
661             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
662             \l__enumext_current_widest_dim
663         },

```

```

664     label .initial:n = #3,
665     label .value_required:n = true,
666     ref .code:n = \__enumext_starred_ref:n {##1},
667     ref .value_required:n = true,
668   }
669 }
670 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
671 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

__enumext_starred_ref:n
 __enumext_starred_ref:

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

```

672 \cs_new_protected:Npn \__enumext_starred_ref:n #1
673 {
674   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
675   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
676   {
677     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
678     {
679       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
680     }
681     {
682       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
683       \__enumext_regex_counter_style:
684       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
685       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
686       {
687         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
688         { \exp_not:V \l__enumext_ref_key_arg_tl }
689       }
690     }
691   }
692   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
693   {
694     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
695     {
696       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
697     }
698     {
699       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
700       \__enumext_regex_counter_style:
701       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
702       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
703       {
704         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
705         { \exp_not:V \l__enumext_ref_key_arg_tl }
706       }
707     }
708   }
709 }

```

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.

```

710 \cs_new_protected:Npn \__enumext_starred_ref:
711 {
712   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
713   {
714     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
715     {
716       \tl_use:N \l__enumext_renew_the_count_vii_tl
717     }
718   }
719   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
720   {
721     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
722     {
723       \tl_use:N \l__enumext_renew_the_count_viii_tl
724     }
725   }
726 }

```

(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 $\langle label \rangle$ for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` and `ref` key. The `keyanspic` environment use the same $\langle label \rangle$ as the `keyans` environment.

```

label
ref
\l__enumext_label_v_tl
\l__enumext_label_vi_tl
727 \keys_define:nn { enumext / keyans }
728 {
729   label .code:n = {
730     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
731     { \l__enumext_counter_v_tl } {#1}
732     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
733     \l__enumext_current_widest_dim
734     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
735     { \l__enumext_counter_vi_tl } {#1}
736     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
737     \l__enumext_current_widest_dim
738   },
739   label .initial:n = \Alph*,
740   label .value_required:n = true,
741   ref .code:n = \__enumext_keyans_ref:n {#1},
742   ref .value_required:n = true,
743 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
744 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
745 {
746   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
747   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
748   {
749     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
750   }
751   {
752     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
753     \__enumext_regex_counter_style:
754     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
755     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
756     {
757       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
758       { \exp_not:V \l__enumext_ref_key_arg_tl }
759     }
760   }
761 }

```

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

```

762 \cs_new_protected:Nn \__enumext_keyans_ref:
763 {
764   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
765   {
766     \tl_use:N \l__enumext_renew_the_count_v_tl
767   }
768 }

```

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

12.14 Setting start, start* and widest keys

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

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn
\__enumext_start_from:cce
#1: \l__enumext_label_X_tl
#2: \l__enumext_start_X_int
#3:  $\langle integer \text{ or } string \rangle$ 

```

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 $\langle integer \rangle$ or $\langle string \rangle$ of the form `\Alph`, `\alph`, `\Roman` or `\roman`. This effectively allows `start=A` or `start=1` to be used.

```

769 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
770 {
771   \__enumext_if_is_int:nTF { #3 }
772   {
773     \int_set:Nn #2 {#3}
774   }
775   {

```

```

776         \regex_match:nVT { \c{Alpha} | \c{alpha} } {#1}
777         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
778         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
779         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
780     }
781 }
782 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

#1: The counter associated with the environment level

#2: `\l__enumext_label_X_tl`

#3: `\l__enumext_labelwidth_X_dim`

#4: *⟨integer or string⟩*

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

```

783 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
784 {
785     \__enumext_if_is_int:nTF {#4}
786     {
787         \setcounter{enumX#1} { #4 }
788     }
789     {
790         \regex_match:nVT { \c{Alpha} | \c{alpha} } {#2}
791         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
792         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
793         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
794     }
795     \__enumext_label_width_by_box:cv
796     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
797 }
798 \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.

```

799 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
800 {
801     \keys_define:nn { enumext / #1 }
802     {
803         start* .code:n = {
804             \__enumext_start_from:ccn
805             { \l__enumext_label_#2_tl }
806             { \l__enumext_start_#2_int } {##1}
807         },
808         start* .value_required:n = true,
809         start .code:n = {
810             \__enumext_start_from:cce
811             { \l__enumext_label_#2_tl }
812             { \l__enumext_start_#2_int } { \int_eval:n {##1} }
813         },
814         start .initial:n = 1,
815         start .value_required:n = true,
816         widest .code:n = {
817             \__enumext_widest_from:nccn {#2}
818             { \l__enumext_label_#2_tl }
819             { \l__enumext_labelwidth_#2_dim } {##1}
820         },
821         widest .value_required:n = true,
822     }
823 }
824 \clist_map_inline:Nn \c__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.

```

topsep
partopsep
parsep
noitemsep
nosep
825 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
826 {
827   \keys_define:nn { enumext / #1 }
828   {
829     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
830     topsep .initial:n = {#3},
831     topsep .value_required:n = true,
832     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
833     partopsep .initial:n = {#4},
834     partopsep .value_required:n = true,
835     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
836     parsep .initial:n = {#5},
837     parsep .value_required:n = true,
838     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
839     itemsep .initial:n = {#6},
840     itemsep .value_required:n = true,
841     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
842     noitemsep .value_forbidden:n = true,
843     nosepp .meta:n = {
844       itemsep = 0pt, parsep= 0pt,
845       topsep = 0pt, partopsep = 0pt,
846     },
847     nosepp .value_forbidden:n = true,
848   }
849 }
```

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

```

850 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
851 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
852 { 4.0pt plus 2.0pt minus 1.0pt }
853 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
854 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
855 { 2.0pt plus 1.0pt minus 1.0pt }
856 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
857 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
858 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
859 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
860 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
861 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
862 { 2.0pt plus 1.0pt minus 1.0pt }
863 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
864 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
865 { 4.0pt plus 2.0pt minus 1.0pt }
866 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
867 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
868 { 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.

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

```

base-fix
\__enumext_nested_base_line_fix:
869 \keys_define:nn { enumext / level-1 }
870 {
871   base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
872   base-fix .initial:n = false,
873   base-fix .value_forbidden:n = true,
874 }
```

The function `__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the `(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)

• This key is enabled by default in the command `\printkeyans` (§12.46).

```

875 \bool_new:N \__enumext_print_keyans_star_bool
876 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
877 {
878   \mode_leave_vertical:
879   \bool_lazy_and:nnT
880     { \bool_if_p:N \__enumext_starred_first_bool }
881     { \bool_not_p:n { \__enumext_print_keyans_star_bool } } % only for pdflatex-dev (not tagged)
882     {
883       \vspace{-\dimeval{\baselineskip + \parsep}}
884     }
885   \bool_lazy_and:nnT
886     { \bool_if_p:N \__enumext_starred_first_bool }
887     { \bool_if_p:N \__enumext_print_keyans_star_bool } % for tagged PDF
888     {
889       \skip_vertical:n { -\baselineskip }
890       \skip_vertical:N \c_zero_skip
891     }
892   \keys_set:nn { enumext / level-1 }
893   {
894     topsep = 0pt, above = 0pt, above* = 0pt,
895   }
896   \bool_set_false:N \__enumext_base_line_fix_bool
897 }

```

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

```

898 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
899 {
900   \keys_define:nn { enumext / #1 }
901   {
902     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
903     itemindent .value_required:n = true,
904     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
905     rightmargin .value_required:n = true,
906     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
907     listparindent .value_required:n = true,
908     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
909     list-offset .value_required:n = true,
910     list-indent .code:n =
911       {
912         \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
913         \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
914       }
915   }
916   \clist_map_inline:nn
917     {
918       {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
919     }
920     { \__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.

```

921 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
922 {
923   \keys_define:nn { enumext / #1 }
924   {
925     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
926     itemindent .value_required:n = true,
927     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
928     rightmargin .value_required:n = true,
929     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
930     listparindent .value_required:n = true,
931     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
932     list-offset .value_required:n = true,
933     list-indent .meta:n = { list-offset = ##1 },

```

```

934     list-indent .value_required:n = true,
935   }
936 }
937 \clist_map_inline:nn
938 {
939   {enumext*}{vii}, {keyans*}{viii}
940 }
941 { \__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 `\opt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

942 \cs_set_protected:Nn \__enumext_fake_item_indent:
943 {
944   \dim_compare:nNnT
945     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
946     >
947     { \c_zero_dim }
948   {
949     \tl_set:ce { \__enumext_fake_item_indent_ \__enumext_level: _tl }
950     {
951       \exp_not:N \mode_leave_vertical:
952       \exp_not:n { \skip_horizontal:n }
953       { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
954       \ignorespaces
955     }
956   }
957 }
958 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
959 {
960   \dim_compare:nNnT
961     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
962     {
963       \tl_set:Nc \l__enumext_fake_item_indent_v_tl
964       {
965         \exp_not:N \mode_leave_vertical:
966         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
967         \ignorespaces
968       }
969     }
970   }
971 \cs_set_protected:Nn \__enumext_fake_item_vii:
972 {
973   \dim_compare:nNnT
974     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
975     {
976       \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
977       {
978         \exp_not:N \mode_leave_vertical:
979         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
980         \ignorespaces
981       }
982     }
983   }
984 \cs_set_protected:Nn \__enumext_fake_item_viii:
985 {
986   \dim_compare:nNnT
987     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
988     {
989       \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
990       {
991         \exp_not:N \mode_leave_vertical:
992         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
993         \ignorespaces
994       }
995     }
996   }

```

(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 `__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.

```

997 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
998 {
999   \keys_define:nn { enumext / #1 }
1000   {
1001     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1002     show-length .initial:n = false,
1003   }
1004 }
1005 \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

before

before*

after

first

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

```

1006 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1007 {
1008   \keys_define:nn { enumext / #1 }
1009   {
1010     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
1011     before .value_required:n = true,
1012     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1013     before* .value_required:n = true,
1014     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1015     after .value_required:n = true,
1016     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1017     first .value_required:n = true,
1018   }
1019 }
1020 \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}`.

```

1021 \cs_new_protected:Nn \__enumext_before_args_exec:
1022 {
1023   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1024 }

```

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}`

```

1025 \cs_new_protected:Nn \__enumext_before_keys_exec:
1026 {
1027   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1028 }

```

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

```

1029 \cs_new_protected:Nn \__enumext_after_stop_list:
1030 {
1031   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1032 }

```

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.`

```

1033 \cs_new_protected:Nn \__enumext_after_args_exec:
1034 {
1035   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1036 }

```

(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:
1037 \cs_new_protected:Nn __enumext_before_args_exec_v:
1038 {
1039   \tl_use:N \l__enumext_before_starred_key_v_tl
1040 }
1041 \cs_new_protected:Nn __enumext_before_keys_exec_v:
1042 {
1043   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1044 }
1045 \cs_new_protected:Nn __enumext_after_stop_list_v:
1046 {
1047   \tl_use:N \l__enumext_after_stop_list_v_tl
1048 }
1049 \cs_new_protected:Nn __enumext_after_args_exec_v:
1050 {
1051   \tl_use:N \l__enumext_after_list_args_v_tl
1052 }

```

(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*

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

```

__enumext_before_args_exec_vii:
__enumext_before_keys_exec_vii
__enumext_after_stop_list_vii:
__enumext_after_args_exec_vii:
1053 \cs_new_protected:Nn __enumext_before_args_exec_vii:
1054 {
1055   \tl_use:N \l__enumext_before_starred_key_vii_tl
1056 }
1057 \cs_new_protected:Nn __enumext_before_args_exec_viii:
1058 {
1059   \tl_use:N \l__enumext_before_starred_key_viii_tl
1060 }
1061 \cs_new_protected:Nn __enumext_before_keys_exec_vii:
1062 {
1063   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1064 }
1065 \cs_new_protected:Nn __enumext_before_keys_exec_viii:
1066 {
1067   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1068 }
1069 \cs_new_protected:Nn __enumext_after_stop_list_vii:
1070 {
1071   \tl_use:N \l__enumext_after_stop_list_vii_tl
1072 }
1073 \cs_new_protected:Nn __enumext_after_stop_list_viii:
1074 {
1075   \tl_use:N \l__enumext_after_stop_list_viii_tl
1076 }
1077 \cs_new_protected:Nn __enumext_after_args_exec_vii:
1078 {
1079   \tl_use:N \l__enumext_after_list_args_vii_tl
1080 }
1081 \cs_new_protected:Nn __enumext_after_args_exec_viii:
1082 {
1083   \tl_use:N \l__enumext_after_list_args_viii_tl
1084 }

```

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

12.20 Setting keys for multicols and minipage

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

```

1085 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1086 {
1087   \keys_define:nn { enumext / #1 }
1088   {
1089     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1090     mini-env .value_required:n = true,
1091     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },

```

```

1092     mini-sep .initial:n = 0.3333em,
1093     mini-sep .value_required:n = true,
1094     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1095     columns-sep .value_required:n = true,
1096     columns .int_set:c = { l__enumext_columns_#2_int },
1097     columns .initial:n = 1,
1098     columns .value_required:n = true,
1099   }
1100 }
1101 \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.

```

1102 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1103 {
1104   \keys_define:nn { enumext / #1 }
1105   {
1106     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1107     mini-right .value_required:n = true,
1108     mini-right* .code:n = {
1109       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1110       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1111     },
1112     mini-right* .value_required:n = true,
1113   }
1114 }
1115 \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 multicols

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

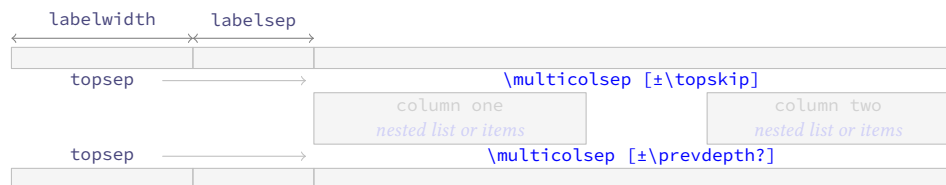


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

🔗 I should make it clear that the implementation here is a “bit questionable”. At first glance doing `\multicolsep=\topsep` seemed right, but the results were not always as expected. An almost *imperceptible* detail is that in some cases the `\itemsep` values of are “stretched”, possibly due to the use of `\raggedcolumns` and this affects the lower space when closing the environment, which is “smaller” than expected. My attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

12.21.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

1116 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1117 {
1118   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1119   {
1120     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1121   }
1122   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1123   {
1124     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1125   }

```



```

1126   \__enumext_add_pre_parsep:
1127   }

```

(End of definition for __enumext_multi_set_vskip:.)

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

```

1128 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1129 {
1130   \int_case:nn { \l__enumext_level_int }
1131   {
1132     { 2 }{
1133       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1134       {
1135         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1136         {
1137           \l__enumext_parsep_i_skip
1138         }
1139       }
1140     }
1141     { 3 }{
1142       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1143       {
1144         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1145         {
1146           \l__enumext_parsep_ii_skip
1147         }
1148       }
1149     }
1150     { 4 }{
1151       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1152       {
1153         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1154         {
1155           \l__enumext_parsep_iii_skip
1156         }
1157       }
1158     }
1159   }
1160 }

```

(End of definition for __enumext_add_pre_parsep:.)

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

```

1161 \cs_new_protected:Nn \__enumext_multi_addvspace:
1162 {
1163   \__enumext_multi_set_vskip:
1164   \mode_if_vertical:T
1165   {
1166     \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1167     {
1168       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1169     }
1170     \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1171     {
1172       \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1173     }
1174   }
1175   %% \__enumext_unskip_unkern: % remove for pdflatex-dev
1176   \par\nopagebreak
1177   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1178 }

```

(End of definition for __enumext_multi_addvspace:.)

12.21.2 Adjustment of vertical spaces for multicol in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

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

```

1179 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1180 {
1181   \skip_set:Nn \l__enumext_multicols_above_v_skip
1182   {
1183     \l__enumext_topsep_v_skip
1184   }
1185   \skip_set:Nn \l__enumext_multicols_below_v_skip
1186   {
1187     \l__enumext_topsep_v_skip
1188   }
1189 }
1190 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1191 {
1192   \__enumext_keyans_multi_set_vskip:
1193   \mode_if_vertical:T
1194   {
1195     \skip_add:Nn \l__enumext_multicols_above_v_skip
1196     {
1197       \skip_use:N \l__enumext_partopsep_v_skip
1198     }
1199     \skip_add:Nn \l__enumext_multicols_below_v_skip
1200     {
1201       \skip_use:N \l__enumext_partopsep_v_skip
1202     }
1203   }
1204   %% \__enumext_unskip_unkern: % remove for pdflatex-dev
1205   \par\nopagebreak
1206   \addvspace{ \l__enumext_multicols_above_v_skip }
1207 }

```

(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 `multicol` 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 `multicol` 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`.

```

1208 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1209 {

```

```

1210 \skip_set:Nn \l__enumext_minipage_right_skip
1211 {
1212   \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1213 }
1214 \mode_if_vertical:T
1215 {
1216   \skip_add:Nn \l__enumext_minipage_right_skip
1217   {
1218     \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1219   }
1220 }
1221 \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:`.

```

1222 \skip_set_eq:cN
1223 { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1224 \skip_set_eq:cN
1225 { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1226 \__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`.

```

1227 \int_compare:nNnT
1228 { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1229 {
1230   \skip_zero:N \topskip
1231   \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1232 }
1233 }

```

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*.

```

1234 \cs_new_protected:Nn \__enumext_minipage_add_space:
1235 {
1236   \__enumext_minipage_set_skip:
1237   \__enumext_unskip_unkern:
1238   \mode_if_vertical:TF
1239   {
1240     \nopagebreak\nointerlineskip
1241   }
1242   {
1243     \par\nopagebreak\nointerlineskip
1244     \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1245   }
1246   \int_compare:nNnTF
1247   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1248   {
1249     \addvspace{ 0.445\box_ht:N \strutbox }
1250   }
1251   {
1252     \addvspace{ 0.250\box_ht:N \strutbox }
1253   }
1254 }

```

(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.

```

1255 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1256 {
1257   \int_case:nn { \l__enumext_level_int }
1258   {
1259     { 2 }{
1260       \skip_if_eq:nnTF
1261       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1262       {
1263         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1264       }
1265     }
1266   }

```

```

1264         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1265     }
1266     {
1267         \dim_compare:nNnT
1268         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1269         {
1270             \skip_sub:Nn
1271             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1272             \skip_sub:Nn
1273             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1274             \skip_add:Nn
1275             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1276             \skip_add:Nn
1277             \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1278         }
1279         \dim_compare:nNnT
1280         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1281         {
1282             \skip_set:Nn \l__enumext_minipage_temp_skip
1283             {
1284                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1285             }
1286             \skip_sub:Nn
1287             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1288             \skip_sub:Nn
1289             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1290             \skip_add:Nn
1291             \l__enumext_minipage_after_skip
1292             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1293             \skip_add:Nn
1294             \l__enumext_multicols_below_ii_skip
1295             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1296         }
1297     }
1298 }
1299 { 3 }{
1300 \skip_if_eq:nNnTF
1301 { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1302 {
1303     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1304     \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1305 }
1306 {
1307     \dim_compare:nNnT
1308     { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1309     {
1310         \skip_sub:Nn
1311         \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1312         \skip_sub:Nn
1313         \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1314         \skip_add:Nn
1315         \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1316         \skip_add:Nn
1317         \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1318     }
1319     \dim_compare:nNnT
1320     { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1321     {
1322         \skip_set:Nn \l__enumext_minipage_temp_skip
1323         {
1324             \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1325         }
1326         \skip_sub:Nn
1327         \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1328         \skip_sub:Nn
1329         \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1330         \skip_add:Nn
1331         \l__enumext_minipage_after_skip
1332         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1333         \skip_add:Nn
1334         \l__enumext_multicols_below_iii_skip

```

```

1335         { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1336     }
1337 }
1338 }
1339 { 4 }{
1340     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1341     {
1342         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1343         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1344     }
1345     {
1346         \dim_compare:nNnT
1347         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1348         {
1349             \skip_sub:Nn
1350             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1351             \skip_sub:Nn
1352             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1353             \skip_add:Nn
1354             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1355             \skip_add:Nn
1356             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1357         }
1358         \dim_compare:nNnT
1359         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1360         {
1361             \skip_set:Nn \l__enumext_minipage_temp_skip
1362             {
1363                 \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1364             }
1365             \skip_sub:Nn
1366             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1367             \skip_sub:Nn
1368             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1369             \skip_add:Nn
1370             \l__enumext_minipage_after_skip
1371             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1372             \skip_add:Nn
1373             \l__enumext_multicols_below_iv_skip
1374             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1375         }
1376     }
1377 }
1378 }
1379 }

```

(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`.

```

1380 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1381 {
1382     \skip_zero:N \l__enumext_minipage_after_skip
1383     \skip_zero:N \l__enumext_minipage_left_skip
1384     \skip_zero:N \l__enumext_minipage_right_skip
1385     \skip_set:Nn \l__enumext_minipage_right_skip
1386     {
1387         \l__enumext_topsep_v_skip
1388     }
1389     \mode_if_vertical:T
1390     {
1391         \skip_add:Nn \l__enumext_minipage_right_skip
1392         {
1393             \l__enumext_partopsep_v_skip
1394         }
1395     }
1396     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1397     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1398     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip

```

```

1399   \__enumext_keyans_pre_itemsep_skip:
1400   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1401   {
1402     \skip_zero:N \topskip
1403     \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1404   }
1405 }
1406 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1407 {
1408   \__enumext_keyans_minipage_set_skip:
1409   \__enumext_unskip_unkern:
1410   \mode_if_vertical:TF
1411   {
1412     \nopagebreak\nointerlineskip
1413   }
1414   {
1415     \par\nopagebreak\nointerlineskip
1416     \skip_zero:N \l__enumext_partopsep_v_skip
1417   }
1418   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1419   {
1420     \addvspace{ 0.445\box_ht:N \strutbox }
1421   }
1422   {
1423     \addvspace{ 0.250\box_ht:N \strutbox }
1424   }
1425 }
1426 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1427 {
1428   \skip_if_eq:nnTF
1429   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1430   {
1431     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1432     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1433   }
1434   {
1435     \dim_compare:nNnT
1436     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1437     {
1438       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1439       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1440       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1441       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1442     }
1443     \dim_compare:nNnT
1444     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1445     {
1446       \skip_set:Nn \l__enumext_minipage_temp_skip
1447       {
1448         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1449       }
1450       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1451       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1452       \skip_add:Nn \l__enumext_minipage_after_skip
1453       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1454       \skip_add:Nn \l__enumext_multicols_below_v_skip
1455       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1456     }
1457   }
1458 }

```

(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*`.

```

1459 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1460 {
1461   \skip_zero_new:N \l__enumext_minipage_left_skip

```



```

1462 \skip_gzero_new:N \g__enumext_minipage_right_skip
1463 \skip_gzero_new:N \g__enumext_minipage_after_skip
1464 \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1465 {
1466   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1467   \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1468 }
1469 {
1470   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1471   \skip_gset:Nn \g__enumext_minipage_right_skip
1472   {
1473     \l__enumext_topsep_vii_skip
1474   }
1475   \skip_gset:Nn \g__enumext_minipage_after_skip
1476   {
1477     0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1478   }
1479 }
1480 }
1481 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1482 {
1483   \skip_zero_new:N \l__enumext_minipage_after_skip
1484   \skip_zero_new:N \l__enumext_minipage_left_skip
1485   \skip_zero_new:N \l__enumext_minipage_right_skip
1486   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1487   {
1488     \skip_set:Nn \l__enumext_minipage_left_skip
1489     {
1490       0.5\box_dp:N \strutbox
1491     }
1492     \skip_set:Nn \l__enumext_minipage_right_skip
1493     {
1494       \l__enumext_partopsep_viii_skip
1495     }
1496     \skip_set:Nn \l__enumext_minipage_after_skip
1497     {
1498       1.6\box_dp:N \strutbox
1499     }
1500   }
1501   {
1502     \skip_set:Nn \l__enumext_minipage_left_skip
1503     {
1504       0.5875\box_dp:N \strutbox
1505     }
1506     \skip_set:Nn \l__enumext_minipage_right_skip
1507     {
1508       \l__enumext_topsep_viii_skip
1509     }
1510     \skip_set:Nn \l__enumext_minipage_after_skip
1511     {
1512       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1513     }
1514   }
1515 }

```

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

__enumext_mini_addvspace_vii:
 __enumext_mini_addvspace_viii:

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

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

```

1516 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1517 {
1518   \__enumext_mini_set_vskip_vii:
1519   \par\nopagebreak
1520   \addvspace { \l__enumext_minipage_left_skip }
1521 }
1522 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1523 {
1524   \__enumext_mini_set_vskip_viii:

```

```

1525     \par\nopagebreak
1526     \addvspace { \l__enumext_minipage_left_skip }
1527 }

```

(End of definition for \l__enumext_mini_addvspace_vii: and \l__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 L^AT_EX justification is maintained in the `__enumext_mini_page` on the “right side”.

`\miniright`

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

```

1528 \NewDocumentCommand \miniright { s }
1529 {
1530     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1531     {
1532         \msg_error:nnn { enumext } { wrong-miniright-place }
1533     }
1534     % outside
1535     \bool_lazy_and:nnT
1536     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1537     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1538     {
1539         \msg_error:nnn { enumext } { wrong-miniright-place }
1540     }
1541     % starred env
1542     \bool_if:NT \l__enumext_starred_bool
1543     {
1544         \msg_error:nnn { enumext } { wrong-miniright-starred }
1545     }
1546     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1547     {
1548         \__enumext_keyans_mini_right_cmd:n {#1}
1549     }
1550     { \__enumext_mini_right_cmd:n {#1} }
1551 }

```

(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.

```

1552 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1553 {
1554     \dim_compare:nNtF
1555     { \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } } > { \c_zero_dim }
1556     {
1557         \__enumext_multicols_stop:
1558         \int_compare:nNt
1559         { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } = { 1 }
1560         {
1561             \par\addvspace{ \l__enumext_minipage_after_skip }
1562         }
1563         \end__enumext_mini_page
1564         \hfill
1565         \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } }
1566         \par\nointerlineskip
1567         \addvspace { \l__enumext_minipage_right_skip }
1568         \bool_if:nF {#1}
1569         {
1570             \centering
1571         }
1572         \int_gzero:N \g__enumext_minipage_stat_int

```

```

1573     }
1574     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1575 % paranoia
1576 \RenewDocumentCommand \miniright { s }
1577 {
1578     \msg_error:nn { enumext } { many-miniright-used }
1579 }
1580 }

```

(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.

```

1581 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1582 {
1583     \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1584     {
1585         \__enumext_keyans_multicols_stop:
1586         \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1587         {
1588             \par\addvspace{ \__enumext_minipage_after_skip }
1589         }
1590         \end__enumext_mini_page
1591         \hfill
1592         \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1593         \par\nointerlineskip
1594         \addvspace { \__enumext_minipage_right_skip }
1595         \bool_if:nF {#1}
1596         {
1597             \centering
1598         }
1599         \int_gzero:N \g__enumext_minipage_stat_int
1600     }
1601     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1602 % paranoia
1603 \RenewDocumentCommand \miniright { s }
1604 {
1605     \msg_error:nn { enumext } { many-miniright-used }
1606 }
1607 }

```

(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*

below

below*

```

1608 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1609 {
1610     \keys_define:nn { enumext / #1 }
1611     {
1612         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1613         above .value_required:n = true,
1614         above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1615             \keys_set:nn { enumext / #1 } { above = {##1} },
1616         above* .value_required:n = true,
1617         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1618         below .value_required:n = true,
1619         below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1620             \keys_set:nn { enumext / #1 } { below = {##1} },
1621         below* .value_required:n = true,
1622     }
1623 }
1624 \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.

```

1625 \cs_new_protected:Nn \__enumext_vspace_above:
1626 {
1627   \skip_if_eq:nnF
1628   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1629   {
1630     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1631     {
1632       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1633     }
1634     {
1635       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1636     }
1637   }
1638 }

```

(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.

```

1639 \cs_new_protected:Nn \__enumext_vspace_below:
1640 {
1641   \skip_if_eq:nnF
1642   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1643   {
1644     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1645     {
1646       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1647     }
1648     {
1649       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1650     }
1651   }
1652 }

```

(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.

```

1653 \cs_new_protected:Nn \__enumext_vspace_above_v:
1654 {
1655   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1656   {
1657     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1658     {
1659       \vspace*{ \l__enumext_vspace_above_v_skip }
1660     }
1661     { \vspace { \l__enumext_vspace_above_v_skip } }
1662   }
1663 }

```

(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.

```

1664 \cs_new_protected:Nn \__enumext_vspace_below_v:
1665 {
1666   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1667   {
1668     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1669     {
1670       \vspace*{ \l__enumext_vspace_below_v_skip }
1671     }
1672     { \vspace { \l__enumext_vspace_below_v_skip } }
1673   }
1674 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

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

```

1675 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1676 {
1677   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1678   {
1679     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1680     {
1681       \vspace*{ \l__enumext_vspace_above_vii_skip }
1682     }
1683     { \vspace { \l__enumext_vspace_above_vii_skip } }
1684   }
1685 }
1686 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1687 {
1688   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1689   {
1690     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1691     {
1692       \vspace*{ \l__enumext_vspace_above_viii_skip }
1693     }
1694     { \vspace { \l__enumext_vspace_above_viii_skip } }
1695   }
1696 }

```

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

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

```

1697 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1698 {
1699   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1700   {
1701     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1702     {
1703       \vspace*{ \l__enumext_vspace_below_vii_skip }
1704     }
1705     { \vspace { \l__enumext_vspace_below_vii_skip } }
1706   }
1707 }
1708 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1709 {
1710   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1711   {
1712     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1713     {
1714       \vspace*{ \l__enumext_vspace_below_viii_skip }
1715     }
1716     { \vspace { \l__enumext_vspace_below_viii_skip } }
1717   }
1718 }

```

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

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

```

series
resume
resume*
1719 \cs_set_protected:Npn \__enumext_tmp:n #1
1720 {
1721   \keys_define:nn { enumext / #1 }
1722   {
1723     series .str_set:N = \l__enumext_series_str,
1724     series .value_required:n = true,
1725     resume .code:n = \__enumext_resume_series:n {##1},
1726     resume* .code:n = \__enumext_resume_starred:,

```

```

1727         resume* .value_forbidden:n = true,
1728     }
1729 }
1730 \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 *⟨keys⟩* we want to store where *{#1}* represents the *optional argument* passed to the environment.

```

1731 \cs_new:Npn \__enumext_filter_series:n #1
1732 {
1733     \use:e
1734     {
1735         \keyval_parse:NNn
1736         \__enumext_filter_series_key:n
1737         \__enumext_filter_series_pair:nn {#1}
1738     }
1739 }

```

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.

```

1740 \cs_new:Npn \__enumext_filter_series_key:n #1
1741 {
1742     \str_case:nnF {#1}
1743     {
1744         { resume } {} { resume* } {} { base-fix } {}
1745     }
1746     { , { \exp_not:n {#1} } }
1747 }

```

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.

```

1748 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1749 {
1750     \str_case:nnF {#1}
1751     {
1752         { series } {} { resume } {} { start } {}
1753         { start* } {} { save-ans } {} { save-key } {}
1754     }
1755     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
1756 }

```

(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).

```

1757 \cs_new_protected:Npn \__enumext_parse_series:n #1
1758 {
1759     \str_if_empty:NTF \l__enumext_series_str
1760     {
1761         \bool_if:NF \l__enumext_resume_active_bool
1762         {
1763             \__enumext_resume_last:n {#1}
1764         }
1765     }
1766     {
1767         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1768         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1769         { \__enumext_filter_series:n {#1} }
1770         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1771         {

```

```

1772         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1773     }
1774 }
1775 }

```

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. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the *series* key is not being used.

```

1776 \cs_new_protected:Npn \__enumext_resume_last:n #1
1777 {
1778     \bool_if:NT \l__enumext_standar_first_bool
1779     {
1780         \tl_gclear:N \g__enumext_standar_series_tl
1781         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1782     }
1783     \bool_if:NT \l__enumext_starred_first_bool
1784     {
1785         \tl_gclear:N \g__enumext_starred_series_tl
1786         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1787     }
1788 }

```

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

12.24.2 Internal function to save counter value

`__enumext_resume_save_counter:`

The `__enumext_resume_save_counter:` function will save the last counter value to `\g__enumext_series_⟨series name⟩_int` if the `series={⟨series name⟩}` key has been passed, to `\g__enumext_resume_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).

```

1789 \cs_new_protected:Npn \__enumext_resume_save_counter:
1790 {
1791     \bool_if:NT \g__enumext_standar_bool
1792     {
1793         \tl_if_empty:NF \l__enumext_series_str
1794         {
1795             \int_gset_eq:cN
1796             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1797         }
1798         \tl_if_empty:NTF \l__enumext_resume_name_tl
1799         {
1800             \str_if_empty:NT \l__enumext_series_str
1801             {
1802                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1803             }
1804         }
1805         {
1806             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1807             {
1808                 \int_gset_eq:cN
1809                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1810             }
1811         }
1812         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1813         {
1814             \int_gset_eq:cN
1815             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1816         }
1817     }
1818     \bool_if:NT \g__enumext_starred_bool
1819     {
1820         \tl_if_empty:NF \l__enumext_series_str
1821         {

```



```

1822         \int_gset_eq:cN
1823         { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1824     }
1825     \tl_if_empty:NTF \l__enumext_resume_name_tl
1826     {
1827         \str_if_empty:NT \l__enumext_series_str
1828         {
1829             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1830         }
1831     }
1832     {
1833         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1834         {
1835             \int_gset_eq:cN
1836             { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1837         }
1838     }
1839     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1840     {
1841         \int_gset_eq:cN
1842         { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1843     }
1844 }
1845 }

```

(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.

```

1846 \cs_new_protected:Npn \__enumext_resume_series:n #1
1847 {
1848     \tl_if_empty:NTF {#1}
1849     {
1850         \__enumext_resume_counter:n { }
1851     }
1852     {
1853         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1854         {
1855             \__enumext_resume_counter:n {#1}
1856             \bool_if:NT \g__enumext_standar_bool
1857             {
1858                 \keys_set:nv { enumext / level-1 }
1859                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1860             }
1861             \bool_if:NT \g__enumext_starred_bool
1862             {
1863                 \keys_set:nv { enumext / enumext* }
1864                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1865             }
1866         }
1867         {
1868             \bool_if:NT \g__enumext_standar_bool
1869             {
1870                 \msg_error:nnn { enumext } { unknown-series } {#1}
1871             }
1872             \bool_if:NT \g__enumext_starred_bool
1873             {
1874                 \msg_error:nnn { enumext } { unknown-series } {#1}
1875             }
1876         }
1877     }
1878 }

```

(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`.

```

1879 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1880 {
1881   \bool_set_true:N \l__enumext_resume_active_bool
1882   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1883   \tl_if_empty:NTF \l__enumext_resume_name_tl
1884   {
1885     \__enumext_resume_counter:
1886   }
1887   {
1888     \__enumext_resume_counter_series:
1889   }
1890   \__enumext_resume_counter_save_ans:
1891 }

```

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.

```

1892 \cs_new_protected:Nn \__enumext_resume_counter:
1893 {
1894   \bool_if:NT \g__enumext_standar_bool
1895   {
1896     \int_gincr:N \g__enumext_resume_int
1897     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1898   }
1899   \bool_if:NT \g__enumext_starred_bool
1900   {
1901     \int_gincr:N \g__enumext_resume_vii_int
1902     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1903   }
1904 }

```

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.

```

1905 \cs_new_protected:Nn \__enumext_resume_counter_series:
1906 {
1907   \bool_if:NT \g__enumext_standar_bool
1908   {
1909     \int_set:Nn \l__enumext_start_i_int
1910     {
1911       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1912     }
1913   }
1914   \bool_if:NT \g__enumext_starred_bool
1915   {
1916     \int_set:Nn \l__enumext_start_vii_int
1917     {
1918       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1919     }
1920   }
1921 }

```

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.

```

1922 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1923 {
1924   \bool_lazy_and:nnT
1925   { \bool_if_p:N \l__enumext_standar_first_bool }
1926   { \bool_if_p:N \l__enumext_store_active_bool }
1927   {
1928     \int_set:Nn \l__enumext_start_i_int
1929     {
1930       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1931     }

```

```

1932     }
1933     \bool_lazy_and:nnT
1934     { \bool_if_p:N \l__enumext_starred_first_bool }
1935     { \bool_if_p:N \l__enumext_store_active_bool }
1936     {
1937         \int_set:Nn \l__enumext_start_vii_int
1938         {
1939             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1940         }
1941     }
1942 }

```

(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.

```

1943 \cs_new_protected:Nn \__enumext_resume_starred:
1944 {
1945     \bool_if:NT \g__enumext_standar_bool
1946     {
1947         \tl_if_empty:NF \g__enumext_standar_series_tl
1948         {
1949             \__enumext_resume_counter:n { }
1950             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1951         }
1952     }
1953     \bool_if:NT \g__enumext_starred_bool
1954     {
1955         \tl_if_empty:NF \g__enumext_starred_series_tl
1956         {
1957             \__enumext_resume_counter:n { }
1958             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1959         }
1960     }
1961 }

```

(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*`.

```

1962 \cs_set_protected:Npn \__enumext_tmp:n #1
1963 {
1964     \keys_define:nn { enumext / #1 }
1965     {
1966         save-ans .code:n = \__enumext_storing_set:n {##1},
1967         save-ans .value_required:n = true,
1968     }
1969 }
1970 \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:`.

```

1971 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1972 {
1973     \msg_term:nnVV { enumext } { save-ans-log }
1974     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1975 }

```

```

1976 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1977 {
1978   \msg_term:nnVV { enumext } { save-ans-log-hook }
1979   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1980 }

```

(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.

```

1981 \cs_new_protected:Npn \__enumext_storing_set:n #1
1982 {
1983   \tl_set:Nx \l__enumext_store_name_tl {#1}
1984   \tl_if_empty:NTF \l__enumext_store_name_tl
1985   {
1986     \bool_lazy_or:nnT
1987     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1988     {
1989       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1990     }
1991   }
1992   {
1993     \bool_lazy_or:nnT
1994     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1995     {
1996       \__enumext_start_save_ans_msg:
1997       \__enumext_storing_exec:
1998     }
1999   }
2000 }

```

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).

```

2001 \cs_new_protected:Nn \__enumext_storing_exec:
2002 {
2003   \bool_set_true:N \l__enumext_store_active_bool
2004   \bool_set_true:N \l__enumext_check_answers_bool
2005   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2006   \__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*`.

```

2007   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2008   {
2009     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2010     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2011   }
2012   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2013   {
2014     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2015     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2016   }
2017   \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2018   {
2019     \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2020     \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2021   }
2022 }

```

(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 2023 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   2024 {
            2025   \keys_define:nn { enumext / #1 }
            2026   {
            2027     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
            2028     check-ans .initial:n = false,
            2029     check-ans .value_required:n = true,
            2030     no-store .code:n = {
            2031                                   \bool_set_false:N \__enumext_check_answers_bool
            2032                                   \bool_set_false:N \__enumext_check_ans_key_bool
            2033                                   },
            2034     no-store .value_forbidden:n = true,
            2035   }
            2036 }
2037 \clist_map_inline:nn
2038 {
2039   level-1, level-2, level-3, level-4, enumext*
2040 }
2041 { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

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

```

2042 \cs_new_protected:Nn \__enumext_check_ans_active:
2043 {
2044   \tl_if_empty:NF \l__enumext_store_name_tl
2045   {
2046     \bool_if:NT \l__enumext_check_answers_bool
2047     {
2048       \__enumext_check_ans_level:
2049     }
2050   }
2051 }
```

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”.

```

2052 \cs_new_protected:Nn \__enumext_check_ans_level:
2053 {
2054   \int_case:nn { \l__enumext_level_int }
2055   {
2056     { 1 }{
2057       \bool_lazy_all:nT
2058       {
2059         { \bool_if_p:N \g__enumext_starred_bool }
2060         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2061       }
2062       {
2063         \int_gdecr:N \g__enumext_item_number_int
2064         \bool_set_false:N \l__enumext_item_number_bool
2065       }
2066     }
2067     { 2 }{
2068       \int_gdecr:N \g__enumext_item_number_int
2069       \bool_set_false:N \l__enumext_item_number_bool
2070     }
2071     { 3 }{
2072       \int_gdecr:N \g__enumext_item_number_int
2073       \bool_set_false:N \l__enumext_item_number_bool
2074     }
2075     { 4 }{
2076       \int_gdecr:N \g__enumext_item_number_int
2077       \bool_set_false:N \l__enumext_item_number_bool
2078     }
2079   }

```

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.

```

2080   \int_case:nn { \l__enumext_level_h_int }
2081   {
2082     { 1 }{
2083       \bool_lazy_all:nT
2084       {
2085         { \bool_if_p:N \g__enumext_standar_bool }
2086         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2087       }
2088       {
2089         \int_gdecr:N \g__enumext_item_number_int
2090         \bool_set_false:N \l__enumext_item_number_bool
2091       }
2092     }
2093   }
2094 }

```

(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.

```

2095 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2096 {
2097   \bool_lazy_and:nnT
2098   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2099   { \bool_if_p:N \g__enumext_standar_bool }
2100   {
2101     \bool_gset_true:N \g__enumext_check_ans_key_bool
2102   }
2103   \bool_lazy_and:nnT
2104   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2105   { \bool_if_p:N \g__enumext_starred_bool }
2106   {
2107     \bool_gset_true:N \g__enumext_check_ans_key_bool
2108   }
2109 }

```

(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:`.

```
2110 \cs_new_protected:Nn \__enumext_item_answer_diff:
2111 {
2112     \int_gset:Nn \g__enumext_item_answer_diff_int
2113     {
2114         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2115     }
2116 }
```

(End of definition for `__enumext_item_answer_diff:`.)

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

```
2117 \cs_new_protected:Nn \__enumext_check_ans_show:
2118 {
2119     \int_case:nn { \g__enumext_item_answer_diff_int }
2120     {
2121         { -1 } { \__enumext_check_ans_msg_less: }
2122         { 0 } { \__enumext_check_ans_msg_same_ok: }
2123         { 1 } { \__enumext_check_ans_msg_greater: }
2124     }
2125 }
2126 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2127 {
2128     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2129     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2130 }
2131 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2132 {
2133     \msg_term:nnee { enumext } { items-same-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_msg_greater:
2137 {
2138     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2139     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2140 }
```

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

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

```
2141 \cs_new_protected:Nn \__enumext_check_ans_log:
2142 {
2143     \int_case:nn { \g__enumext_item_answer_diff_int }
2144     {
2145         { -1 } { \__enumext_check_ans_log_msg_less: }
2146         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2147         { 1 } { \__enumext_check_ans_log_msg_greater: }
2148     }
2149 }
2150 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2151 {
2152     \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2153     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2154 }
2155 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2156 {
2157     \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2158     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2159 }
```



```

2160 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2161 {
2162   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2163   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2164 }

```

(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.

```

2165 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2166 {
2167   \int_compare:nNt
2168   { \g__enumext_check_starred_cmd_int } = { 0 }
2169   {
2170     \msg_warning:nnnV
2171     { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2172   }
2173   \int_compare:nNt
2174   { \g__enumext_check_starred_cmd_int } > { 1 }
2175   {
2176     \msg_warning:nnnV
2177     { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2178   }
2179   \int_gzero:N \g__enumext_check_starred_cmd_int
2180   \tl_clear:N \l__enumext_check_start_line_env_tl
2181 }

```

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

12.26 Keys and functions associated with storage

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

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

```

2182 \cs_set_protected:Npn \__enumext_tmp:n #1
2183 {
2184   \keys_define:nn { enumext / #1 }
2185   {
2186     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2187     wrap-ans .initial:n =
2188       {
2189         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2190       },
2191     wrap-ans .value_required:n = true,
2192     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2193     wrap-opt .initial:n = [{##1}],
2194     wrap-opt .value_required:n = true,
2195     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2196     save-sep .initial:n = {, ~ },
2197     save-sep .value_required:n = true,
2198     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2199     mark-ans .initial:n = \textasteriskcentered,
2200     mark-ans .value_required:n = true,
2201     mark-pos .choice:,
2202     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2203     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2204     mark-pos / unknown .code:n =
2205       \msg_error:nneee { enumext } { unknown-choice }
2206       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2207     mark-pos .initial:n = right,
2208     mark-pos .value_required:n = true,
2209     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2210     show-ans .initial:n = false,
2211     show-ans .value_required:n = true,
2212     show-pos .bool_set:N = \l__enumext_show_position_bool,
2213     show-pos .initial:n = false,
2214     show-pos .value_required:n = true,
2215     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,

```

```

2216         mark-ref .initial:n = \textasteriskcentered,
2217         mark-ref .value_required:n = true,
2218         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2219         save-ref .initial:n = false,
2220         save-ref .value_required:n = true,
2221     }
2222 }
2223 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

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

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

```

2224 \cs_set_protected:Npn \l__enumext_tmp:n #1
2225 {
2226     \keys_define:nn { enumext / #1 }
2227     {
2228         mark-pos .choice:,
2229         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2230         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2231         mark-pos .initial:n = right,
2232         mark-pos .value_required:n = true,
2233         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2234         show-ans .initial:n = false,
2235         show-ans .value_required:n = true,
2236         show-pos .bool_set:N = \l__enumext_show_position_bool,
2237         show-pos .initial:n = false,
2238         show-pos .value_required:n = true,
2239     }
2240 }
2241 \clist_map_inline:nn { keyans, keyans* } { \l__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>*.

```

2242 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2243 {
2244     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2245     {
2246         \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2247         \tl_set:ce
2248             { \l__enumext_store_save_key_ \__enumext_level: _tl }
2249             { \__enumext_filter_save_key:n {#1} }
2250     }
2251 }
2252 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2253 {
2254     \bool_if:NF \l__enumext_store_save_key_vii_bool
2255     {
2256         \tl_clear:N \l__enumext_store_save_key_vii_tl
2257         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2258     }
2259 }

```

(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 `__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.

```
2260 \cs_set_protected:Npn \__enumext_tmp:n #1
2261 {
2262   \keys_define:nn { enumext / enumext* }
2263   {
2264     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2265     save-key .value_required:n = true,
2266   }
2267   \keys_define:nn { enumext / #1 }
2268   {
2269     save-key .code:n = \__enumext_parse_save_key:n {##1},
2270     save-key .value_required:n = true,
2271   }
2272 }
2273 \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 `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```
2274 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2275 {
2276   \bool_set_true:c { \__enumext_store_save_key_ \__enumext_level: _bool }
2277   \tl_clear:c { \__enumext_save_key_ \__enumext_level: _tl }
2278   \tl_set:ce
2279   { \__enumext_store_save_key_ \__enumext_level: _tl }
2280   { \__enumext_filter_save_key:n {#1} }
2281 }
2282 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2283 {
2284   \bool_set_true:N \__enumext_store_save_key_vii_bool
2285   \tl_clear:N \__enumext_store_save_key_vii_tl
2286   \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2287 }
```

(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

```
\__enumext_filter_save_key:n
  \__enumext_filter_save_key_key:n
  \__enumext_filter_save_key_pair:nn
```

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

```
2288 \cs_new:Npn \__enumext_filter_save_key:n #1
2289 {
2290   \use:e
2291   {
2292     \keyval_parse:NNn
2293     \__enumext_filter_save_key_key:n
2294     \__enumext_filter_save_key_pair:nn {#1}
2295   }
2296 }
```

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.

```
2297 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2298 {
2299   \str_case:nnF {#1}
2300   {
2301     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2302   }
2303   { , { \exp_not:n {#1} } }
2304 }
```

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.

```

2305 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2306 {
2307   \str_case:nnF {#1}
2308   {
2309     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2310     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2311     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2312     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2313     { mini-right* } {}
2314   }
2315   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2316 }

```

(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

`__enumext_store_addto_prop:n`
`__enumext_store_addto_prop:V`

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

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

```

2317 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2318 {
2319   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2320   {
2321     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2322   }
2323   { #1 }
2324 }
2325 \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`
`__enumext_store_addto_seq:v`
`__enumext_store_addto_seq:V`

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

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

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

```

2326 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2327 {
2328   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2329 }
2330 \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 the list structure in the sequence

`__enumext_store_level_open:`
`__enumext_store_level_close:`

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

```

2331 \cs_new_protected:Nn \__enumext_store_level_open:
2332 {
2333   \bool_if:NT \__enumext_check_answers_bool
2334   {
2335     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2336     {
2337       \__enumext_store_addto_seq:n
2338       {
2339         \item \begin{enumext}
2340       }
2341     }
2342     {
2343       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2344       {

```

```

2345         \item \begin{enumext} [
2346         ]
2347         \tl_put_right:cn { \__enumext_store_save_key_ \__enumext_level: _tl }
2348         {
2349         ]
2350         }
2351         \__enumext_store_addto_seq:v { \__enumext_store_save_key_ \__enumext_level: _tl }
2352     }
2353 }
2354 }
2355 \cs_new_protected:Nn \__enumext_store_level_close:
2356 {
2357     \bool_if:NT \l__enumext_check_answers_bool
2358     {
2359         \__enumext_store_addto_seq:n { \end{enumext} }
2360     }
2361 }

```

(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.

```

2362 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2363 {
2364     \bool_if:NT \l__enumext_check_answers_bool
2365     {
2366         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2367         {
2368             \__enumext_store_addto_seq:n
2369             {
2370                 \item \begin{enumext*}
2371             }
2372         }
2373         {
2374             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2375             {
2376                 \item \begin{enumext*}[
2377             }
2378             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2379             {
2380             ]
2381             }
2382             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2383         }
2384     }
2385 }
2386 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2387 {
2388     \bool_if:NT \l__enumext_check_answers_bool
2389     {
2390         \__enumext_store_addto_seq:n { \end{enumext*} }
2391     }
2392 }

```

(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 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

```

2393 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2394 {
2395     \mode_leave_vertical:
2396     \skip_horizontal:n { -\dim_use:N #2 }
2397     \makebox[0pt][ r ]
2398     {
2399         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2400         {
2401             \tl_use:N \l__enumext_mark_answer_sym_tl

```

```

2402     }
2403   }
2404   \skip_horizontal:n { \dim_use:N #2 }
2405 }
2406 \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`.

```

2407 \cs_new_protected:Nn \__enumext_store_internal_ref:
2408 {
2409   \cs_set_protected:Npn \__enumext_tmp:n ##1
2410   {
2411     \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2412     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2413     \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2414     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2415   }
2416   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2417   \cs_set:Npn \__enumext_tmp:n ##1
2418   { . \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.

```

2419   \bool_lazy_all:nT
2420   {
2421     { \bool_if_p:N \g__enumext_starred_bool }
2422     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2423   }
2424   {
2425     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2426     { \tl_use:N \l__enumext_label_copy_vii_tl }
2427   }
2428   \bool_lazy_all:nT
2429   {
2430     { \bool_not_p:n { \g__enumext_standar_bool } }
2431     { \bool_if_p:N \l__enumext_standar_bool }
2432     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2433   }
2434   {
2435     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2436     {
2437       \tl_use:N \l__enumext_label_copy_vii_tl
2438       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2439     }
2440   }

```

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.

```

2441   \bool_lazy_all:nT
2442   {
2443     { \bool_if_p:N \g__enumext_standar_bool }
2444     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2445     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2446   }
2447   {
2448     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2449     {
2450       \tl_use:N \l__enumext_label_copy_i_tl
2451       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2452     }
2453   }
2454   \cs_set:Npn \__enumext_tmp:n ##1
2455   { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2456   \bool_lazy_all:nT

```

```

2457     {
2458       { \bool_if_p:N \g__enumext_standar_bool }
2459       { \bool_if_p:N \l__enumext_starred_bool }
2460       { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2461     }
2462     {
2463       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2464       {
2465         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2466         \tl_use:N \l__enumext_label_copy_vii_tl
2467       }
2468     }

```

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

```

2469     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2470     {
2471       \l__enumext_store_name_tl \c_colon_str
2472       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2473     }

```

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

```

2474     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2475     {
2476       \__enumext_newlabel:nn
2477       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2478       { \l__enumext_newlabel_arg_two_tl }
2479     }
2480     \l__enumext_write_aux_file_tl
2481   }

```

(End of definition for `__enumext_store_internal_ref:`)

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

`__enumext_store_anskey_code:n`

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

```

2482 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2483 {
2484   \int_gincr:N \g__enumext_item_anskey_int
2485   \__enumext_store_addto_prop:n {#1}
2486   \bool_if:NT \l__enumext_store_ref_key_bool
2487   {
2488     \__enumext_store_internal_ref:
2489   }
2490   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2491   \tl_clear:N \l__enumext_store_anskey_arg_tl
2492   \bool_lazy_and:nnT
2493   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2494   { \bool_not_p:n { \l__enumext_starred_bool } }
2495   {
2496     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2497   }
2498   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2499   \bool_lazy_and:nnT
2500   { \bool_not_p:n { \l__enumext_starred_bool } }
2501   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2502   {
2503     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2504     {
2505       ( \exp_not:V \l__enumext_store_item_join_int )
2506     }
2507   }

```


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

```

2508 \bool_if:NTF \l__enumext_store_item_star_bool
2509 {
2510   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2511   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2512   {
2513     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2514     {
2515       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2516     }
2517   }
2518   \dim_compare:nT
2519   {
2520     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2521   }
2522   {
2523     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2524     {
2525       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2526     }
2527   }
2528   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2529 }
2530 {
2531   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2532 }

```

Finally we check if the `save-ref` key are active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with “*symbol*” set by `mark-ref` key and then store in *sequence*.

```

2533 \bool_lazy_and:nnT
2534 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2535 { \bool_if_p:N \l__enumext_hyperref_bool }
2536 {
2537   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2538   {
2539     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2540     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2541   }
2542 }
2543 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2544 }

```

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

`\l__enumext_anskey_show_wrap_arg:n`

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

```

2545 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_arg:n #1
2546 {
2547   \par
2548   \bool_if:NTF \l__enumext_starred_bool
2549   {
2550     \l__enumext_print_keyans_box:NN
2551     \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2552   }
2553   {
2554     \l__enumext_print_keyans_box:cc
2555     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2556     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2557   }
2558   \l__enumext_anskey_wrapper:n { #1 }
2559 }

```

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

`\l__enumext_anskey_show_wrap_left:n`

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

```

2560 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_left:n #1
2561 {

```

```

2562 \bool_if:NT \l__enumext_show_answer_bool
2563 {
2564   \__enumext_anskey_show_wrap_arg:n { #1 }
2565 }
2566 \bool_if:NT \l__enumext_show_position_bool
2567 {
2568   \tl_set:Nx \l__enumext_mark_answer_sym_tl
2569   {
2570     \group_begin:
2571     \exp_not:N \normalfont
2572     \exp_not:N \footnotesize [ \int_eval:n
2573       {
2574         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2575       }
2576     ]
2577     \group_end:
2578   }
2579   \__enumext_anskey_show_wrap_arg:n { #1 }
2580 }
2581 }

```

(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[⟨key = val⟩]{⟨content⟩}`.

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

```

\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:nn
2582 \keys_define:nn { enumext / anskey }
2583 {
2584   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2585   break-col .default:n = true,
2586   break-col .value_forbidden:n = true,
2587   item-join .int_set:N = \l__enumext_store_item_join_int,
2588   item-join .value_required:n = true,
2589   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2590   item-star .default:n = true,
2591   item-star .value_forbidden:n = true,
2592   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2593   item-sym* .value_required:n = true,
2594   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2595   item-pos* .value_required:n = true,
2596   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2597 }

```

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

```

2598 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2599 {
2600   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2601 }
2602 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2603 {
2604   \tl_if_blank:nTF {#2}
2605   {
2606     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2607   }
2608   {
2609     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2610   }
2611 }

```

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

- 🔗 The `\anskey` command will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error.

\anskey We will first call the function `__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and

execute the function `__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[⟨key = val⟩]` and call the function `__enumext_store_anskey_code:n`.

```

2612 \NewDocumentCommand \anskey { o +m }
2613 {
2614   \__enumext_anskey_safe_outer:
2615   \group_begin:
2616     \bool_if:NT \l__enumext_check_answers_bool
2617     {
2618       \tl_if_novalue:nF {#1}
2619       {
2620         \keys_set:nn { enumext / anskey } {#1}
2621       }
2622       \tl_if_blank:nTF {#2}
2623       {
2624         \msg_error:nn { enumext } { anskey-empty-arg }
2625       }
2626       {
2627         \__enumext_anskey_safe_inner:
2628         \__enumext_store_anskey_code:n {#2}
2629       }
2630     }
2631   \group_end:
2632 }

```

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

12.29.1 Internal functions for the command

`__enumext_anskey_safe_outer:`
`__enumext_anskey_safe_inner:`

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

```

2633 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2634 {
2635   \bool_if:NF \l__enumext_store_active_bool
2636   {
2637     \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2638   }
2639   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2640   {
2641     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2642   }
2643   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2644   {
2645     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2646   }
2647   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2648   {
2649     \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2650   }
2651 }

```

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.

```

2652 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2653 {
2654   \int_incr:N \l__enumext_anskey_level_int
2655   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2656   {
2657     \msg_error:nn { enumext } { anskey-nested }
2658   }
2659   \bool_if:NF \l__enumext_item_number_bool
2660   {
2661     \msg_error:nn { enumext } { anskey-unnumber-item }
2662   }
2663   \mode_if_math:T
2664   {
2665     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2666   }
2667 }

```

(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.

```
2668 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2669 {
2670   \cs_undefine:c { anskey* }
2671   \cs_undefine:c { endanskey* }
2672   \cs_undefine:c { __scontents_anskey*_env_begin: }
2673   \cs_undefine:c { __scontents_anskey*_env_end: }
2674 }
```

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

```
2675 \__enumext_before_env:nn { enumext }
2676 {
2677   \bool_lazy_and:nnT
2678     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2679     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2680     {
2681       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2682       {
2683         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2684       }
2685     }
2686 }
2687 \__enumext_before_env:nn { enumext* }
2688 {
2689   \bool_lazy_and:nnT
2690     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2691     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2692     {
2693       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2694       {
2695         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2696       }
2697     }
2698 }
```

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.

```
2699 \__enumext_before_env:nn { anskey* }
2700 {
2701   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2702   {
2703     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2704   }
2705   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2706   {
2707     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2708   }
2709   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2710   {
2711     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2712   }
2713   \bool_if:NF \__enumext_item_number_bool
2714   {
```

```

2715         \msg_error:nn { enumext } { anskey-unnumber-item }
2716     }
2717     \mode_if_math:T
2718     {
2719         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2720     }
2721 }

```

(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 `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package **scontents**.

```

2722 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2723 {
2724     \bool_if:NT \l__enumext_anskey_env_bool
2725     {
2726         \newenvsc{anskey*}[store-env=#1,print-env=false]
2727         \__enumext_anskey_env_exec:
2728     }
2729 }
2730 \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`.

```

2731 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2732 {
2733     \keys_define:nn { scontents / scontents }
2734     {
2735         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2736         break-col .default:n   = true,
2737         break-col .value_forbidden:n = true,
2738         item-join .int_gset:N   = \g__enumext_store_item_join_int,
2739         item-join .value_required:n = true,
2740         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2741         item-star .default:n    = true,
2742         item-star .value_forbidden:n = true,
2743         item-sym* .tl_gset:N    = \g__enumext_store_item_symbol_tl,
2744         item-sym* .value_required:n = true,
2745         item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2746         item-pos* .value_required:n = true,
2747         print-env .undefine:,
2748         store-env .undefine:,
2749         write-out .undefine:,
2750         unknown .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2751     }
2752 }

```

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`.

```

2753 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2754 {
2755     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2756 }
2757 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2758 {
2759     \tl_if_blank:nTF {#2}
2760     {
2761         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2762     }
2763     {
2764         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2765     }
2766 }

```

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`.

```

2767 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2768 {
2769   \keys_define:nn { scontents / scontents }
2770   {
2771     break-col .undefine:,
2772     item-join .undefine:,
2773     item-star .undefine:,
2774     item-sym* .undefine:,
2775     item-pos* .undefine:,
2776     write-out .code:n = {
2777       \bool_set_false:N \l__scontents_storing_bool
2778       \bool_set_true:N \l__scontents_writing_bool
2779       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2780     },
2781     write-out .value_required:n = true,
2782     print-env .meta:nn = { scontents } { print-env = ##1 },
2783     print-env .default:n = true,
2784     store-env .meta:nn = { scontents } { store-env = ##1 },
2785     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2786   }
2787 }

```

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*.

```

2788 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2789 {
2790   \group_begin:
2791   \int_set:Nn \tex_newlinechar:D { `^^J }
2792   \__scontents_rescan_tokens:x
2793   {
2794     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2795     #1
2796   }
2797 }

```

(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)*.

```

2798 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2799 {
2800   \__enumext_before_env:nn { anskey* }
2801   {
2802     \__enumext_anskey_env_define_keys:
2803   }

```

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.

```

2804   \hook_if_empty:nF {env/anskey*/after}
2805   {
2806     \hook_gremove_code:nn {env/anskey*/after} { * }
2807   }
2808   \__enumext_after_env:nn { anskey* }
2809   {
2810     \__enumext_anskey_env_save_keys:
2811     \tl_clear:N \l__enumext_store_anskey_env_tl
2812     \tl_clear:N \l__enumext_store_anskey_opt_tl
2813     \bool_if:NT \l__enumext_check_answers_bool
2814     {
2815       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2816       {
2817         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2818       }
2819       \regex_match:nVTF
2820       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2821       \l__enumext_store_anskey_env_tl

```

```

2822         {
2823             \msg_error:nn { enumext } { anskey-empty-arg }
2824         }
2825         {
2826             \__enumext_anskey_env_store:
2827         }
2828     }
2829     \__enumext_anskey_env_clean_vars:
2830     \__enumext_anskey_env_reset_keys:
2831 }
2832 }

```

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

(End of definition for `__enumext_anskey_env_exec:.`)

```

\__enumext_anskey_env_save_keys:
\__enumext_anskey_env_store:
\__enumext_anskey_env_clean_vars:

```

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

```

2833 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2834 {
2835     \bool_lazy_and:nnT
2836     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2837     { \bool_not_p:n { \l__enumext_starred_bool } }
2838     {
2839         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2840     }

```

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

```

2841     \bool_lazy_and:nnT
2842     { \bool_not_p:n { \l__enumext_starred_bool } }
2843     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2844     {
2845         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2846         {
2847             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2848         }
2849     }

```

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

```

2850     \bool_if:NT \g__enumext_store_item_star_bool
2851     {
2852         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2853         {
2854             ,item-star,
2855         }
2856         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2857         {
2858             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2859             {
2860                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2861             }
2862         }
2863         \dim_compare:nT
2864         {
2865             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2866         }
2867         {
2868             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2869             {
2870                 ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2871             }
2872         }
2873     }
2874 }

```

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`.

```

2875 \cs_new_protected:Nn \__enumext_anskey_env_store:

```



```

2876 {
2877     \group_begin:
2878     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2879     {
2880         \exp_args:Ne
2881         \__enumext_store_anskey_code:n
2882         {
2883             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2884         }
2885     }
2886     {
2887         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2888         \exp_args:Ne
2889         \__enumext_store_anskey_code:n
2890         {
2891             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2892         }
2893     }
2894     \group_end:
2895 }

```

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

```

2896 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2897 {
2898     \bool_gset_false:N \g__enumext_store_columns_break_bool
2899     \int_gzero:N \g__enumext_store_item_join_int
2900     \bool_gset_false:N \g__enumext_store_item_star_bool
2901     \tl_gclear:N \g__enumext_store_item_symbol_tl
2902     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2903 }

```

(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.

```

2904 \cs_new_protected:Nn \__enumext_execute_after_env:
2905 {
2906     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
2907     {
2908         \tl_if_empty:NF \g__enumext_store_name_tl
2909         {
2910             \__enumext_stop_save_ans_msg:
2911             \__enumext_item_answer_diff:
2912             \__enumext_log_global_vars:
2913             \__enumext_log_answer_vars:
2914             \bool_if:NTF \g__enumext_check_ans_key_bool
2915             {
2916                 \__enumext_check_ans_show:
2917             }
2918             { \__enumext_check_ans_log: }
2919             \__enumext_undefine_anskey_env:
2920         }
2921         \__enumext_reset_global_vars:
2922     }
2923 }

```

(End of definition for `__enumext_execute_after_env:`.)

- This function is passed to the function `__enumext_after_env:nn` 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 $\langle label \rangle$ for $\backslash item^*$ in `keyans` environment and the current $\langle label \rangle$ for $\backslash anspic^*$ in `keyanspic` environment followed by the $\langle contents \rangle$ of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `__enumext_store_addto_prop:V`.

```

2924 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2925 {
2926   \tl_clear:N \l__enumext_store_current_label_tl
2927   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2928   {
2929     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2930   }
2931   {
2932     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2933   }

```

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

```

2934   \tl_if_novalue:nF { #1 }
2935   {
2936     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2937     {
2938       \tl_put_right:Ne \l__enumext_store_current_label_tl
2939       {
2940         \l__enumext_store_keyans_item_opt_sep_tl
2941       }
2942     }
2943     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2944   }
2945   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2946 }

```

(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 $\langle label \rangle$ for $\backslash item^*$ and $\backslash anspic^*$ with the $\langle contents \rangle$ of the *optional argument*. The mechanism defined here will allow to execute `\ref{⟨store name : position⟩}` and will return `1.` (A).

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`\enumext_keyans_store_ref_aux_ii:`

The function `__enumext_keyans_store_ref:` handles the “*internal label and ref*” system used by the `save-ref` key for $\backslash item^*$ and $\backslash anspic^*$ commands. First we will create copies of the current $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in references.

```

2947 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2948 {
2949   \bool_if:NT \l__enumext_store_ref_key_bool
2950   {
2951     \cs_set_protected:Npn \__enumext_tmp:n ##1
2952     {
2953       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2954       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2955       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2956       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2957     }
2958     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2959     \__enumext_keyans_store_ref_aux_i:
2960   }
2961 }

```

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

```

2962 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2963 {
2964   \bool_if:NT \g__enumext_starred_bool
2965   {
2966     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2967   }
2968   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

2969     {
2970       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2971       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2972     }
2973     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2974     {
2975       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2976       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2977     }
2978     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2979     {
2980       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2981       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2982     }
2983     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2984     {
2985       \l__enumext_store_name_tl \c_colon_str
2986       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2987     }
2988     \__enumext_keyans_store_ref_aux_ii:
2989   }

```

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.

```

2990 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2991 {
2992   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2993   {
2994     \__enumext_newlabel:nn
2995     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2996     { \l__enumext_newlabel_arg_two_tl }
2997   }
2998   \l__enumext_write_aux_file_tl
2999 }

```

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

12.32.3 Storing content in sequence

`__enumext_keyans_addto_seq:n`
`__enumext_keyans_addto_seq_link:`

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

```

3000 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3001 {
3002   \tl_clear:N \l__enumext_store_current_label_tl
3003   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
3004   {
3005     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3006   }
3007   {
3008     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3009   }
3010   \tl_if_no_value:nF { #1 }
3011   {
3012     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3013     {
3014       \tl_put_right:Ne \l__enumext_store_current_label_tl
3015       {
3016         \l__enumext_store_keyans_item_opt_sep_tl
3017       }
3018     }
3019     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3020   }
3021   \__enumext_keyans_addto_seq_link:
3022 }

```

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.

```

3023 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3024 {
3025   \bool_lazy_and:nnT
3026     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3027     { \bool_if_p:N \l__enumext_hyperref_bool }
3028   {
3029     \tl_put_right:Ne \l__enumext_store_current_label_tl
3030     {
3031       \hfill \exp_not:N \hyperlink
3032       {
3033         \exp_not:V \l__enumext_newlabel_arg_one_tl
3034       }
3035       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3036     }
3037   }
3038   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3039   \bool_if:NT \l__enumext_check_answers_bool
3040   {
3041     \int_gincr:N \g__enumext_item_anskey_int
3042   }
3043 }

```

(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.

```

3044 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3045 {
3046   \tl_if_novalue:nF { #1 }
3047   {
3048     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3049   }
3050   \bool_if:NT \l__enumext_show_answer_bool
3051   {
3052     \__enumext_keyans_show_ans:
3053   }
3054   \bool_if:NT \l__enumext_show_position_bool
3055   {
3056     \__enumext_keyans_show_pos:
3057   }
3058 }
3059 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3060 {
3061   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3062   {
3063     \bool_lazy_or:nnT
3064       { \bool_if_p:N \l__enumext_show_answer_bool }
3065       { \bool_if_p:N \l__enumext_show_position_bool }
3066     {
3067       \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3068     }
3069   }
3070 }
3071 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3072 {
3073   \bool_if:NT \l__enumext_starred_bool
3074   {
3075     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3076     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3077   }
3078   \tl_put_left:Nn \l__enumext_label_v_tl
3079   {
3080     \__enumext_print_keyans_box:NN
3081     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3082   }

```

```

3083   }
3084   \cs_new_protected:Nn \__enumext_keyans_show_pos:
3085   {
3086     \bool_if:NT \l__enumext_starred_bool
3087     {
3088       \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3089       \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3090     }
3091     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3092     {
3093       \tl_set:Ne \l__enumext_mark_answer_sym_tl
3094       {
3095         \group_begin:
3096         \exp_not:N \normalfont
3097         \exp_not:N \footnotesize [ \int_eval:n
3098           {
3099             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3100           }
3101         ]
3102         \group_end:
3103       }
3104     }
3105     {
3106       \tl_set:Ne \l__enumext_mark_answer_sym_tl
3107       {
3108         \group_begin:
3109         \exp_not:N \normalfont
3110         \exp_not:N \footnotesize [ \int_eval:n
3111           {
3112             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3113           }
3114         ]
3115         \group_end:
3116       }
3117     }
3118     \tl_put_left:Nn \l__enumext_label_v_tl
3119     {
3120       \__enumext_print_keyans_box:NN
3121       \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3122     }
3123   }

```

(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`.

```

3124 \cs_new_protected:Npn \__enumext_default_item:n #1
3125 {
3126   \tl_if_novalue:nTF {#1}
3127   {
3128     \bool_if:NT \l__enumext_check_answers_bool
3129     {
3130       \int_gincr:N \g__enumext_item_number_int
3131       \bool_set_true:N \l__enumext_item_number_bool
3132     }
3133     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3134     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3135   }
3136   {

```

```

3137         \bool_set_eq:cc
3138         { l__enumext_wrap_label_ \__enumext_level: _bool }
3139         { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3140         \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3141         }
3142     }

```

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

`__enumext_starred_item:nn`
`__enumext_item_star_exec:`

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

```

3143 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3144 {
3145     \tl_if_novalue:nTF {#1}
3146     {
3147         \tl_gset_eq:Nc
3148         \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3149     }
3150     {
3151         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3152     }
3153     \tl_if_novalue:nTF {#2}
3154     {
3155         \dim_set_eq:cc
3156         { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3157         { l__enumext_labelsep_ \__enumext_level: _dim }
3158     }
3159     {
3160         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3161     }
3162     \bool_if:NT \l__enumext_check_answers_bool
3163     {
3164         \int_gincr:N \g__enumext_item_number_int
3165         \bool_set_true:N \l__enumext_item_number_bool
3166     }
3167     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3168     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3169 }

```

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

```

3170 \cs_new_protected:Nn \__enumext_item_star_exec:
3171 {
3172     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3173     {
3174         \mode_leave_vertical:
3175         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3176         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3177         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3178     }
3179 }

```

(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).

```

3180 \cs_new_protected:Nn \__enumext_redefine_item:
3181 {
3182     \RenewDocumentCommand \item { s o o }
3183     {

```

```

3184         \bool_if:nTF {##1}
3185         {
3186             \__enumext_starred_item:nn {##2} {##3}
3187         }
3188         { \__enumext_default_item:n {##2} }
3189     }
3190 }

```

(End of definition for `__enumext_redefine_item:`)

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

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

```

3191 \cs_new_protected:Nn \__enumext_make_label:
3192 {
3193     \IfDocumentMetadataTF
3194     {
3195         \__enumext_make_label_box:
3196     }
3197     { \__enumext_make_label_std: }
3198 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3199 \cs_new_protected:Nn \__enumext_make_label_std:
3200 {
3201     \RenewDocumentCommand \makelabel { m }
3202     {
3203         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3204         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3205         \bool_if:cTF { l__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_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3212         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3213     }
3214 }

```

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

```

3215 \cs_new_protected:Nn \__enumext_make_label_box:
3216 {
3217     \RenewDocumentCommand \makelabel { m }
3218     {
3219         \makebox
3220         [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3221         [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3222         {
3223             \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3224             \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3225             {
3226                 \__enumext_item_star_exec:
3227                 \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3228             }
3229             { ##1 }
3230             \tl_gclear:N \g__enumext_item_symbol_aux_tl
3231         }
3232     }
3233 }

```

(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*.
item-pos*
3234 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3235 {
3236   \keys_define:nn { enumext / #1 }
3237   {
3238     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3239     item-sym* .value_required:n = true,
3240     item-sym* .initial:n = {$\star$},
3241     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3242     item-pos* .value_required:n = true,
3243   }
3244 }
3245 \clist_map_inline:nn
3246 {
3247   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3248 }
3249 { \__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*

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

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3250 \cs_set_protected:Npn \__enumext_tmp:n #1
3251 {
3252   \keys_define:nn { enumext / #1 }
3253   {
3254     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3255   }
3256 }
3257 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3258 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3259 {
3260   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3261 }
3262 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3263 {
3264   \tl_if_blank:nTF {#2}
3265   {
3266     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3267   }
3268   {
3269     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3270   }
3271 }

```

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

12.35.2 Handling unknown keys for enumext*

Define and set unknown key for enumext* environment.

```

\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3272 \keys_define:nn { enumext / enumext* }
3273 {
3274   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3275 }

```

Internal functions for handling unknown key.

```

3276 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3277 {
3278   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3279 }
3280 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3281 {
3282   \tl_if_blank:nTF {#2}
3283   {
3284     \msg_error:nnn { enumext } { starred-unknown-key } {#1}

```

```

3285     }
3286     {
3287         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3288     }
3289 }

```

(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
3290 \cs_set_protected:Npn \__enumext_tmp:n #1
3291 {
3292     \keys_define:nn { enumext / #1 }
3293     {
3294         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3295     }
3296 }
3297 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3298 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3299 {
3300     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3301 }
3302 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3303 {
3304     \tl_if_blank:nTF {#2}
3305     {
3306         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3307     }
3308     {
3309         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3310     }
3311 }

```

(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`.

```

3312 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3313 {
3314     \tl_if_novalue:nTF { #1 }
3315     {
3316         \bool_set_true:N \__enumext_wrap_label_v_bool
3317         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3318     }
3319     {
3320         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3321         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3322     }
3323 }

```

(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.

```

3324 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3325 {
3326     \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3327     \__enumext_keyans_show_left:n { #1 }
3328     \bool_set_true:N \__enumext_wrap_label_v_bool
3329     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3330     \__enumext_keyans_show_item_opt:

```

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

```

3331 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3332 \__enumext_keyans_addto_prop:n { #1 }
3333 \__enumext_keyans_store_ref:
3334 \__enumext_keyans_addto_seq:n { #1 }
3335 \int_gincr:N \g__enumext_check_starred_cmd_int
3336 }

```

(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).

```

3337 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3338 {
3339   \RenewDocumentCommand \item { s o }
3340   {
3341     \bool_if:nTF {##1}
3342     {
3343       \peek_remove_spaces:n
3344       {
3345         \__enumext_keyans_starred_item:n {##2}
3346       }
3347     }
3348     {
3349       \__enumext_keyans_default_item:n {##2}
3350     }
3351   }
3352 }

```

(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).

```

3353 \cs_new_protected:Nn \__enumext_keyans_make_label:
3354 {
3355   \IfDocumentMetadataTF
3356   {
3357     \__enumext_keyans_make_label_box:
3358   }
3359   { \__enumext_keyans_make_label_std: }
3360 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3361 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3362 {
3363   \RenewDocumentCommand \makeLabel { m }
3364   {
3365     \tl_use:N \l__enumext_label_fill_left_v_tl
3366     \tl_use:N \l__enumext_label_font_style_v_tl
3367     \bool_if:NTF \l__enumext_wrap_label_v_bool
3368     {
3369       \__enumext_wrapper_label_v:n { ##1 }
3370     }
3371     { ##1 }
3372     \tl_use:N \l__enumext_label_fill_right_v_tl
3373   }
3374 }

```

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

```

3375 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3376 {
3377   \RenewDocumentCommand \makeLabel { m }
3378   {
3379     \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3380     {

```

```

3381 \tl_use:N \l__enumext_label_font_style_v_tl
3382 \bool_if:NTF \l__enumext_wrap_label_v_bool
3383 {
3384     \__enumext_wrapper_label_v:n { ##1 }
3385 }
3386 { ##1 }
3387 }
3388 }
3389 }

```

(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.

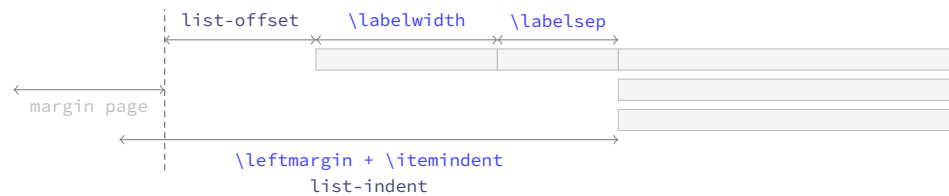


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

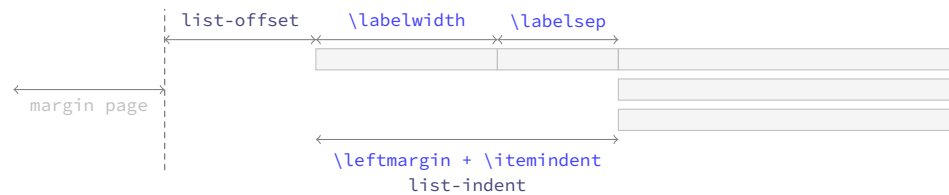


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

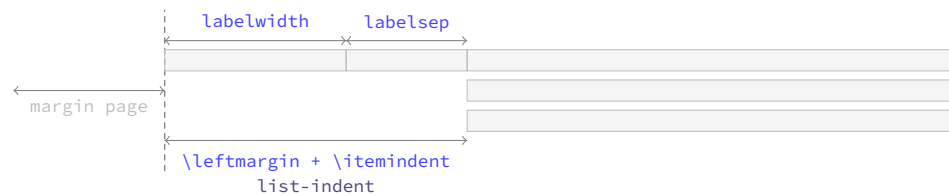


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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

```

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

```

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

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

```

3390 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3391 {
3392     \dim_compare:nNt { #1 } < { \c_zero_dim }
3393     {
3394         \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3395         \dim_set:Nn #1 { \dim_abs:n { #1 } }

```

```

3396     }
3397     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3398     {
3399         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3400         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3401     }

```

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

```

3402     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3403     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3404     {
3405         \dim_set:Nn #6 { #1 + #2 - #4 }
3406         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3407     }
3408     {
3409         \dim_compare:nNnT { #4 } = { #1 + #2 }
3410         { \dim_set:Nn #6 { \c_zero_dim } }
3411         \dim_compare:nNnT { #4 } < { #1 + #2 }
3412         { \dim_set:Nn #6 { #1 + #2 - #4 } }
3413         \dim_compare:nNnT { #4 } > { #1 + #2 }
3414         {
3415             \dim_set:Nn #6 { -#1 - #2 + #4 }
3416             \dim_set:Nn #6 { #6*-1 }
3417         }
3418         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3419     }
3420 }
3421 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

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.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3422 \cs_set_protected:Npn \__enumext_tmp:n #1
3423 {
3424     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3425     {
3426         \__enumext_calc_hspace:ccccc
3427         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3428         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3429         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3430         { \__enumext_leftmargin_tmp_#1_bool }
3431         \clist_map_inline:nn
3432         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3433         { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3434         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3435         { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3436         \usecounter { enumX#1 }
3437         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3438         \str_if_eq:nnTF {#1} { v }
3439         {
3440             \__enumext_keyans_redefine_item:
3441             \__enumext_keyans_make_label:
3442             \__enumext_keyans_ref:
3443             \__enumext_keyans_fake_item_indent:
3444             \bool_if:cT { \__enumext_show_length_#1_bool }
3445             {
3446                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3447             }
3448         }
3449         {
3450             \__enumext_redefine_item:
3451             \__enumext_make_label:
3452             \__enumext_standar_ref:
3453             \__enumext_fake_item_indent:
3454             \bool_if:cT { \__enumext_show_length_#1_bool }
3455             {

```

```

3456         \msg_term:nne { enumext } { list-lengths } {#1}
3457         { \int_use:N \l__enumext_level_int }
3458     }
3459 }
3460 }
3461 }
3462 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

3463 \cs_set_protected:Npn \__enumext_tmp:n #1
3464 {
3465     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3466     {
3467         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3468         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3469         \__enumext_calc_hspace:ccccc
3470         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3471         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3472         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3473         { l__enumext_leftmargin_tmp_#1_bool }
3474         \clist_map_inline:nn
3475         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3476         { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3477         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3478         { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3479         \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3480         \skip_zero:N \partopsep
3481         \usecounter { enumX#1 }
3482         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3483         \__enumext_starred_ref:
3484         \str_if_eq:nnTF {#1} { vii }
3485         {
3486             \__enumext_fake_item_vii:
3487             \bool_if:cT { l__enumext_show_length_vii_bool }
3488             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3489         }
3490         {
3491             \__enumext_fake_item_viii:
3492             \bool_if:cT { l__enumext_show_length_#1_bool }
3493             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3494         }
3495     }
3496 }
3497 \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”.

```

3498 \cs_new_protected:Nn \__enumext_safe_exec:
3499 {
3500     \__enumext_internal_mini_page:
3501     \__enumext_is_not_nested:
3502     \int_incr:N \l__enumext_level_int
3503     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3504     { \msg_fatal:nn { enumext } { list-too-deep } }
3505     \bool_set_true:N \l__enumext_standar_bool
3506     \bool_set_false:N \l__enumext_starred_bool
3507     \__enumext_is_on_first_level:
3508 }

```

(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 storage *(sequence)* if the key `save-key` is not active.

```

3509 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3510 {
3511     \tl_if_novalue:nF {#1}
3512     {
3513         \str_clear:N \l__enumext_series_str
3514         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3515         {
3516             \keys_set:nn { enumext / level-1 } {#1}
3517             \__enumext_parse_series:n {#1}
3518             \__enumext_nested_base_line_fix:
3519         }
3520         {
3521             \exp_args:Ne \keys_set:nn
3522                 { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3523         }
3524         \__enumext_store_active_keys:n {#1}
3525     }
3526 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level:

The __enumext_start_store_level: function activate the level saving mechanism for *storage* in *(sequence)* for the command `\anskey` and the environment `anskey*`.

```

3527 \cs_new_protected:Nn \__enumext_start_store_level:
3528 {
3529     \bool_lazy_all:nT
3530     {
3531         { \bool_if_p:N \l__enumext_store_active_bool }
3532         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3533         { \bool_if_p:N \g__enumext_standar_bool }
3534     }
3535     {
3536         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3537         {
3538             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3539             \__enumext_store_level_open:
3540         }
3541     }

```

If `enumext` are nested in `enumext*` add __enumext_store_level_open: to preserve the stored structure.

```

3542     \bool_lazy_all:nT
3543     {
3544         { \bool_if_p:N \l__enumext_store_active_bool }
3545         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3546         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3547     }
3548     {
3549         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3550         {
3551             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3552             \__enumext_store_level_open:
3553         }
3554     }
3555 }

```

(End of definition for __enumext_start_store_level:.)

__enumext_stop_store_level:

The __enumext_stop_store_level: function stop the level saving mechanism for *storage* in *(sequence)* for the command `\anskey` and the environment `anskey*`.

```

3556 \cs_new_protected:Nn \__enumext_stop_store_level:
3557 {
3558     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3559     {

```



```

3560     \__enumext_store_level_close:
3561   }
3562 }

```

(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.

```

3563 \cs_new_protected:Nn \__enumext_multicols_start:
3564 {
3565   \int_compare:nNtT
3566     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3567   {
3568     \dim_compare:nNtT
3569       { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3570     {
3571       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
3572         {
3573           ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
3574             + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3575             ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3576             - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3577           }
3578         }
3579       \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
3580       \int_compare:nNtT { \__enumext_level_int } > { 1 }
3581       {
3582         \dim_zero:N \columnseprule
3583       }
3584     }
3585   }

```

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.

```

3584   \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3585   {
3586     \skip_zero:N \multicolsep
3587     \__enumext_multi_addvspace:
3588   }
3589   \raggedcolumns
3590   \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
3591 }
3592 }

```

(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:`.

```

3593 \cs_new_protected:Nn \__enumext_multicols_stop:
3594 {
3595   \int_compare:nNtTF
3596     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3597   {
3598     \__enumext_stop_list:
3599     \__enumext_stop_store_level:
3600     \end{multicols}
3601     \__enumext_unskip_unkern:
3602     \__enumext_unskip_unkern:
3603     \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3604   }
3605   {
3606     \__enumext_stop_list:
3607     \__enumext_stop_store_level:
3608   }
3609 }

```

(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.

```

3610 \cs_new_protected:Nn \__enumext_before_list:
3611 {
3612   \__enumext_vspace_above:
3613   \__enumext_before_args_exec:
3614   \__enumext_check_ans_active:

```

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

```

3615   \dim_compare:nNt
3616   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3617   {
3618     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3619     {
3620       \linewidth
3621       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3622       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3623     }

```

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

```

3624   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3625   \int_gincr:N \g__enumext_minipage_stat_int
3626   \__enumext_minipage_add_space:
3627   \noindent
3628   \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3629 }
3630 \__enumext_multicols_start:
3631 }

```

(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.

```

3632 \cs_new_protected:Nn \__enumext_second_part:
3633 {
3634   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3635   {
3636     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3637     {
3638       \msg_warning:nn { enumext } { missing-miniright }
3639       \miniright
3640     }
3641     \int_gzero:N \g__enumext_minipage_stat_int
3642     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3643     \end__enumext_mini_page
3644   }
3645   {
3646     \__enumext_multicols_stop:
3647   }

```

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.

```

3648   \__enumext_after_stop_list:
3649   \__enumext_check_ans_key_hook:
3650   \__enumext_vspace_below:
3651   \bool_set_false:N \l__enumext_standar_bool
3652   \__enumext_resume_save_counter:
3653 }

```

(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.

```

3654 \cs_new_protected:Nn \__enumext_set_item_width:
3655 {
3656   \dim_set:Nn \itemwidth { \linewidth }
3657   \dim_compare:nT
3658   {
3659     \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3660   }
3661   {
3662     \dim_sub:Nn \itemwidth
3663     {
3664       \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3665     }
3666   }
3667 }

```

(End of definition for `__enumext_set_item_width:`.)

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

```

3668 \NewDocumentEnvironment{enumext}{0}{ }
3669 {
3670   \__enumext_safe_exec:
3671   \__enumext_parse_keys:n {#1}
3672   \__enumext_before_list:
3673   \__enumext_start_store_level:
3674   \__enumext_start_list:nn
3675   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3676   {
3677     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3678     \__enumext_before_keys_exec:
3679   }
3680   \__enumext_set_item_width:
3681   \__enumext_after_args_exec:
3682 }
3683 {
3684   \__enumext_second_part:
3685 }

```

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`.

```

3686 \__enumext_after_env:nn {enumext}
3687 {
3688   \__enumext_execute_after_env:
3689 }

```

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

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”.

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

```

3690 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3691 {
3692   \bool_if:NF \__enumext_store_active_bool
3693   {
3694     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3695   }
3696   \int_incr:N \__enumext_keyans_level_int
3697   \bool_set_true:N \__enumext_keyans_env_bool
3698   \__enumext_keyans_name_and_start:
3699   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3700   \bool_set_false:N \__enumext_store_active_bool
3701   \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
3702   {

```

```

3703         \msg_error:nn { enumext } { keyans-nested }
3704     }
3705     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3706     {
3707         \msg_error:nn { enumext } { keyans-wrong-level }
3708     }
3709 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3710 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3711 {
3712     \keys_set:nn { enumext / keyans } {#1}
3713 }

```

(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:
3714 \cs_new_protected:Nn \__enumext_before_list_v:
3715 {
3716     \__enumext_vspace_above_v:
3717     \__enumext_before_args_exec_v:
3718     \dim_compare:nNt { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3719     {
3720         \dim_set:Nn \l__enumext_minipage_left_v_dim
3721         {
3722             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3723         }
3724         \bool_set_true:N \l__enumext_minipage_active_v_bool
3725         \int_gincr:N \g__enumext_minipage_stat_int
3726         \__enumext_keyans_minipage_add_space:
3727         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3728     }
3729     \__enumext_keyans_multicols_start:
3730 }
3731 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3732 {
3733     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3734     {
3735         \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3736         {
3737             \dim_set:Nn \l__enumext_columns_sep_v_dim
3738             {
3739                 (
3740                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3741                 ) / \l__enumext_columns_v_int
3742                 - \l__enumext_listoffset_v_dim
3743             }
3744         }
3745         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3746         \dim_zero:N \columnseprule % no rule here
3747         \bool_if:NF \l__enumext_minipage_active_v_bool
3748         {
3749             \skip_zero:N \multicolsep
3750             \__enumext_keyans_multi_addvspace:
3751         }
3752         \raggedcolumns
3753         \begin{multicols}{ \l__enumext_columns_v_int }
3754     }
3755 }
3756 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3757 {
3758     \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }
3759     {
3760         \__enumext_stop_list:
3761         \end{multicols}
3762         \__enumext_unskip_unkern:
3763         \__enumext_unskip_unkern:
3764         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3765     }

```

```

3766     {
3767         \__enumext_stop_list:
3768     }
3769 }
3770 \cs_new_protected:Nn \__enumext_second_part_v:
3771 {
3772     \bool_if:NTF \l__enumext_minipage_active_v_bool
3773     {
3774         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3775         {
3776             \msg_warning:nn { enumext } { missing-miniright }
3777             \miniright
3778         }
3779         \int_gzero:N \g__enumext_minipage_stat_int
3780         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3781         \end__enumext_mini_page
3782         \par\addvspace{ \l__enumext_minipage_after_skip }
3783     }
3784     {
3785         \__enumext_keyans_multicols_stop:
3786     }
3787     \bool_set_false:N \l__enumext_keyans_env_bool
3788     \__enumext_after_stop_list_v:
3789     \__enumext_vspace_below_v:
3790 }

```

(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.

```

3791 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3792 {
3793     \dim_set:Nn \itemwidth { \linewidth }
3794     \dim_compare:nT
3795     {
3796         \l__enumext_listoffset_v_dim != \c_zero_dim
3797     }
3798     {
3799         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3800     }
3801 }

```

(End of definition for __enumext_keyans_set_item_width:.)

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

```

3802 \NewDocumentEnvironment{keyans}{0}{}
3803 {
3804     \__enumext_keyans_safe_exec:
3805     \__enumext_keyans_parse_keys:n {#1}
3806     \__enumext_before_list_v:
3807     \__enumext_start_list:nn
3808     { \tl_use:N \l__enumext_label_v_tl }
3809     {
3810         \__enumext_list_arg_two_v:
3811         \__enumext_before_keys_exec_v:
3812     }
3813     \__enumext_keyans_set_item_width:
3814     \__enumext_after_args_exec_v:
3815 }
3816 {
3817     \__enumext_check_starred_cmd:n { item }
3818     \__enumext_second_part_v:
3819 }

```

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

12.40 Tagging PDF support for non-standart list environments

The L^AT_EX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the **list** environment. Unfortunately non-standard *list environments* like **keyanspic** or the horizontal list environments **enumext*** and **keyans*** are not structured in a nice way, i.e. the expected

result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a `list` environment is a `list` environment, no matter what it looks like in the PDF file.

To maintain a correct `list` structure when `\DocumentMetadata` is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my `\item` redefinition to be compatible with tagging-pdf](#).

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
3820 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3821 \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3822 {
3823   \tag_resume:n {#1}
3824   \tag_struct_begin:n {tag=LI}
3825   \tag_struct_begin:n {tag=Lbl}
3826   \tag_mc_begin:n {tag=Lbl}
3827 }
3828 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3829 {
3830   \tag_mc_end:
3831   \tag_struct_end:n {tag=Lbl}
3832   \tag_struct_begin:n {tag=LBody}
3833   \tag_struct_begin:n {tag=text-unit}
3834   \tag_struct_begin:n {tag=text}
3835 }
3836 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3837 {
3838   \tag_struct_end:n {tag=text}
3839   \tag_struct_end:n {tag=text-unit}
3840   \tag_struct_end:n {tag=LBody}
3841   \tag_struct_end:n {tag=LI}
3842   \tag_suspend:n {#1}
3843 }
```

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

```
3844 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
3845 {
3846   \IfDocumentMetadataTF
3847   {
3848     \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3849     \socket_use:n {tagsupport/enumext/starred} {#1}
3850   } {}
3851 }
3852 \cs_new_protected_nopar:Npn __enumext_stop_start_list_tag:
3853 {
3854   \IfDocumentMetadataTF
3855   {
3856     \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3857     \socket_use:nn {tagsupport/enumext/starred} { }
3858   } {}
3859 }
3860 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
3861 {
3862   \IfDocumentMetadataTF
3863   {
3864     \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3865     \socket_use:nn {tagsupport/enumext/starred} {#1}
3866   } {}
3867 }
```

(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:
3868 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3869 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3870 {
3871   \tag_resume:n {keyanspic}
3872   \tag_struct_begin:n {tag=LI}
3873   \tag_struct_begin:n {tag=Lbl}
3874   \tag_mc_begin:n {tag=Lbl}
3875 }
```

```
3876 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3877 {
3878   \tag_mc_end:
3879   \tag_struct_end:n {tag=Lbl}
3880   \tag_struct_begin:n {tag=LBody}
3881   \tag_struct_begin:n {tag=text-unit}
3882   \tag_struct_begin:n {tag=text}
3883   \tag_mc_begin:n {tag=text}
3884 }
3885 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3886 {
3887   \tag_mc_end:
3888   \tag_struct_end:n {tag=text-unit}
3889   \tag_struct_end:n {tag=text}
3890   \tag_struct_end:n {tag=LBody}
3891   \tag_struct_end:n {tag=LI}
3892   \tag_suspend:n {keyanspic}
3893 }
```

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

```
3894 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3895 {
3896   \IfDocumentMetadataTF
3897   {
3898     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3899     \socket_use:n {tagsupport/enumext/keyanspic}
3900   } {}
3901 }
3902 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3903 {
3904   \IfDocumentMetadataTF
3905   {
3906     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3907     \socket_use:nn {tagsupport/enumext/keyanspic}
3908   } {}
3909 }
3910 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3911 {
3912   \IfDocumentMetadataTF
3913   {
3914     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3915     \socket_use:nn {tagsupport/enumext/keyanspic}
3916   } {}
3917 }
```

(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 $\langle label \rangle$ as the `keyans` environment, but it does not use `\item`. The $\langle contents \rangle$ are passed to the environment by means of the `\anspic` command as replacement for `\item` command and placed inside `minipage` environments, with the $\langle label \rangle$ centered “above” or “below”, adjusting *widths* and *position* according to the options passed to the environment.

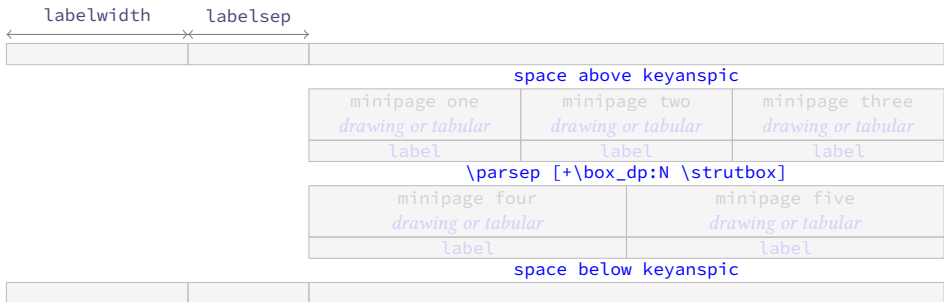


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

The environment `keyanspic` will take two arguments, the first *starred argument* “`*`” will set the position of the $\langle label \rangle$ 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 [`$\langle n^\circ upper, n^\circ lower \rangle$`] and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “upper” and “lower” within the environment, if not present these will be printed on a *single line*.

- ◆ One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of $\langle label \rangle$, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to $\langle label \rangle$ and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is $\langle label \rangle$, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

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* ‘*’.

```

3918 \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3919 {
3920   \int_incr:N \l__enumext_keyans_pic_level_int
3921   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3922   {
3923     \msg_error:nn { enumext } { keyanspic-nested }
3924   }
3925   \__enumext_keyans_name_and_start:
3926   \bool_if:nTF { #1 }
3927   {
3928     \bool_set_true:N \l__enumext_keyans_pic_star_bool
3929     \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
3930   }
3931   {
3932     \str_set:Nn \l__enumext_anspic_mini_pos_str { b }
3933   }
3934 }

```

(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`.

```

3935 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3936 {
3937   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3938   {
3939     \skip_set:Nn #1 { -#1 }
3940   }
3941 }

```

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

`__enumext_keyans_pic_arg_two:`

The `__enumext_keyans_pic_arg_two:` function will be used in the *second argument* of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the “spaces” and the $\langle keys \rangle$ label and `wrap-label` from the `keyans` environment.

The first thing we need to do is set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to “false”, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```

3942 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3943 {
3944   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3945   \__enumext_list_arg_two_v:
3946   \__enumext_keyans_pic_skip_abs:N \parsep

```

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

```

3947   \bool_if:NF \l__enumext_keyans_pic_star_bool
3948   {
3949     \stepcounter { enumXv }
3950     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3951     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3952     {
3953       \box_ht_plus_dp:N \l__enumext_anspic_label_box
3954     }
3955     \skip_add:Nn \parsep

```

```

3956         {
3957             \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3958         }
3959         \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3960     }

```

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.

```

3961     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3962     \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3963     \dim_zero:N \labelwidth
3964     \dim_zero:N \listparindent
3965     \dim_zero:N \labelsep
3966     \skip_zero:N \partopsep
3967     \skip_zero:N \itemsep
3968 }

```

(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`.

```

3969 \NewDocumentEnvironment{keyanspic}{s o }
3970 {
3971     \__enumext_keyans_pic_safe_exec:n { #1 }
3972     \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3973     \IfDocumentMetadataTF
3974     {
3975         \tag_suspend:n {list}
3976     }{}
3977     \item[] \scan_stop:
3978     % paranoia
3979     \RenewDocumentCommand \item {}
3980     {
3981         \msg_error:nn { enumext } { keyanspic-item-cmd }
3982     }
3983     \IfDocumentMetadataTF
3984     {
3985         \tag_resume:n {keyanspic}
3986         \tag_tool:n {para/tagging=false}
3987         \tag_suspend:n {keyanspic}
3988     } { }
3989 }
3990 {
3991     \IfDocumentMetadataTF
3992     {
3993         \tag_resume:n {keyanspic}
3994         \tag_struct_begin:n {tag=L,attribute=enumerate}
3995     } { }

```

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.

```

3996     \tl_if_novalue:nTF { #2 }
3997     {
3998         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
3999     }
4000     { \__enumext_anspic_print:n { #2 } }
4001     \IfDocumentMetadataTF
4002     {
4003         \tag_suspend:n {keyanspic}
4004     } { }
4005     \end{list}
4006     \IfDocumentMetadataTF
4007     {
4008         \tag_struct_end:
4009         \tag_struct_end:
4010     } { }

```

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

```

4011     \__enumext_check_starred_cmd:n { anspic }
4012     \setcounter { enumXvi } { 0 }
4013     \bool_if:NTF \l__enumext_keyans_pic_star_bool

```

```

4014     {
4015         \par\addvspace{ 0.5\box_dp:N \strutbox }
4016     }
4017     {
4018         \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
4019     }
4020     %\bool_set_false:N \l__enumext_store_active_bool
4021 }

```

(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*` [`⟨content⟩`] store the current `⟨label⟩` next to the [`⟨content⟩`] (if it is present) in the `⟨sequence⟩` and `⟨prop list⟩` defined by `save-ans` key. The third (mandatory) argument “drawing or tabular” is NOT stored in the `⟨sequence⟩` or `⟨prop list⟩`.

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

```

4022 \NewDocumentCommand \anspic { s o +m }
4023 {
4024     \bool_if:NF \l__enumext_store_active_bool
4025     {
4026         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4027     }
4028     \int_compare:nNt { \l__enumext_level_int } > { 1 }
4029     {
4030         \msg_error:nn { enumext } { keyanspic-wrong-level }
4031     }
4032     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4033     {
4034         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4035     }
4036     \seq_put_right:Nn \l__enumext_anspic_args_seq
4037     {
4038         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4039     }
4040 }

```

(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* ‘*’.

```

4041 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4042 {
4043     \bool_if:NF \l__enumext_keyans_pic_star_bool
4044     {
4045         \IfDocumentMetadataTF
4046         {
4047             \tag_suspend:n {keyanspic}
4048         } { }
4049         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4050         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4051         {
4052             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4053         }
4054         \IfDocumentMetadataTF
4055         {
4056             \tag_resume:n {keyanspic}
4057         } { }
4058     }
4059 }

```

(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.

```

4060 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4061 {
4062   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4063   {
4064     \bool_if:nT { #1 }
4065     {
4066       \__enumext_keyans_addto_prop:n { #2 }
4067       \__enumext_keyans_store_ref:
4068       \__enumext_keyans_addto_seq:n { #2 }
4069       \int_gincr:N \g__enumext_check_starred_cmd_int
4070       \bool_lazy_or:nnT
4071       { \bool_if_p:N \l__enumext_show_answer_bool }
4072       { \bool_if_p:N \l__enumext_show_position_bool }
4073       {
4074         \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4075         \__enumext_keyans_show_left:n { #2 }
4076         \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4077       }
4078     }
4079     \tl_use:N \l__enumext_label_font_style_v_tl
4080     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4081     \__enumext_keyans_show_item_opt:
4082   }
4083 }

```

(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.

```

4084 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4085 {
4086   \stepcounter { enumXvi }
4087   \__enumext_anspic_body_dim:n { #3 }
4088   \bool_if:NTF \l__enumext_keyans_pic_star_bool
4089   {
4090     \__enumext_anspic_label:nn { #1 } { #2 }
4091   }
4092   {
4093     \raisebox
4094     {
4095       -\dim_eval:n
4096       {
4097         \l__enumext_anspic_label_htdp_dim
4098         + \l__enumext_anspic_body_htdp_dim
4099         + \box_dp:N \strutbox
4100       }
4101     }
4102     [ opt ] [ opt ]
4103     {
4104       \__enumext_anspic_label:nn { #1 } { #2 }
4105     }
4106   }
4107 }
4108 %

```

(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.

```

4109 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4110 {
4111   \__enumext_anspic_start_list_tag:
4112   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4113   \__enumext_anspic_stop_start_list_tag:
4114   \\ #3
4115   \__enumext_anspic_stop_list_tag:
4116 }

```

(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`.

```

4117 \cs_new_protected:Nn \__enumext_anspic_print:n
4118 {
4119   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4120 }
4121 \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.

```

4122 \cs_new_protected:Nn \__enumext_anspic_row:n
4123 {
4124   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4125   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4126   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4127   \int_step_inline:nnn
4128     { \l__enumext_anspic_above_int + 1 }
4129     { \l__enumext_anspic_below_int }
4130     {
4131       \IfDocumentMetadataTF
4132       {
4133         \tag_suspend:n {minipage}
4134       } { }
4135       \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4136         \centering
4137         \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4138       \end{minipage}
4139       \IfDocumentMetadataTF
4140       {
4141         \tag_resume:n {minipage}
4142       } { }
4143     }
4144   \par
4145 }

```

(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 \TeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makeLabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

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

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

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

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

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

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

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:

```

```

4146 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4147 {
4148   \dim_compare:nNNT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4149   {
4150     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4151     {

```

```

4152         ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4153         / \l__enumext_columns_vii_int
4154     }
4155 }
4156 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4157 \dim_set:Nn \l__enumext_item_width_vii_dim
4158 {
4159     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4160     / \l__enumext_columns_vii_int
4161     - \l__enumext_labelwidth_vii_dim
4162     - \l__enumext_labelsep_vii_dim
4163 }

```

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

```

4164 \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4165 {
4166     \dim_sub:Nn \l__enumext_item_width_vii_dim
4167     {
4168         ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4169         / \l__enumext_columns_vii_int
4170     }
4171     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4172     {
4173         \l__enumext_rightmargin_vii_dim
4174     }
4175 }
4176 }

```

Same implementation for the `keyans*` environment.

```

4177 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4178 {
4179     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4180     {
4181         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4182         {
4183             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4184             / \l__enumext_columns_viii_int
4185         }
4186     }
4187     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4188     \dim_set:Nn \l__enumext_item_width_viii_dim
4189     {
4190         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4191         / \l__enumext_columns_viii_int
4192         - \l__enumext_labelwidth_viii_dim
4193         - \l__enumext_labelsep_viii_dim
4194     }
4195     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4196     {
4197         \dim_sub:Nn \l__enumext_item_width_viii_dim
4198         {
4199             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
4200             / \l__enumext_columns_viii_int
4201         }
4202         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4203         {
4204             \l__enumext_rightmargin_viii_dim
4205         }
4206     }
4207 }

```

(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.

```

4208 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4209 {
4210     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4211     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4212     {

```

```

4213     \msg_warning:nnee { enumext } { item-joined }
4214     { \int_use:N \l__enumext_joined_item_vii_int }
4215     { \int_use:N \l__enumext_columns_vii_int }
4216     \int_set:Nn \l__enumext_joined_item_vii_int
4217     {
4218         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4219     }
4220 }
4221 \int_compare:nNnT
4222 { \l__enumext_joined_item_vii_int }
4223 >
4224 { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4225 {
4226     \msg_warning:nnee { enumext } { item-joined-columns }
4227     { \int_use:N \l__enumext_joined_item_vii_int }
4228     {
4229         \int_eval:n
4230         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4231     }
4232     \int_set:Nn \l__enumext_joined_item_vii_int
4233     {
4234         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4235     }
4236 }
4237 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4238 {
4239     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4240     \int_decr:N \l__enumext_joined_item_aux_vii_int
4241     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4242     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4243     \dim_set:Nn \l__enumext_joined_width_vii_dim
4244     {
4245         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4246         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4247             + \l__enumext_columns_sep_vii_dim
4248             ) * \l__enumext_joined_item_aux_vii_int
4249     }
4250     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4251 }
4252 {
4253     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4254     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4255 }
4256 }

```

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

```

4257 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4258 {
4259     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4260     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4261     {
4262         \msg_warning:nnee { enumext } { item-joined }
4263         { \int_use:N \l__enumext_joined_item_viii_int }
4264         { \int_use:N \l__enumext_columns_viii_int }
4265         \int_set:Nn \l__enumext_joined_item_viii_int
4266         {
4267             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4268         }
4269     }
4270     \int_compare:nNnT
4271     { \l__enumext_joined_item_viii_int }
4272     >
4273     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4274     {
4275         \msg_warning:nnee { enumext } { item-joined-columns }
4276         { \int_use:N \l__enumext_joined_item_viii_int }
4277         {
4278             \int_eval:n
4279             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4280         }
4281         \int_set:Nn \l__enumext_joined_item_viii_int
4282         {

```



```

4283         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4284     }
4285 }
4286 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4287 {
4288     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4289     \int_decr:N \l__enumext_joined_item_aux_viii_int
4290     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4291     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4292     \dim_set:Nn \l__enumext_joined_width_viii_dim
4293     {
4294         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4295         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4296             + \l__enumext_columns_sep_viii_dim
4297             ) * \l__enumext_joined_item_aux_viii_int
4298     }
4299     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4300 }
4301 {
4302     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4303     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4304 }
4305 }

```

(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

```

\__enumext_start_mini_vii:
\__enumext_stop_mini_vii:

```

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

```

4306 \cs_new_protected:Nn \__enumext_start_mini_vii:
4307 {
4308     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4309     {
4310         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4311         {
4312             \linewidth
4313             - \l__enumext_minipage_right_vii_dim
4314             - \l__enumext_minipage_hsep_vii_dim
4315         }
4316         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4317         \dim_gset_eq:NN
4318             \g__enumext_minipage_right_vii_dim
4319             \l__enumext_minipage_right_vii_dim
4320         \__enumext_mini_addvspace_vii:
4321         \nointerlineskip\noindent
4322         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4323     }
4324 }

```

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

```

4325 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4326 {
4327     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4328     {
4329         \__enumext_stop_list:
4330         \__enumext_stop_store_level_vii:
4331         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { } { }
4332         \end__enumext_mini_page
4333         \hfill
4334         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4335     }
4336     {
4337         \__enumext_stop_list:

```

```

4338     \__enumext_stop_store_level_vii:
4339   }
4340 }

```

(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.

```

4341 \__enumext_after_env:n {enumext*}
4342 {
4343   \bool_if:NT \g__enumext_minipage_active_vii_bool
4344   {
4345     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4346     \legacy_if_gset_false:n { @minipage }
4347     \skip_vertical:N \c_zero_skip
4348     \par\addvspace { \g__enumext_minipage_right_skip }
4349     \bool_if:NF \g__enumext_minipage_center_vii_bool
4350     {
4351       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4352       {
4353         \centering
4354       }
4355     }
4356     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4357     {
4358       \tl_use:N \g__enumext_miniright_code_vii_tl
4359     }
4360     \box_use_drop:N \l__enumext_miniright_code_vii_box
4361     \skip_vertical:N \c_zero_skip
4362     \__enumext_endminipage:
4363     \par\addvspace{ \g__enumext_minipage_after_skip }
4364   }
4365   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4366   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4367   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4368   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4369   \bool_gset_false:N \g__enumext_starred_bool
4370 }

```

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

```

4371 \cs_new_protected:Nn \__enumext_start_mini_viii:
4372 {
4373   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4374   {
4375     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4376     {
4377       \linewidth
4378       - \l__enumext_minipage_right_viii_dim
4379       - \l__enumext_minipage_hsep_viii_dim
4380     }
4381     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4382     \dim_gset_eq:NN
4383       \g__enumext_minipage_right_viii_dim
4384       \l__enumext_minipage_right_viii_dim
4385     \__enumext_mini_addvspace_viii:
4386     \nointerlineskip\noindent
4387     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4388   }
4389 }
4390 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4391 {
4392   \bool_if:NTF \l__enumext_minipage_active_viii_bool
4393   {
4394     \__enumext_stop_list:
4395     \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4396     \end__enumext_mini_page
4397     \hfill
4398     \bool_gset_true:N \g__enumext_minipage_active_viii_bool

```

```

4399     }
4400     {
4401         \__enumext_stop_list:
4402     }
4403 }
4404 \__enumext_after_env:nn {keyans*}
4405 {
4406     \bool_if:NT \g__enumext_minipage_active_viii_bool
4407     {
4408         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4409         \par\addvspace { \g__enumext_minipage_right_skip }
4410         \bool_if:NF \g__enumext_minipage_center_viii_bool
4411         {
4412             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4413             {
4414                 \centering
4415             }
4416         }
4417         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4418         {
4419             \tl_use:N \g__enumext_miniright_code_viii_tl
4420         }
4421         \box_use_drop:N \l__enumext_miniright_code_viii_box
4422         \end__enumext_mini_page
4423         \par\addvspace{ \g__enumext_minipage_after_skip }
4424     }
4425     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4426     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4427     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4428     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4429 }

```

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

12.42.4 Redefining \footnote command

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

```

4430 \cs_new_protected:Nn \__enumext_footnotetext:nn
4431 {
4432     \footnotetext[#1]{#2}
4433 }
4434 \cs_new_protected:Nn \__enumext_renew_footnote:
4435 {
4436     \seq_gclear:N \g__enumext_footnote_arg_seq
4437     \seq_gclear:N \g__enumext_footnote_int_seq
4438     \RenewDocumentCommand \footnote { o +m }
4439     {
4440         \tl_if_novalue:nTF {##1}
4441         {
4442             \stepcounter{footnote}
4443             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4444         }
4445         {
4446             \int_gset:Nn \g__enumext_footnote_int { ##1 }
4447         }
4448         \footnotemark [ \g__enumext_footnote_int ]
4449         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4450         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4451     }
4452 }
4453 \cs_new_protected:Nn \__enumext_print_footnote:
4454 {
4455     \seq_if_empty:NF \g__enumext_footnote_int_seq
4456     {
4457         \seq_map_pairwise_function:NNN
4458         \g__enumext_footnote_int_seq
4459         \g__enumext_footnote_arg_seq
4460         \__enumext_footnotetext:nn
4461     }
4462 }

```

(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 pdf \TeX and Lua \TeX .

```

4463 \NewDocumentEnvironment{enumext*}{o }
4464 {
4465   \__enumext_safe_exec_vii:
4466   \__enumext_parse_keys_vii:n {#1}
4467   \__enumext_before_list_vii:
4468   \__enumext_start_store_level_vii:
4469   \__enumext_start_list:nn { }
4470   {
4471     \__enumext_list_arg_two_vii:
4472     \__enumext_before_keys_exec_vii:
4473   }
4474   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4475   \__enumext_starred_columns_set_vii:
4476   \item[] \scan_stop:
4477   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4478   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4479   \ignorespaces
4480 }
4481 {
4482   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4483   \__enumext_stop_item_tmp_vii:
4484   \__enumext_remove_extra_parsep_vii:
4485   \__enumext_after_list_vii:
4486 }

```

(End of definition for `enumext*`. This function is documented on page 6.)

`__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.

```

4487 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4488 {
4489   \__enumext_internal_mini_page:
4490   \__enumext_is_not_nested:
4491   \int_incr:N \l__enumext_level_h_int
4492   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4493   {
4494     \msg_error:nn { enumext } { nested }
4495   }
4496   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4497   {
4498     \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4499   }
4500   \bool_set_true:N \l__enumext_starred_bool
4501   \bool_set_false:N \l__enumext_standar_bool
4502   \__enumext_is_on_first_level:
4503 }

```

(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.

```

4504 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4505 {
4506   \tl_if_novalue:nF {#1}

```

```

4507     {
4508         \str_clear:N \__enumext_series_str
4509         \keys_set:nn { enumext / enumext* } {#1}
4510         \__enumext_parse_series:n {#1}
4511         \__enumext_store_active_keys_vii:n {#1}
4512     }
4513 }

```

(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*`.

```

4514 \cs_new_protected:Nn \__enumext_before_list_vii:
4515 {
4516     \__enumext_vspace_above_vii:
4517     \__enumext_check_ans_active:
4518     \__enumext_before_args_exec_vii:
4519     \__enumext_start_mini_vii:
4520 }

```

(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.

```

4521 \cs_new_protected:Nn \__enumext_after_list_vii:
4522 {
4523     \__enumext_stop_mini_vii:
4524     \__enumext_after_stop_list_vii:
4525     \__enumext_check_ans_key_hook:
4526     \__enumext_vspace_below_vii:
4527     \bool_set_false:N \__enumext_starred_bool
4528     \__enumext_resume_save_counter:
4529 }

```

(End of definition for __enumext_after_list_vii:.)

`__enumext_start_store_level_vii:` and `__enumext_stop_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`.

```

4530 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4531 {
4532     \bool_if:NT \__enumext_store_active_bool
4533     {
4534         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4535         {
4536             \__enumext_store_level_open_vii:
4537         }
4538     }
4539 }
4540 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4541 {
4542     \bool_if:NT \__enumext_store_active_bool
4543     {
4544         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4545         {
4546             \__enumext_store_level_close_vii:
4547         }
4548     }
4549 }

```

(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.

```
4550 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4551 {
4552   \skip_horizontal:n { -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim }
4553 }
```

(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`.

```
4554 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4555 {
4556   \__enumext_stop_item_tmp_vii:
4557   \int_incr:N \__enumext_item_column_pos_vii_int
4558   \int_gincr:N \g__enumext_item_count_all_vii_int
4559   \__enumext_item_peek_args_vii:
4560 }
```

(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).

```
4561 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4562 {
4563   \peek_meaning:NTF (
4564     { \__enumext_joined_item_vii:w }
4565     { \__enumext_joined_item_vii:w (1) }
4566 }
```

(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`.

```
4567 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4568 {
4569   \__enumext_starred_joined_item_vii:n {#1}
4570   \peek_meaning_remove:NTF *
4571     { \__enumext_starred_item_vii:w }
4572     { \__enumext_standar_item_vii:w }
4573 }
```

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

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

```
4574 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4575 {
4576   \bool_set_false:N \__enumext_item_starred_vii_bool
4577   \peek_meaning:NTF [
4578     {
4579       \bool_set_eq:NN \__enumext_wrap_label_vii_bool \__enumext_wrap_label_opt_vii_bool
4580       \__enumext_start_item_vii:w
4581     }
4582     {
4583       \bool_set_true:N \__enumext_wrap_label_vii_bool
```

```

4584         \legacy_if_set_true:n { @noitemarg }
4585         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4586     }
4587 }

```

(End of definition for __enumext_standar_item_vii:w)

```

\__enumext_starred_item_vii:w
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w

```

The function __enumext_starred_item_vii:w together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]`.

```

4588 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4589 {
4590     \bool_set_true:N \__enumext_item_starred_vii_bool
4591     \bool_set_true:N \__enumext_wrap_label_vii_bool
4592     \peek_meaning:NTF [
4593         { \__enumext_starred_item_vii_aux_i:w }
4594         { \__enumext_starred_item_vii_aux_ii:w }
4595     }
4596     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4597     {
4598         \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4599         \__enumext_starred_item_vii_aux_ii:w
4600     }
4601     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4602     {
4603         \peek_meaning:NTF [
4604             { \__enumext_starred_item_vii_aux_iii:w }
4605             {
4606                 \dim_set_eq:NN \__enumext_item_symbol_sep_vii_dim \__enumext_labelsep_vii_dim
4607                 \legacy_if_set_true:n { @noitemarg }
4608                 \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4609             }
4610         }
4611     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4612     {
4613         \dim_set:Nn \__enumext_item_symbol_sep_vii_dim {#1}
4614         \legacy_if_set_true:n { @noitemarg }
4615         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
4616     }

```

(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` 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.

```

4617 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4618 {
4619     \legacy_if:nT { @noitemarg }
4620     {
4621         \legacy_if_set_false:n { @noitemarg }
4622         \legacy_if:nT { @nmbrrlist }
4623         {
4624             \IfDocumentMetadataTF
4625             {
4626                 \bool_if:NT \__enumext_hyperref_bool
4627                 {
4628                     \legacy_if_set_true:n { @hyper@item }
4629                 }
4630             } { }
4631             \refstepcounter{enumXvii}
4632             \bool_if:NT \__enumext_check_answers_bool
4633             {
4634                 \int_gincr:N \g__enumext_item_number_int
4635                 \bool_set_true:N \__enumext_item_number_bool

```



```

4636         }
4637     }
4638 }
4639 \bool_if:NT \l__enumext_item_starred_vii_bool
4640 {
4641     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4642     {
4643         \tl_gset_eq:NN
4644         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4645     }
4646     \mode_leave_vertical:
4647     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4648     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4649     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4650     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4651 }
4652 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4653 {
4654     \tl_use:N \l__enumext_label_font_style_vii_tl
4655     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4656     {
4657         \__enumext_wrapper_label_vii:n {#1}
4658     }
4659     { #1 }
4660 }
4661 \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4662 }

```

(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.

```

4663 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4664 {
4665     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4666     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4667     {
4668         \l__enumext_joined_width_vii_dim
4669         + \l__enumext_labelwidth_vii_dim
4670         + \l__enumext_labelsep_vii_dim
4671     }

```

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

```

4672 \IfDocumentMetadataTF { }
4673 {
4674     \bool_if:NF \l__enumext_footnotes_key_bool
4675     {
4676         \__enumext_renew_footnote:
4677     }
4678 }

```

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

```

4679 \__enumext_start_list_tag:n {enumext*}
4680 \__enumext_fake_make_label_vii:n {#1}
4681 \__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.

```

4682 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4683 \tl_use:N \l__enumext_after_list_args_vii_tl
4684 \tl_use:N \l__enumext_fake_item_indent_vii_tl
4685 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4686 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4687 }

```

(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*.

```
4688 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4689 {
4690     \__enumext_endminipage:
4691     \__enumext_stop_list_tag:n {enumext*}
4692     \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`.

```
4693     \int_set:Nn \hbadness { 10000 }
4694     \box_use_drop:N \l__enumext_item_text_vii_box
4695     \IfDocumentMetadataTF { }
4696     {
4697         \bool_if:NF \l__enumext_footnotes_key_bool
4698         {
4699             \__enumext_print_footnote:
4700         }
4701     }
```

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

```
4702     \int_compare:nNnTF
4703     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4704     {
4705         \par\noindent
4706         \int_zero:N \l__enumext_item_column_pos_vii_int
4707     }
4708     {
4709         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4710     }
4711 }
```

(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⟩`.

```
4712 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4713 {
4714     \int_compare:nNnTF
4715     {
4716         \int_mod:nn
4717         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4718     }
4719     =
4720     { 0 }
4721     {
4722         \para_end:
4723         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4724         \skip_vertical:N \c_zero_skip
4725         \int_gzero:N \g__enumext_item_count_all_vii_int
4726     }
4727 }
```

(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`.

```
4728 \__enumext_after_env:nn {enumext*}
4729 {
4730     \__enumext_execute_after_env:
4731 }
```

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.

```

4732 \NewDocumentEnvironment{keyans*}{o}{
4733   {
4734     \__enumext_safe_exec_viii:
4735     \__enumext_parse_keys_viii:n {#1}
4736     \__enumext_before_list_viii:
4737     \__enumext_start_list:nn { }
4738     {
4739       \__enumext_list_arg_two_viii:
4740       \__enumext_before_keys_exec_viii:
4741     }
4742     % Stop tagging
4743     \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4744     \__enumext_starred_columns_set_viii:
4745     \item[] \scan_stop:
4746     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4747     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4748     \ignorespaces
4749   }
4750   {
4751     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4752     \__enumext_stop_item_tmp_viii:
4753     \__enumext_remove_extra_parsep_viii:
4754     \__enumext_check_starred_cmd:n { item }
4755     \__enumext_after_list_viii:
4756   }

```

(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.

```

4757 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4758   {
4759     \bool_if:NF \l__enumext_store_active_bool
4760     {
4761       \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4762     }
4763     \int_incr:N \l__enumext_keyans_level_h_int
4764     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4765     {
4766       \msg_error:nn { enumext } { nested }
4767     }
4768     \__enumext_keyans_name_and_start:
4769     \bool_if:NT \l__enumext_starred_bool
4770     {
4771       \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4772     }
4773     \bool_set_true:N \l__enumext_starred_bool
4774     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4775     \bool_set_false:N \l__enumext_store_active_bool
4776     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4777     {
4778       \msg_error:nn { enumext } { keyans-wrong-level }
4779     }
4780   }

```

(End of definition for __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

Parse [`<key = val>`] for `keyans*`.

```

4781 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4782 {
4783   \tl_if_novalue:nF {#1}
4784   {
4785     \keys_set:nn { enumext / keyans* } {#1}
4786   }
4787 }

```

(End of definition for __enumext_parse_keys_viii:n)

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

```

4788 \cs_new_protected:Nn \__enumext_before_list_viii:
4789 {
4790   \__enumext_vspace_above_viii:
4791   \__enumext_before_args_exec_viii:
4792   \__enumext_start_mini_viii:
4793 }

```

(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.

```

4794 \cs_new_protected:Nn \__enumext_after_list_viii:
4795 {
4796   \__enumext_stop_mini_viii:
4797   \__enumext_after_stop_list_viii:
4798   \__enumext_vspace_below_viii:
4799 }

```

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

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

```

4800 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4801 {
4802   \skip_horizontal:n { -\labelwidth_viii_dim - \labelsep_viii_dim }
4803 }

```

(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`.

```

4804 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4805 {
4806   \__enumext_stop_item_tmp_viii:
4807   \int_incr:N \l__enumext_item_column_pos_viii_int
4808   \int_gincr:N \g__enumext_item_count_all_viii_int
4809   \__enumext_item_peek_args_viii:
4810 }

```

(End of definition for __enumext_start_item_tmp_viii:.)

`__enumext_item_peek_args_viii:` The function `__enumext_item_peek_args_viii:` will handle the `\item(<number>)`. Look for the argument “(”, if it is present we will call the function `__enumext_joined_item_viii:w (<number>)`, which is in charge of joining the item’s in the same row, in case they are not present we will set the default value (1).

```

4811 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4812 {
4813   \peek_meaning:NTF (
4814     { \__enumext_joined_item_viii:w }
4815     { \__enumext_joined_item_viii:w (1) }
4816   }

```

(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`.

```

4817 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4818 {
4819   \__enumext_starred_joined_item_viii:n {#1}
4820   \peek_meaning_remove:NTF *
4821   { \__enumext_starred_item_viii:w }
4822   { \__enumext_standar_item_viii:w }
4823 }

```

(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]`.

```

4824 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4825 {
4826   \bool_set_false:N \l__enumext_item_starred_viii_bool
4827   \peek_meaning:NTF [
4828   {
4829     \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4830     \__enumext_start_item_viii:w
4831   }
4832   {
4833     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4834     \legacy_if_set_true:n { @noitemarg }
4835     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4836   }
4837 }

```

(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
4838 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4839 {
4840   \bool_set_true:N \l__enumext_item_starred_viii_bool
4841   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4842   \peek_meaning:NTF [
4843   { \__enumext_starred_item_viii_aux_i:w }
4844   { \__enumext_starred_item_viii_aux_ii:w }
4845 }

```

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`.

```

4846 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4847 {
4848   \tl_clear:N \l__enumext_store_current_label_tl
4849   \tl_if_novalue:nF { #1 }
4850   {

```

```

4851         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4852         {
4853             \tl_put_right:Ne \l__enumext_store_current_label_tl
4854             {
4855                 \l__enumext_store_keyans_item_opt_sep_tl
4856             }
4857             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4858         }
4859         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4860     }
4861     \__enumext_starred_item_viii_aux_ii:w
4862 }
4863 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4864 {
4865     \legacy_if_set_true:n { @noitemarg }
4866     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4867 }

```

(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.

```

4868 \cs_new_protected:Nn \__enumext_starred_item_exec:
4869 {
4870     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4871     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4872     \__enumext_keyans_store_ref:
4873     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4874     \__enumext_keyans_addto_seq_link:
4875     \int_gincr:N \g__enumext_check_starred_cmd_int
4876     \bool_if:NT \l__enumext_show_answer_bool
4877     {
4878         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4879     }
4880     \bool_if:NT \l__enumext_show_position_bool
4881     {
4882         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4883         {
4884             \group_begin:
4885             \exp_not:N \normalfont
4886             \exp_not:N \footnotesize [ \int_eval:n
4887                 {
4888                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4889                 }
4890             ]
4891             \group_end:
4892         }
4893         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4894     }
4895 }

```

(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.

```

4896 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4897 {
4898     \legacy_if:nT { @noitemarg }
4899     {
4900         \legacy_if_set_false:n { @noitemarg }
4901         \legacy_if:nT { @nmbrlist }
4902         {
4903             \refstepcounter{enumXviii}
4904         }
4905     }
4906     \bool_if:NT \l__enumext_item_starred_viii_bool
4907     {
4908         \__enumext_starred_item_exec:
4909     }
4910     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]

```

```

4911     {
4912         \tl_use:N \l__enumext_label_font_style_viii_tl
4913         \bool_if:NTF \l__enumext_wrap_label_viii_bool
4914         {
4915             \__enumext_wrapper_label_viii:n {#1}
4916         }
4917         { #1 }
4918     }
4919     \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
4920 }

```

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

12.44.2 Real definition of `\item` in `keyans*`

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

```

4921 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4922 {
4923     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4924     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4925     {
4926         \l__enumext_joined_width_viii_dim
4927         + \l__enumext_labelwidth_viii_dim
4928         + \l__enumext_labelsep_viii_dim
4929     }
4930     \IfDocumentMetadataTF { }
4931     {
4932         \bool_if:NF \l__enumext_footnotes_key_bool
4933         {
4934             \__enumext_renew_footnote:
4935         }
4936     }
4937     \__enumext_start_list_tag:n {keyans*}
4938     \__enumext_fake_make_label_viii:n {#1}
4939     \__enumext_stop_start_list_tag:
4940     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4941     \tl_use:N \l__enumext_after_list_args_viii_tl
4942     \bool_if:NT \l__enumext_item_starred_viii_bool
4943     {
4944         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4945         \__enumext_keyans_show_item_opt:
4946         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4947     }
4948     {
4949         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4950     }
4951     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4952     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4953 }

```

(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.

```

4954 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4955 {
4956     \__enumext_endminipage:
4957     \__enumext_stop_list_tag:n {keyans*}
4958     \hbox_set_end:
4959     \int_set:Nn \hbadness { 10000 }
4960     \box_use_drop:N \l__enumext_item_text_viii_box
4961     \IfDocumentMetadataTF { }
4962     {
4963         \bool_if:NF \l__enumext_footnotes_key_bool
4964         {
4965             \__enumext_print_footnote:
4966         }
4967     }
4968     \int_compare:nNnTF
4969     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }

```



```

4970     {
4971       \par\noindent
4972       \int_zero:N \l__enumext_item_column_pos_viii_int
4973     }
4974     {
4975       \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4976     }
4977   }

```

(End of definition for `\l__enumext_stop_item_viii:`)

`\l__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.

```

4978 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_viii:
4979 {
4980   \int_compare:nNnT
4981   {
4982     \int_mod:nn
4983     { \g__enumext_item_count_all_viii_int }
4984     { \l__enumext_columns_viii_int }
4985   }
4986   =
4987   { 0 }
4988   {
4989     \para_end:
4990     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4991     \skip_vertical:N \c_zero_skip
4992     \int_gzero:N \g__enumext_item_count_all_viii_int
4993   }
4994 }

```

(End of definition for `\l__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 $\langle prop\ list \rangle$ defined by `save-ans` key.

```

4995 \NewDocumentCommand \getkeyans { m }
4996 {
4997   \exp_args:Ne \l__enumext_getkeyans_aux:n
4998   { \tl_to_str:e { \text_expand:n {#1} } }
4999 }

```

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

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

```

5000 \cs_new_protected:Npn \l__enumext_getkeyans_aux:n #1
5001 {
5002   \str_if_in:nnTF {#1} { : }
5003   {
5004     \use:e
5005     {
5006       \cs_set:Npn \exp_not:N \l__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5007       { {##1} {##2} }
5008     }
5009     \exp_after:wN \l__enumext_getkeyans:nn \l__enumext_tmp:w #1 \scan_stop:
5010   }
5011   { \msg_error:nnn { enumext } { missing-colon } {#1} }
5012 }

```

(End of definition for `\l__enumext_getkeyans_aux:n`)

`\l__enumext_getkeyans:nn` The internal function `\l__enumext_getkeyans:nn` will check for the existence of the $\langle prop\ list \rangle$, 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 $\langle prop\ list \rangle$.

```

5013 \cs_new_protected:Npn \l__enumext_getkeyans:nn #1 #2
5014 {
5015   \prop_if_exist:ctF { g__enumext_#1_prop }
5016   {
5017     \prop_item:cn { g__enumext_#1_prop }{#2}

```

```

5018     }
5019     {
5020         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5021     }
5022 }

```

(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 *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default *keys* for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[⟨print,*⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print,level⟩]`.

```

5023 \keys_define:nn { enumext / print }
5024 {
5025     print* .code:n = \keys_precompile:neN { enumext / enumext* }
5026                 { \__enumext_filter_save_key:n {#1} }
5027                 \l__enumext_print_keyans_starred_tl, % starred cmd
5028     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5029     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5030                     { \__enumext_filter_save_key:n {#1} }
5031                     \l__enumext_print_keyans_i_tl,
5032     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5033     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5034                     { \__enumext_filter_save_key:n {#1} }
5035                     \l__enumext_print_keyans_ii_tl,
5036     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5037     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5038                     { \__enumext_filter_save_key:n {#1} }
5039                     \l__enumext_print_keyans_iii_tl,
5040     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5041     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
5042                     { \__enumext_filter_save_key:n {#1} }
5043                     \l__enumext_print_keyans_iv_tl,
5044     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5045     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
5046                     { \__enumext_filter_save_key:n {#1} }
5047                     \l__enumext_print_keyans_vii_tl, % starred nested
5048     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5049 }

```

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

`\printkeyans` Create a user command to print “all stored content” in *sequence* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

5050 \NewDocumentCommand \printkeyans { s O{} m }
5051 {
5052     \group_begin:
5053         \tl_use:N \l__enumext_print_keyans_i_tl
5054         \tl_use:N \l__enumext_print_keyans_ii_tl
5055         \tl_use:N \l__enumext_print_keyans_iii_tl
5056         \tl_use:N \l__enumext_print_keyans_iv_tl
5057         \tl_use:N \l__enumext_print_keyans_vii_tl
5058         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5059     \group_end:
5060 }

```

(End of definition for `\printkeyans`. This function is documented on page 18.)

`__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.

```

5061 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3

```

```

5062 {
5063   \seq_if_exist:cTF { g__enumext_#3_seq }
5064   {
5065     \seq_if_empty:cF { g__enumext_#3_seq }
5066     {

```

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 *keys* 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*.

```

5067     \bool_if:nTF {#1}
5068     {
5069       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5070       {
5071         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5072       }
5073       {
5074         \tl_use:N \__enumext_print_keyans_starred_tl
5075         \bool_set_true:N \__enumext_base_line_fix_bool
5076         \bool_set_true:N \__enumext_print_keyans_star_bool
5077         \begin{enumext*}[#2]
5078           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5079         \end{enumext*}
5080       }
5081     }

```

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

```

5082     {
5083       \begin{enumext}[#2]
5084       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5085       \end{enumext}
5086     }
5087   }
5088 }
5089 {
5090   \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5091 }
5092 }

```

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

12.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the *keys* passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “*first level*” of the `enumext` environment so as not to capture *keys* that complicate us.

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

```

5093 \cs_new:Npn \__enumext_filter_first_level:n #1
5094 {
5095   \use:e
5096   {
5097     \keyval_parse:NNn
5098     \__enumext_filter_first_level_key:n
5099     \__enumext_filter_first_level_pair:nn {#1}
5100   }
5101 }

```

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

```

5102 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5103 {
5104   \str_case:nnF {#1}
5105   {
5106     { resume } {}
5107     { resume* } {}
5108   }
5109   { , { \exp_not:n {#1} } }
5110 }

```

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

```

5111 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5112 {
5113   \str_case:nnF {#1}
5114   {
5115     { series } {}
5116     { resume } {}
5117     { save-ans } {}
5118   }
5119   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5120 }

```

(End of definition for `__enumext_filter_first_level:n`, `__enumext_filter_first_level_key:n`, and `__enumext_filter_first_level_pair:nn`.)

Now define a “meta families” of *(keys)* to access from `\setenumext`.

```

5121 \keys_define:nn { enumext / meta-families }
5122 {
5123   enumext-1 .code:n =
5124   {
5125     \keys_set:ne { enumext / level-1 }
5126     {
5127       \__enumext_filter_first_level:n {#1}
5128     }
5129   } ,
5130   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5131   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5132   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5133   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5134   enumext* .code:n =
5135   {
5136     \keys_set:ne { enumext / enumext* }
5137     {
5138       \__enumext_filter_first_level:n {#1}
5139     }
5140   } ,
5141   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5142   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5143   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5144   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5145   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5146   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5147   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5148   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5149 }

```

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

```

5150 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5151 {
5152   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5153   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5154 }

```

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

```

5155 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5156 {
5157   \seq_clear:N \l__enumext_setkey_tmpa_seq
5158   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5159   \int_set:Nn \l__enumext_setkey_tmpa_int
5160   {
5161     \seq_count:N \l__enumext_setkey_tmpb_seq
5162   }
5163   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5164   {
5165     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5166     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5167     \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5168     {
5169       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5170     }

```

```

5171     }
5172     {
5173         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5174     }
5175     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5176     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5177     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5178     {
5179         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5180     }
5181 }

```

(End of definition for `\setenumext`. This function is documented on page 7.)

```

\__enumext_set_parse:n
\__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

5182 \cs_new_protected:Npn \__enumext_set_parse:n #1
5183 {
5184     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5185     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5186     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5187     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5188     {
5189         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5190         { \tl_trim_spaces:n {#1} }
5191     }
5192     { \__enumext_set_error:nn {#1} { } }
5193 }
5194 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5195 { \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

```

```

5196 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5197 {
5198     {enumext,1} = level-1,
5199     {enumext,2} = level-2,
5200     {enumext,3} = level-3,
5201     {enumext,4} = level-4,
5202     {enumext*} = enumext*
5203 }

```

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

```

5204 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5205 {
5206     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5207     { \msg_error:nn { enumext } { prohibited-unknown } }
5208     {
5209         \bool_if:NTF {#1}
5210         {
5211             \int_step_inline:nn { 4 }
5212             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5213             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5214         }
5215         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5216     }
5217 }

```

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

```

5218 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5219 {
5220     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5221     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5222     \prop_get:NVNTF
5223     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5224     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }

```

```

5225     {
5226         \msg_error:nnn { enumext } { unknown-set } {#1}
5227         \use_none:nn
5228     }
5229 }
5230 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5231 {
5232     \bool_lazy_or:nnTF
5233     { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5234     { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5235     { \msg_error:nnn { enumext } { already-defined } {#2} }
5236     {
5237         \keys_define:nn { enumext / #1 }
5238         {
5239             #2 .meta:n = {#3},
5240             #2 .value_forbidden:n = true
5241         }
5242     }
5243 }
5244 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

(End of definition for `\setenumextmeta` and others. This function is documented on page 7.)

12.49 The command `\foreachkeys`

The command `\foreachkeys` 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](#).

`\foreachkeys`

```

\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
\__enumext_foreach_keys: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.

```

5245 \keys_define:nn { enumext / foreach }
5246 {
5247     before .tl_set:N = \l__enumext_foreach_before_tl,
5248     before .value_required:n = true,
5249     after .tl_set:N = \l__enumext_foreach_after_tl,
5250     after .value_required:n = true,
5251     start .int_set:N = \l__enumext_foreach_start_int,
5252     start .value_required:n = true,
5253     stop .int_set:N = \l__enumext_foreach_stop_int,
5254     stop .value_required:n = true,
5255     step .int_set:N = \l__enumext_foreach_step_int,
5256     step .value_required:n = true,
5257     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5258     wrapper .value_required:n = true,
5259     sep .tl_set:N = \l__enumext_foreach_sep_tl,
5260     sep .value_required:n = true,
5261     unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5262 }
5263 \keys_precompile:nnN { enumext / foreach }
5264 {
5265     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5266 }
5267 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown *keys*.

```

5268 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5269 {
5270     \tl_if_blank:nTF {#2}
5271     {
5272         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5273     }
5274     {
5275         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5276     }
5277 }
5278 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5279 {
5280     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5281 }

```

We create the command.

```

5282 \NewDocumentCommand \foreachkeyans { +0{ } m }
5283 {
5284   \__enumext_foreach_keyans:nn {#1} {#2}
5285 }

```

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

```

5286 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5287 {
5288   \tl_use:N \g__enumext_foreach_default_keys_tl
5289   \keys_set:nn { enumext / foreach } {#1}
5290   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5291   \prop_if_exist:cF { g__enumext_#2_prop }
5292   {
5293     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5294   }
5295   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5296   {
5297     \int_set:Nn \l__enumext_foreach_stop_int
5298     { \prop_count:c { g__enumext_#2_prop } }
5299   }
5300   \seq_clear:N \l__enumext_foreach_print_seq
5301   \int_step_function:nnnN
5302   { \l__enumext_foreach_start_int }
5303   { \l__enumext_foreach_step_int }
5304   { \l__enumext_foreach_stop_int }
5305   \__enumext_foreach_add_body:n
5306   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5307 }
5308 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5309 {
5310   \seq_put_right:Ne \l__enumext_foreach_print_seq
5311   {
5312     \exp_not:V \l__enumext_foreach_before_tl
5313     \__enumext_foreach_wrapper:n
5314     {
5315       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5316     }
5317     \exp_not:V \l__enumext_foreach_after_tl
5318   }
5319 }

```

(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.

```

5320 \msg_new:nnn { enumext } { package-load }
5321 {
5322   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5323 }
5324 \msg_new:nnn { enumext } { package-not-load }
5325 {
5326   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5327 }
5328 \msg_new:nnn { enumext } { package-load-foot }
5329 {
5330   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5331 }

```

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

```

5332 \msg_new:nnn { enumext } { counters }
5333 {
5334   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \
5335   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5336 }

```

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

```

5337 \msg_new:nnn { enumext } { unknown-choice }
5338 {
5339   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5340 }

```


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

```
5341 \msg_new:nnn { enumext } { anskey-env-error }
5342 {
5343   The ~ '#1' ~ environment ~is ~ reserved ~ by ~\
5344   'enumext' ~ package, ~ It~ is~ already~ defined.
5345 }
5346 {
5347   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5348   for ~ the ~ 'save-ans' ~ key.\
5349 }
```

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

```
5350 \msg_new:nnn { enumext } { store-prop }
5351 {
5352   * ~ Package ~ enumext: ~ Creating ~
5353   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5354 }
5355 \msg_new:nnn { enumext } { store-seq }
5356 {
5357   * ~ Package ~ enumext: ~ Creating ~
5358   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5359 }
5360 \msg_new:nnn { enumext } { store-int }
5361 {
5362   * ~ Package ~ enumext: ~ Creating ~
5363   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5364 }
5365 \msg_new:nnn { enumext } { prop-seq-int-hook }
5366 {
5367   * ~ Package ~ enumext: ~ Elements ~ in ~
5368   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\
5369   * ~ Package ~ enumext: ~ Elements ~ in ~
5370   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\
5371   * ~ Package ~ enumext: ~ Value ~ off ~
5372   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5373 }
5374 \msg_new:nnn { enumext } { item-answer-hook }
5375 {
5376   * ~ Package ~ enumext: ~ Value ~ off ~
5377   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
5378   * ~ Package ~ enumext: ~ Value ~ off ~
5379   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
5380   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5381 }
```

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

```
5382 \msg_new:nnn { enumext } { invalid-key }
5383 {
5384   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5385 }
5386 \msg_new:nnn { enumext } { unknown-key-family }
5387 {
5388   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5389 }
```

Messages used in length calculation.

```
5390 \msg_new:nnn { enumext } { width-negative }
5391 {
5392   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5393   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
5394 }
5395 \msg_new:nnn { enumext } { width-zero }
5396 {
5397   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5398   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
5399 }
```

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

```
5400 \msg_new:nnn { enumext } { list-lengths }
5401 {
5402   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
5403   \__enumext_show_length:nnn { dim } { labelsep } { #1}
5404   \__enumext_show_length:nnn { dim } { labelwidth } { #1}
```

```

5405     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5406     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5407     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5408     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5409     \__enumext_show_length:nnn { skip } { topsep } {#1}
5410     \__enumext_show_length:nnn { skip } { parsep } {#1}
5411     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5412     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5413     *****
5414 }

```

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

```

5415 \msg_new:nnn { enumext } { list-lengths-not-nested }
5416 {
5417     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5418     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5419     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5420     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5421     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5422     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5423     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5424     \__enumext_show_length:nnn { skip } { topsep } {#1}
5425     \__enumext_show_length:nnn { skip } { parsep } {#1}
5426     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5427     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5428     *****
5429 }

```

Messages used by `ref` key.

```

5430 \msg_new:nnn { enumext } { key-ref-empty }
5431 {
5432     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5433 }

```

Messages used by `save-ans` key.

```

5434 \msg_new:nnn { enumext } { save-ans-empty }
5435 {
5436     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5437 }
5438 \msg_new:nnn { enumext } { save-ans-log }
5439 {
5440     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5441 }
5442 \msg_new:nnn { enumext } { save-ans-log-hook }
5443 {
5444     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5445 }
5446 \msg_new:nnn { enumext } { save-ans-hook }
5447 {
5448     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5449 }

```

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

```

5450 \msg_new:nnn { enumext } { need-save-ans }
5451 {
5452     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5453 }
5454 \msg_new:nnn { enumext } { items-same-answer }
5455 {
5456     *****\\
5457     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5458     for ~ \c_left_brace_str #2 \c_right_brace_str\\
5459     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5460     'OK', ~ all ~ items ~ with ~ answer.\\
5461     *****
5462 }
5463 \msg_new:nnn { enumext } { item-greater-answer }
5464 {
5465     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5466     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5467     Items ~ > ~ Answers.
5468 }
5469 \msg_new:nnn { enumext } { item-less-answer }

```

```

5470 {
5471     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5472     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5473     Items ~ < ~ Answers.
5474 }

```

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

```

5475 \msg_new:nnn { enumext } { missing-starred }
5476 {
5477     Missing ~ '\c_backslash_str #1*' ~ #2.
5478 }
5479 \msg_new:nnn { enumext } { many-starred }
5480 {
5481     Many ~ '\c_backslash_str #1*' ~ #2.
5482 }

```

Messages used by \printkeyans* command.

```

5483 \msg_new:nnn { enumext } { print-starred }
5484 {
5485     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5486     #2 ~ environment ~ \msg_line_context:.
5487 }

```

Message for the nesting depth of the environment enumext.

```

5488 \msg_new:nnn { enumext } { list-too-deep }
5489 {
5490     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
5491     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5492 }

```

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

```

5493 \msg_new:nnn { enumext } { anskey-unnumber-item }
5494 {
5495     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5496 }
5497 \msg_new:nnn { enumext } { anskey-already-stored }
5498 {
5499     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5500 }
5501 \msg_new:nnn { enumext } { anskey-empty-arg }
5502 {
5503     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5504 }
5505 \msg_new:nnn { enumext } { anskey-wrong-place }
5506 {
5507     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
5508     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5509 }
5510 \msg_new:nnn { enumext } { anskey-nested }
5511 {
5512     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5513 }
5514 \msg_new:nnn { enumext } { anskey-math-mode }
5515 {
5516     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5517 }
5518 \msg_new:nnn { enumext } { anskey-env-wrong }
5519 {
5520     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5521 }
5522 \msg_new:nnn { enumext } { anspic-wrong-place }
5523 {
5524     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
5525     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5526 }
5527 \msg_new:nnn { enumext } { command-wrong-place }
5528 {
5529     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
5530     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5531 }
5532 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5533 {
5534     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~

```

```

5535     'anskey*' ~ and ~ is ~ being ~ ignored.
5536 }
5537 {
5538     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5539     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5540 }
5541 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5542 {
5543     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5544     'anskey*' ~ and ~ is ~ being ~ ignored.
5545 }
5546 {
5547     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5548     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5549 }
5550 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5551 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored. }
5552 {
5553     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5554     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5555 }
5556 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5557 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored. }
5558 {
5559     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5560     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5561 }

```

Messages used by [keyans](#), [keyans*](#) and [keyanspic](#) environment.

```

5562 \msg_new:nnn { enumext } { keyans-nested }
5563 {
5564     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5565 }
5566 \msg_new:nnn { enumext } { keyans-wrong-level }
5567 {
5568     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5569     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5570 }
5571 \msg_new:nnn { enumext } { wrong-place }
5572 {
5573     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5574     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5575 }
5576 \msg_new:nnn { enumext } { keyanspic-nested }
5577 {
5578     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5579 }
5580 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5581 {
5582     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5583     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5584 }
5585 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5586 {
5587     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5588 }
5589 \msg_new:nnnn { enumext } { keyans-unknown-key }
5590 {
5591     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5592     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5593 }
5594 {
5595     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5596     ~ have ~ a ~ key ~ called ~'#1'.\\
5597     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5598 }
5599 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5600 {
5601     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5602     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5603 }
5604 {

```

```

5605     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5606     ~ have ~ a ~ key ~ called ~ '#1'.\\
5607     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5608 }

```

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

```

5609 \msg_new:nnnn { enumext } { starred-unknown-key }
5610 {
5611     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5612     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5613 }
5614 {
5615     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5616     ~ have ~ a ~ key ~ called ~ '#1'.\\
5617     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5618 }
5619 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5620 {
5621     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5622     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5623 }
5624 {
5625     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5626     ~ have ~ a ~ key ~ called ~ '#1'.\\
5627     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5628 }

```

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

```

5629 \msg_new:nnnn { enumext } { standar-unknown-key }
5630 {
5631     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5632     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5633 }
5634 {
5635     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5636     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5637     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5638 }
5639 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5640 {
5641     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5642     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5643 }
5644 {
5645     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5646     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5647     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5648 }

```

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

```

5649 \msg_new:nnnn { enumext } { for-key-unknown }
5650 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5651 {
5652     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5653     Check~that~you~have~spelled~the~key~name~correctly.
5654 }
5655 \msg_new:nnnn { enumext } { for-key-value-unknown }
5656 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5657 {
5658     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5659     Check~that~you~have~spelled~the~key~name~correctly.
5660 }

```

Messages used by `\getkeyans` command.

```

5661 \msg_new:nnn { enumext } { undefined-storage-anskey }
5662 {
5663     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5664 }

```

Messages used by `\miniright` command.

```

5665 \msg_new:nnn { enumext } { missing-miniright }
5666 {
5667     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\

```

```

5668     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5669   }
5670   \msg_new:nnn { enumext } { wrong-miniright-place }
5671   {
5672     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5673     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5674   }
5675   \msg_new:nnn { enumext } { wrong-miniright-use }
5676   {
5677     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5678     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5679   }
5680   \msg_new:nnn { enumext } { wrong-miniright-starred }
5681   {
5682     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5683   }
5684   \msg_new:nnn { enumext } { many-miniright-used }
5685   {
5686     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5687   }

```

Messages used by `\setenumextmeta` command.

```

5688   \msg_new:nnn { enumext } { unknown-set }
5689   {
5690     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5691   }
5692   \msg_new:nnn { enumext } { already-defined }
5693   {
5694     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5695   }
5696   \msg_new:nnn { enumext } { prohibited-unknown }
5697   {
5698     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5699   }

```

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

```

5700   \msg_new:nnn { enumext } { nested }
5701   {
5702     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5703   }
5704   \msg_new:nnn { enumext } { nested-horizontal }
5705   {
5706     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5707   }
5708   \msg_new:nnn { enumext } { item-joined }
5709   {
5710     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5711   }
5712   \msg_new:nnn { enumext } { item-joined-columns }
5713   {
5714     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5715   }

```

12.51 Finish package

Finish package implementation.

```

5716   \file_input_stop:
5717   </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>	227
<code>\+</code>	219
<code>\-</code>	219
<code>\\</code>	235, 2794, 4114, 5334, 5343, 5348, 5368, 5370, 5377, 5379, 5392, 5397, 5402, 5417, 5456, 5458, 5460, 5465, 5466, 5471, 5472, 5490, 5507, 5524, 5529, 5538, 5547, 5553, 5559, 5568, 5573, 5582, 5596, 5606, 5616, 5626, 5636, 5646, 5652, 5658, 5667, 5672, 5677
A	
<code>above</code>	<u>1608</u>
<code>above*</code>	<u>1608</u>
<code>\addvspace</code>	1177, 1206, 1249, 1252, 1420, 1423, 1520, 1526, 1561, 1567, 1588, 1594, 3603, 3764, 3782, 4015, 4018, 4348, 4363, 4409, 4423
<code>after</code>	<u>1006</u>
<code>align</code>	<u>556</u>
<code>\Alph</code>	39, 43, 44
<code>\Alph</code>	508, 626, 671, 739, 5044
<code>\alph</code>	39, 43, 44
<code>\alph</code>	509, 624, 5036
<code>\anskey</code>	13, 77, 79, <u>2612</u>
<code>anskey*</code>	14, <u>2722</u>
<code>\anspic</code>	16, 105, 108, <u>4022</u>
<code>\anspic*</code>	71
<code>\arabic</code>	32, 39
<code>\arabic</code>	507, 623, 670, 5028, 5032, 5048
B	
<code>base-fix</code>	<u>869</u>
<code>\baselineskip</code>	52
<code>\baselineskip</code>	883, 889
<code>before</code>	<u>1006</u>
<code>before*</code>	<u>1006</u>
<code>below</code>	<u>1608</u>
<code>below*</code>	<u>1608</u>
bool commands:	
<code>\bool_gset_false:N</code>	377, 378, 379, 2898, 2900, 4365, 4369, 4425
<code>\bool_gset_true:N</code>	285, 295, 1109, 2101, 2107, 4334, 4366, 4398, 4426
<code>\bool_if:NTF</code>	447, 459, 476, 1542, 1630, 1644, 1657, 1668, 1679, 1690, 1701, 1712, 1761, 1778, 1783, 1791, 1818, 1856, 1861, 1868, 1872, 1894, 1899, 1907, 1914, 1945, 1953, 2046, 2244, 2254, 2333, 2357, 2364, 2388, 2486, 2508, 2548, 2562, 2566, 2616, 2635, 2659, 2713, 2724, 2813, 2850, 2914, 2949, 2964, 3039, 3050, 3054, 3073, 3086, 3128, 3162, 3205, 3224, 3367, 3382, 3444, 3454, 3487, 3492, 3558, 3584, 3634, 3692, 3747, 3772, 3947, 4013, 4024, 4043, 4088, 4327, 4343, 4349, 4392, 4406, 4410, 4532, 4542, 4626, 4632, 4639, 4655, 4674, 4697, 4759, 4769, 4876, 4880, 4906, 4913, 4932, 4942, 4963
<code>\bool_if:nTF</code>	1568, 1595, 3184, 3341, 3402, 3926, 4064, 5067, 5209
<code>\bool_if_p:N</code>	304, 319, 880, 886, 887, 1925, 1926, 1934, 1935, 2059, 2085, 2098, 2099, 2104, 2105, 2421, 2431, 2443, 2458, 2459, 2493, 2534, 2535, 2836, 3026, 3027, 3064, 3065, 3531, 3533, 3544, 4071, 4072
<code>\bool_lazy_all:nTF</code>	302, 317, 2057, 2083, 2419, 2428, 2441, 2456, 3529, 3542
<code>\bool_lazy_and:nnTF</code>	281, 291, 879, 885, 1535, 1924, 1933, 2097, 2103, 2492, 2499, 2533, 2677, 2689, 2835, 2841, 3025
<code>\bool_lazy_or:nnTF</code>	1986, 1993, 3063, 4070, 5232
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 97, 102, 103, 108, 109, 112, 137, 138, 145, 152, 153, 158, 160, 161, 175, 187, 189, 875
<code>\bool_not_p:n</code>	282, 292, 881, 2430, 2494, 2500, 2837, 2842, 3532, 3545
<code>\bool_set_eq:NN</code>	3137, 3320, 4579, 4829
<code>\bool_set_false:N</code>	456, 896, 2031, 2032, 2064, 2069, 2073, 2077, 2090, 2777, 3506, 3651, 3700, 3787, 3944, 4020, 4501, 4527, 4576, 4775, 4826
<code>\bool_set_true:N</code>	309, 310, 324, 325, 436, 440, 549, 911, 1614, 1619, 1881, 2003, 2004, 2276, 2284, 2778, 3131, 3133, 3165, 3167, 3316, 3328, 3467, 3505, 3538, 3551, 3624, 3697, 3724, 3928, 4316, 4381, 4500, 4583, 4590, 4591, 4635, 4773, 4833, 4840, 4841, 5075, 5076
box commands:	
<code>\box_dp:N</code>	1466, 1467, 1470, 1477, 1490, 1498, 1504, 1512, 3957, 3962, 4015, 4099
<code>\box_ht:N</code>	1249, 1252, 1263, 1264, 1275, 1277, 1292, 1295, 1303, 1304, 1315, 1317, 1332, 1335, 1342, 1343, 1354, 1356, 1371, 1374, 1420, 1423, 1431, 1432, 1440, 1441, 1453, 1455
<code>\box_ht_plus_dp:N</code>	3953, 4052
<code>\box_new:N</code>	70, 148, 149, 182, 188
<code>\box_use_drop:N</code>	4360, 4421, 4694, 4960
<code>\box_wd:N</code>	515
C	
<code>\c</code>	227, 228, 776, 778, 790, 792
<code>\catcode</code>	2794
<code>\cB</code>	228
<code>\cE</code>	228
<code>\centering</code>	1570, 1597, 4136, 4353, 4414
<code>check-ans</code>	<u>2023</u>
Document class:	
<code>article</code>	45
clist commands:	
<code>\clist_const:Nn</code>	194
<code>\clist_map_function:nN</code>	4119
<code>\clist_map_inline:Nn</code>	555, 824, 1005, 1020, 1101, 1624
<code>\clist_map_inline:nn</code>	49, 60, 78, 86, 99, 111, 140, 169, 193, 586, 606, 916, 937, 1115, 1730, 1970, 2037, 2223, 2241, 2273, 2416, 2958, 3245, 3257, 3297, 3431, 3434, 3462, 3474, 3477, 3497, 5185
<code>\columnbreak</code>	77
<code>\columnbreak</code>	2496
<code>columns</code>	<u>1085</u>
<code>columns-sep</code>	<u>1085</u>
<code>\columnsep</code>	99
<code>\columnsep</code>	3579, 3745
<code>\columnseprule</code>	99
<code>\columnseprule</code>	3582, 3746
Commands provide by enumext:	
<code>\anskey</code>	30, 67, 68, 73, 74, 76, 78, 79, 86, 88, 98, 117, 127,

128, 136	
\anspic*	30, 31, 71, 74, 86, 87, 107, 108, 127, 128
\anspic	30, 74, 105–108, 136
\foreachkeyans	132, 138
\getkeyans	74, 127, 138
\item* 30, 31, 71, 74, 86, 87, 89, 90, 93, 119, 124, 125, 127, 128	
\item	89, 93, 111, 118, 120, 123, 124
\miniright	29, 50, 58, 59, 100, 138
\printkeyans*	128
\printkeyans	30, 74, 128
\setenumextmeta	131, 139
\setenumext	30, 128, 130, 131, 134
Counters defined by enumext:	
enumXiii	28, 38
enumXii	28, 38
enumXiv	28, 38
enumXi	28, 38
enumXviii	28, 38
enumXvii	28, 38, 119
enumXvi	28, 38
enumXv	28, 38
cs commands:	
\cs_generate_variant:Nn	199, 200, 517, 533, 782, 798, 2325, 2330, 2406, 2730, 3421, 4121, 5244
\cs_if_exist:NTF	487
\cs_if_free:NTF	2681, 2693
\cs_new:Nn	213
\cs_new:Npn	231, 1731, 1740, 1748, 2288, 2297, 2305, 5093, 5102, 5111
\cs_new_eq:NN	404, 405, 410, 411, 461, 462, 465, 466
\cs_new_protected:Nn	223, 237, 254, 274, 300, 333, 363, 369, 375, 381, 387, 395, 413, 431, 647, 710, 762, 876, 1021, 1025, 1029, 1033, 1037, 1041, 1045, 1049, 1053, 1057, 1061, 1065, 1069, 1073, 1077, 1081, 1116, 1128, 1161, 1179, 1190, 1208, 1234, 1255, 1380, 1406, 1426, 1459, 1481, 1516, 1522, 1625, 1639, 1653, 1664, 1675, 1686, 1697, 1708, 1789, 1892, 1905, 1922, 1943, 1971, 1976, 2001, 2042, 2052, 2095, 2110, 2117, 2126, 2131, 2136, 2141, 2150, 2155, 2160, 2331, 2355, 2362, 2386, 2393, 2407, 2633, 2652, 2668, 2731, 2767, 2798, 2833, 2875, 2896, 2904, 2947, 2962, 2990, 3023, 3059, 3071, 3084, 3170, 3180, 3191, 3199, 3215, 3337, 3353, 3361, 3375, 3498, 3527, 3556, 3563, 3593, 3610, 3632, 3654, 3690, 3714, 3731, 3756, 3770, 3791, 3942, 4109, 4117, 4122, 4146, 4177, 4306, 4325, 4371, 4390, 4430, 4434, 4453, 4487, 4514, 4521, 4530, 4540, 4561, 4712, 4757, 4788, 4794, 4811, 4868, 4978
\cs_new_protected:Npn	201, 205, 209, 469, 485, 502, 512, 518, 627, 672, 744, 769, 783, 1552, 1581, 1757, 1776, 1846, 1879, 1981, 2165, 2242, 2252, 2274, 2282, 2317, 2326, 2482, 2545, 2560, 2598, 2602, 2722, 2753, 2757, 2788, 2924, 3000, 3044, 3124, 3143, 3258, 3262, 3276, 3280, 3298, 3302, 3312, 3324, 3390, 3424, 3465, 3509, 3710, 3918, 3935, 4041, 4060, 4084, 4208, 4257, 4504, 4567, 4574, 4588, 4596, 4601, 4611, 4781, 4817, 4824, 4838, 4846, 4863, 5000, 5013, 5061, 5182, 5194, 5218, 5230, 5268, 5278, 5286, 5308
\cs_new_protected_nopar:Nn	3852, 3894, 3902, 3910, 4550, 4554, 4688, 4800, 4804, 4954
\cs_new_protected_nopar:Npn	3844, 3860, 4617, 4663, 4896, 4921
\cs_set:Npn	2417, 2454, 5006
\cs_set_eq:NN	4477, 4478, 4665, 4746, 4747, 4923
\cs_set_protected:Nn	942, 958, 971, 984
\cs_set_protected:Npn	45, 54, 71, 79, 94, 100, 133, 165, 173, 534, 556, 591, 607, 654, 799, 825, 898, 921, 997, 1006, 1085, 1102, 1608, 1719, 1962, 2023, 2182, 2224, 2260, 2409, 2951, 3234, 3250, 3290, 3422, 3463
\cs_to_str:N	504, 527
\cs_undefine:N	2670, 2671, 2672, 2673
D	
\d	219
\DeclareDocumentEnvironment	417
dim commands:	
\dim_abs:n	3395, 3400
\dim_add:Nn	3961, 4171, 4202
\dim_compare:nNnTF	944, 960, 973, 986, 1267, 1279, 1307, 1319, 1346, 1358, 1435, 1443, 1554, 1583, 3392, 3397, 3403, 3409, 3411, 3413, 3568, 3615, 3718, 3735, 3937, 4148, 4164, 4179, 4195, 4308, 4373
\dim_compare:nTF	2518, 2863, 3657, 3794
\dim_eval:n	4095
\dim_gset_eq:NN	4317, 4382
\dim_gzero:N	2902, 4368, 4428
\dim_new:N	67, 74, 75, 76, 96, 142, 150, 151, 181, 183, 184, 190
\dim_set:Nn	515, 912, 3160, 3395, 3400, 3402, 3405, 3406, 3410, 3412, 3415, 3416, 3418, 3571, 3618, 3656, 3720, 3737, 3793, 3951, 4050, 4124, 4150, 4157, 4181, 4188, 4243, 4292, 4310, 4375, 4613
\dim_set_eq:NN	614, 661, 732, 736, 3075, 3076, 3088, 3089, 3155, 3433, 3476, 3579, 3745, 4250, 4253, 4254, 4299, 4302, 4303, 4606, 4685, 4951
\dim_sub:Nn	3662, 3799, 4166, 4197
\dim_use:N	945, 953, 1555, 1565, 2396, 2399, 2404, 3175, 3177, 3220, 3569, 3573, 3574, 3576, 3616, 3621, 3622, 3628, 3659, 3664
\dim_zero:N	3468, 3582, 3746, 3963, 3964, 3965
\dim_zero_new:N	484
\c_zero_dim	947, 961, 974, 987, 1555, 1583, 2520, 2865, 3392, 3397, 3403, 3410, 3569, 3616, 3659, 3718, 3735, 3796, 3937, 4148, 4164, 4179, 4195, 4308, 4373
\dimeval	883, 2189
E	
\end	2359, 2390, 3600, 3761, 4005, 4138, 5069, 5079, 5085
end internal commands:	
\end__enumext_mini_page	1563, 1590, 3643, 3781, 4332, 4396, 4422
\endgroup	2794
\endlist	405
\endminipage	411
enumext	6, 3668
enumext internal commands:	
\l__enumext__ref_the_count_tl	41
\l__enumext__resume_name_tl	63
__enumext_add_meta_key:nnn	131, 5196, 5212, 5213, 5215, 5218
__enumext_add_pre_parsep:	51, 1126, 1128, 1128
__enumext_after_args_exec:	48, 1021, 1033, 3681
__enumext_after_args_exec_v:	1037, 1049, 3814
__enumext_after_args_exec_vii:	1053, 1077
__enumext_after_args_exec_viii:	1081
__enumext_after_env:nn	83–85, 101, 113, 121, 205, 205, 2808, 3686, 4341, 4404, 4728
__enumext_after_hyperref:	37, 429, 431, 431
\l__enumext_after_list_args_v_tl	1051
\l__enumext_after_list_args_vii_tl	1079, 4683

```

\l__enumext_after_list_args_viii_tl .. 1083,
    4941
\__enumext_after_list_vii: 113, 117, 4485, 4521,
    4521
\__enumext_after_list_viii: ... 123, 4755, 4794,
    4794
\__enumext_after_stop_list: 48, 100, 1021, 1029,
    3648
\__enumext_after_stop_list_v: 1037, 1045, 3788
\l__enumext_after_stop_list_v_tl ..... 1047
\__enumext_after_stop_list_vii: .. 117, 1053,
    1069, 4524
\l__enumext_after_stop_list_vii_tl ... 1071
\__enumext_after_stop_list_viii: . 1073, 4797
\l__enumext_after_stop_list_viii_tl ... 1075
\l__enumext_align_label_pos_v_str .... 3379
\l__enumext_align_label_pos_X_str ..... 79
\l__enumext_align_label_vii_str ..... 4652
\l__enumext_align_label_viii_str ..... 4910
\l__enumext_align_label_X_str ..... 173
\c__enumext_all_envs_clist . 194, 555, 824, 1005,
    1020, 1101, 1624
\c__enumext_all_families_seq .. 130, 5150, 5176
\l__enumext_anskey_env_bool 34, 82, 34, 310, 325,
    2724
\__enumext_anskey_env_clean_vars: . 85, 2829,
    2833, 2896
\__enumext_anskey_env_define_keys: 82, 2722,
    2731, 2802
\__enumext_anskey_env_exec: 83, 2727, 2798, 2798
\__enumext_anskey_env_make:n 67, 82, 2006, 2722,
    2722, 2730
\__enumext_anskey_env_reset_keys: 83, 84, 2767,
    2830
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2722
\__enumext_anskey_env_save_keys: .. 84, 2810,
    2833, 2833
\__enumext_anskey_env_store: .. 84, 2826, 2833,
    2875
\__enumext_anskey_env_unknown:n 82, 2750, 2753
\__enumext_anskey_env_unknown:nn . 2755, 2757
\l__enumext_anskey_level_int .. 28, 2654, 2655
\__enumext_anskey_safe_inner: . 80, 2627, 2633,
    2652
\__enumext_anskey_safe_inner:n ..... 80
\__enumext_anskey_safe_outer: . 79, 2614, 2633,
    2633
\__enumext_anskey_show_wrap_arg:n . 78, 2545,
    2545, 2564, 2579
\__enumext_anskey_show_wrap_left:n 78, 2490,
    2560, 2560
\__enumext_anskey_unknown:n 79, 2582, 2596, 2598
\__enumext_anskey_unknown:nn . 2582, 2600, 2602
\__enumext_anskey_wrapper:n ..... 2186, 2558
\l__enumext_anspic_above_int . 141, 4125, 4126,
    4128
\__enumext_anspic_args:nnn 108, 109, 4038, 4109,
    4109
\l__enumext_anspic_args_seq 107-110, 141, 3998,
    4036, 4137
\l__enumext_anspic_below_int . 141, 4125, 4126,
    4129
\l__enumext_anspic_body_box ... 141, 4049, 4052
\__enumext_anspic_body_dim:n .. 108, 4041, 4041,
    4087
\l__enumext_anspic_body_htdp_dim .. 108, 141,
    4050, 4098
\__enumext_anspic_label:nn 108, 4060, 4060, 4090,
    4104
\l__enumext_anspic_label_box .. 141, 3950, 3953
\l__enumext_anspic_label_htdp_dim . 106, 141,
    3951, 3957, 4097
\__enumext_anspic_label_pos:nnn .. 109, 4084,
    4084, 4112
\l__enumext_anspic_mini_pos_str 106, 141, 3929,
    3932, 4135
\l__enumext_anspic_mini_width_dim 141, 4062,
    4124, 4135
\__enumext_anspic_print:n 109, 110, 3998, 4000,
    4117, 4117, 4121
\__enumext_anspic_row:n .. 110, 4117, 4119, 4122
\__enumext_anspic_start_list_tag: 3868, 3894,
    4111
\__enumext_anspic_stop_list_tag: . 3868, 3910,
    4115
\__enumext_anspic_stop_start_list_tag: 3868,
    3902, 4113
\__enumext_at_begin_document:n .. 36, 201, 201,
    402, 408
\l__enumext_base_line_fix_bool 871, 896, 5075
\__enumext_before_args_exec: . 48, 99, 117, 1021,
    1021, 3613
\__enumext_before_args_exec_v: 1037, 1037, 3717
\__enumext_before_args_exec_vii: . 1053, 1053,
    4518
\__enumext_before_args_exec_viii: 1057, 4791
\__enumext_before_env:nn 82, 205, 209, 2675, 2687,
    2699, 2800
\__enumext_before_keys_exec: .. 48, 1021, 1025,
    3678
\__enumext_before_keys_exec_v: 1037, 1041, 3811
\__enumext_before_keys_exec_vii ..... 1053
\__enumext_before_keys_exec_vii: . 1061, 4472
\__enumext_before_keys_exec_viii: 1065, 4740
\__enumext_before_list: ... 99, 3610, 3610, 3672
\__enumext_before_list_v: ... 3714, 3714, 3806
\__enumext_before_list_vii: ... 117, 4467, 4514,
    4514
\__enumext_before_list_viii: .. 123, 4736, 4788,
    4788
\l__enumext_before_no_starred_key_v_tl 1043
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1063
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1067
\l__enumext_before_starred_key_v_tl ... 1039
\l__enumext_before_starred_key_vii_tl . 1055
\l__enumext_before_starred_key_viii_tl 1059
\__enumext_calc_hspace:NNNNNNN 95, 3390, 3390,
    3421, 3426, 3469
\__enumext_check_ans_active: . 68, 99, 117, 2042,
    2042, 3614, 4517
\g__enumext_check_ans_item_tl ..... 87
\g__enumext_check_ans_key_bool 69, 70, 152, 377,
    2101, 2107, 2914
\l__enumext_check_ans_key_bool 69, 2027, 2032,
    2098, 2104

```

```

\__enumext_check_ans_key_hook: . . 69, 100, 117,
    2095, 2095, 3649, 4525
\__enumext_check_ans_level: . 68, 69, 2042, 2048,
    2052
\__enumext_check_ans_log: 70, 85, 2141, 2141, 2918
\__enumext_check_ans_log_msg_greater: 2141,
    2147, 2160
\__enumext_check_ans_log_msg_less: 2141, 2145,
    2150
\__enumext_check_ans_log_msg_same_ok: 2141,
    2146, 2155
\__enumext_check_ans_msg_greater: 2117, 2123,
    2136
\__enumext_check_ans_msg_less: 2117, 2121, 2126
\__enumext_check_ans_msg_same_ok: 2117, 2122,
    2131
\__enumext_check_ans_show: . . 70, 85, 2117, 2117,
    2916
\l__enumext_check_answers_bool 67, 68, 79, 89, 90,
    152, 2004, 2031, 2046, 2333, 2357, 2364, 2388, 2616,
    2813, 3039, 3128, 3162, 4632
\__enumext_check_starred_cmd:n 34, 71, 87, 122,
    2165, 2165, 3817, 4011, 4754
\g__enumext_check_starred_cmd_int . . 94, 152,
    2168, 2174, 2179, 3335, 4069, 4875
\l__enumext_check_start_line_env_tl . 34, 152,
    340, 348, 356, 2171, 2177, 2180
\l__enumext_columns_sep_v_dim 3735, 3737, 3745
\l__enumext_columns_sep_vii_dim . . 4148, 4150,
    4159, 4171, 4247, 4709
\l__enumext_columns_sep_viii_dim . 4179, 4181,
    4190, 4202, 4296, 4975
\l__enumext_columns_v_int 1400, 1418, 1586, 3733,
    3741, 3753, 3758
\l__enumext_columns_vii_int . . 4153, 4156, 4160,
    4169, 4211, 4215, 4218, 4224, 4230, 4234, 4703, 4717
\l__enumext_columns_viii_int . 4184, 4187, 4191,
    4200, 4260, 4264, 4267, 4273, 4279, 4283, 4969, 4984
\l__enumext_counter_i_tl . . . . . 45, 494
\l__enumext_counter_ii_tl . . . . . 45, 495
\l__enumext_counter_iii_tl . . . . . 45, 496
\l__enumext_counter_iv_tl . . . . . 45, 497
\c__enumext_counter_style_tl . . . . 32, 50, 225
\g__enumext_counter_styles_tl. 28, 39, 67, 505,
    523
\l__enumext_counter_v_tl . . . . . 45, 498, 752
\l__enumext_counter_vi_tl . . . . . 45, 499
\l__enumext_counter_vii_tl . . . . 45, 500, 682
\l__enumext_counter_viii_tl . . . . 45, 501, 699
\l__enumext_current_widest_dim 28, 67, 529, 615,
    662, 733, 737
\__enumext_def_meta_key:nnn . . . 131, 5196, 5224,
    5230, 5244
\__enumext_default_item:n . . . 3124, 3124, 3188
\__enumext_define_counters:Nn 28, 485, 485, 494,
    495, 496, 497, 498, 499, 500, 501
\__enumext_endminipage: . 36, 402, 411, 425, 4362,
    4690, 4956
\g__enumext_envir_name_tl 34, 34, 311, 326, 385,
    1974, 1979, 1989, 2129, 2134, 2139, 2153, 2158, 2163
\l__enumext_envir_name_tl . 33, 34, 34, 280, 290,
    339, 347, 355, 5592, 5595, 5602, 5605, 5612, 5615,
    5622, 5625, 5631, 5635, 5641, 5645, 5702, 5706
\__enumext_execute_after_env: 35, 66, 70, 81, 85,
    2904, 2904, 3688, 4730
\__enumext_fake_item_indent: . . 942, 942, 3453
\l__enumext_fake_item_indent_v_dim 961, 966
\l__enumext_fake_item_indent_v_tl 963, 3317,
    3321, 3329
\l__enumext_fake_item_indent_vii_dim 974, 979
\l__enumext_fake_item_indent_vii_tl 976, 4684
\l__enumext_fake_item_indent_viii_dim . 987,
    992, 4946
\l__enumext_fake_item_indent_viii_tl . . 989,
    4944, 4949
\l__enumext_fake_item_indent_X_tl . . . . 100
\__enumext_fake_item_vii: . . . . 942, 971, 3486
\__enumext_fake_item_viii: . . . . 942, 984, 3491
\__enumext_fake_make_label_vii:n . 119, 4617,
    4617, 4680
\__enumext_fake_make_label_viii:n 4896, 4896,
    4938
\__enumext_filter_first_level:n . . 129, 5093,
    5093, 5127, 5138
\__enumext_filter_first_level_key:n 129, 5093,
    5098, 5102
\__enumext_filter_first_level_pair:nn . 130,
    5093, 5099, 5111
\__enumext_filter_save_key:n . . 73, 2249, 2257,
    2280, 2286, 2288, 2288, 5026, 5030, 5034, 5038, 5042,
    5046
\__enumext_filter_save_key_key:n . . 73, 2288,
    2293, 2297
\__enumext_filter_save_key_pair:nn 74, 2288,
    2294, 2305
\__enumext_filter_series:n 62, 1731, 1731, 1769,
    1781, 1786
\__enumext_filter_series_key:n 62, 1731, 1736,
    1740
\__enumext_filter_series_pair:nn . . 62, 1731,
    1737, 1748
\__enumext_first_item_tmp_vii: 116, 118, 4477,
    4550, 4550
\__enumext_first_item_tmp_viii: 122, 123, 4746,
    4800, 4800
\g__enumext_footnote_arg_seq . 170, 4436, 4449,
    4459
\g__enumext_footnote_int . 170, 4443, 4446, 4448,
    4450
\g__enumext_footnote_int_seq . 170, 4437, 4450,
    4455, 4458
\__enumext_footnotes_key_bool . . . . . 37
\l__enumext_footnotes_key_bool 31, 37, 120, 160,
    440, 447, 456, 4674, 4697, 4932, 4963
\__enumext_footnotetext:nn . . . 4430, 4430, 4460
\__enumext_foreach_add_body:n . 133, 5245, 5305,
    5308
\l__enumext_foreach_after_tl . . . . 5249, 5317
\l__enumext_foreach_before_tl . . . 5247, 5312
\g__enumext_foreach_default_keys_tl 132, 126,
    5267, 5288
\__enumext_foreach_keyans:nn . . 133, 5245, 5284,
    5286
\l__enumext_foreach_name_prop_tl . 126, 5290,
    5315
\l__enumext_foreach_print_seq 126, 5300, 5306,
    5310
\l__enumext_foreach_sep_tl . . . . . 5259, 5306
\l__enumext_foreach_start_int . . . 5251, 5302

```

```

\l__enumext_foreach_step_int . . . . . 5255, 5303
\l__enumext_foreach_stop_int . 5253, 5295, 5297,
    5304
\__enumext_foreach_wrapper:n . . . . . 5257, 5313
\__enumext_getkeyans:nn . . 127, 5009, 5013, 5013
\__enumext_getkeyans_aux:n 127, 4997, 5000, 5000
\l__enumext_hyperref_bool 31, 37, 160, 436, 459,
    476, 2535, 3027, 4626
\__enumext_hypertarget:nn 37, 431, 461, 465, 481
\__enumext_if_is_int:n . . . . . 217
\__enumext_if_is_int:nTF . . . . . 217, 771, 785
\__enumext_internal_mini_page: 36, 97, 116, 413,
    413, 3500, 4489
\__enumext_is_not_nested: 28, 33, 97, 116, 274, 274,
    3501, 4490
\__enumext_is_on_first_level: . 28, 34, 97, 116,
    274, 300, 3507, 4502
\g__enumext_item_anskey_int 79, 88, 152, 372, 399,
    400, 2114, 2484, 3041
\__enumext_item_answer_diff: 70, 85, 2110, 2110,
    2911
\g__enumext_item_answer_diff_int 70, 152, 373,
    2112, 2119, 2143
\l__enumext_item_column_pos_vii_int 118, 4218,
    4224, 4230, 4234, 4241, 4557, 4703, 4706
\l__enumext_item_column_pos_viii_int . 123,
    4267, 4273, 4279, 4283, 4290, 4807, 4969, 4972
\l__enumext_item_column_pos_X_int . . . . . 173
\g__enumext_item_count_all_vii_int 118, 4242,
    4558, 4717, 4725
\g__enumext_item_count_all_viii_int 123, 4291,
    4808, 4983, 4992
\g__enumext_item_count_all_X_int . . . . . 173
\g__enumext_item_number_bool . . . . . 152
\l__enumext_item_number_bool 69, 158, 2064, 2069,
    2073, 2077, 2090, 2659, 2713, 3131, 3165, 4635
\g__enumext_item_number_int . . 69, 152, 371, 398,
    400, 2063, 2068, 2072, 2076, 2089, 2114, 3130, 3164,
    4634
\__enumext_item_peek_args_vii: 118, 4559, 4561,
    4561
\__enumext_item_peek_args_viii: 123, 124, 4809,
    4811, 4811
\__enumext_item_star_exec: 90, 3143, 3170, 3207,
    3226
\l__enumext_item_starred_vii_bool 4576, 4590,
    4639
\l__enumext_item_starred_viii_bool 4826, 4840,
    4906, 4942
\l__enumext_item_starred_X_bool . . . . . 173
\__enumext_item_std:w 36, 89, 90, 93, 402, 406, 3134,
    3140, 3168, 3317, 3321, 3329
\g__enumext_item_symbol_aux_tl . 90, 130, 3148,
    3151, 3176, 3212, 3230
\g__enumext_item_symbol_aux_vii_tl 4598, 4641,
    4644, 4648, 4650
\g__enumext_item_symbol_aux_X_tl . . . . . 173
\l__enumext_item_symbol_sep_vii_dim . . 4606,
    4613, 4647, 4649
\l__enumext_item_symbol_vii_tl . . . . . 4644
\l__enumext_item_text_vii_box . . . 4666, 4694
\l__enumext_item_text_viii_box . . 4924, 4960
\l__enumext_item_text_X_box . . . . . 173
\l__enumext_item_width_vii_dim . . 4157, 4166,
    4245, 4253, 4254
\l__enumext_item_width_viii_dim . . 4188, 4197,
    4294, 4302, 4303
\l__enumext_item_width_X_dim . . . . . 173
\l__enumext_itemindent_X_dim . . . . . 71
\l__enumext_itemsep_i_skip . . 1261, 1268, 1271,
    1273, 1280, 1284, 1287, 1289, 1429, 1436, 1438, 1439,
    1444, 1448, 1450, 1451
\l__enumext_itemsep_ii_skip . . 1301, 1308, 1311,
    1313, 1320, 1324, 1327, 1329
\l__enumext_itemsep_iii_skip . 1340, 1347, 1350,
    1352, 1359, 1363, 1366, 1368
\l__enumext_itemsep_vii_skip . . . . . 4723
\l__enumext_itemsep_viii_skip . . . . . 4990
\l__enumext_joined_item_aux_vii_int . . 4239,
    4240, 4241, 4242, 4248
\l__enumext_joined_item_aux_viii_int . 4288,
    4289, 4290, 4291, 4297
\l__enumext_joined_item_aux_X_int . . . . 173
\__enumext_joined_item_vii:w . . 118, 4564, 4565,
    4567, 4567
\l__enumext_joined_item_vii_int . . 4210, 4211,
    4214, 4216, 4222, 4227, 4232, 4237, 4239, 4245
\__enumext_joined_item_viii:w 124, 4814, 4815,
    4817, 4817
\l__enumext_joined_item_viii_int . 4259, 4260,
    4263, 4265, 4271, 4276, 4281, 4286, 4288, 4294
\l__enumext_joined_item_X_int . . . . . 173
\l__enumext_joined_width_vii_dim . 4243, 4250,
    4253, 4668, 4682
\l__enumext_joined_width_viii_dim 4292, 4299,
    4302, 4926, 4940
\l__enumext_joined_width_X_dim . . . . . 173
\__enumext_keyans_addto_prop:n 86, 2924, 2924,
    3332, 4066
\__enumext_keyans_addto_seq:n . 87, 3000, 3000,
    3334, 4068
\__enumext_keyans_addto_seq_link: 3000, 3021,
    3023, 4874
\__enumext_keyans_default_item:n . . 93, 3312,
    3312, 3349
\l__enumext_keyans_env_bool 34, 3532, 3545, 3697,
    3787
\__enumext_keyans_fake_item_indent: 942, 958,
    3443
\l__enumext_keyans_level_h_int . . 122, 28, 692,
    719, 2643, 2705, 2978, 4496, 4763, 4764
\l__enumext_keyans_level_int . . 28, 1546, 2639,
    2701, 2973, 3696, 3701, 4032
\__enumext_keyans_make_label: 39, 94, 3353, 3353,
    3441
\__enumext_keyans_make_label_box: 3353, 3357,
    3375
\__enumext_keyans_make_label_std: 3353, 3359,
    3361
\__enumext_keyans_mini_right_cmd:n 59, 1548,
    1581, 1581
\__enumext_keyans_mini_set_vskip: . . . . . 55
\__enumext_keyans_minipage_add_space: 1380,
    1406, 3726
\__enumext_keyans_minipage_set_skip: . 1380,
    1380, 1408
\__enumext_keyans_multi_addvspace: 1179, 1190,
    3750

```


__enumext_keyans_multi_set_vskip: 52, [1179](#),
 [1179](#), [1192](#)
 __enumext_keyans_multicols_start: [3714](#), [3729](#),
 [3731](#)
 __enumext_keyans_multicols_stop: 1585, [3714](#),
 [3756](#), [3785](#)
 __enumext_keyans_name_and_start: 28, [34](#), [122](#),
 [333](#), [333](#), [3698](#), [3925](#), [4768](#)
 __enumext_keyans_parse_keys:n [3710](#), [3710](#), [3805](#)
 __enumext_keyans_pic_arg_two: 106, [3942](#), [3942](#),
 [3972](#)
 \l__enumext_keyans_pic_level_int .. [28](#), [1530](#),
 [2647](#), [2709](#), [2927](#), [2968](#), [3003](#), [3091](#), [3920](#), [3921](#)
 \g__enumext_keyans_pic_parsep_skip [141](#), [3959](#),
 [4018](#)
 __enumext_keyans_pic_safe_exec:n 106, [3918](#),
 [3918](#), [3971](#)
 __enumext_keyans_pic_skip_abs:N . 106, [3935](#),
 [3935](#), [3946](#)
 \l__enumext_keyans_pic_star_bool .. 106, [141](#),
 [3928](#), [3947](#), [4013](#), [4043](#), [4088](#)
 __enumext_keyans_pre_itemsep_skip: .. [1380](#),
 [1399](#), [1426](#)
 __enumext_keyans_redefine_item: .. [94](#), [3337](#),
 [3337](#), [3440](#)
 __enumext_keyans_ref: [43](#), [744](#), [762](#), [3442](#)
 __enumext_keyans_ref:n [43](#), [741](#), [744](#), [744](#)
 __enumext_keyans_safe_exec: . [3690](#), [3690](#), [3804](#)
 __enumext_keyans_set_item_width: 103, [3791](#),
 [3791](#), [3813](#)
 __enumext_keyans_show_ans: .. [3044](#), [3052](#), [3071](#)
 __enumext_keyans_show_item_opt: .. [93](#), [3044](#),
 [3059](#), [3330](#), [4081](#), [4945](#)
 __enumext_keyans_show_left:n . [93](#), [3044](#), [3044](#),
 [3327](#), [4075](#)
 __enumext_keyans_show_pos: .. [3044](#), [3056](#), [3084](#)
 __enumext_keyans_starred_item:n .. [93](#), [3324](#),
 [3324](#), [3345](#)
 __enumext_keyans_store_ref: .. [86](#), [2947](#), [2947](#),
 [3333](#), [4067](#), [4872](#)
 __enumext_keyans_store_ref_aux_i: 86, [2947](#),
 [2959](#), [2962](#)
 __enumext_keyans_store_ref_aux_ii: [87](#), [2947](#),
 [2988](#), [2990](#)
 __enumext_keyans_unknown_keys:n . [3250](#), [3254](#),
 [3258](#)
 __enumext_keyans_unknown_keys:nn [3250](#), [3260](#),
 [3262](#)
 __enumext_keyans_wrapper_opt:n .. [2192](#), [3067](#)
 \l__enumext_label_copy_i_tl .. [2450](#), [2966](#), [2971](#),
 [2976](#), [2981](#)
 \l__enumext_label_copy_v_tl [2976](#)
 \l__enumext_label_copy_vi_tl [2971](#)
 \l__enumext_label_copy_vii_tl [2426](#), [2437](#), [2466](#),
 [2966](#)
 \l__enumext_label_copy_viii_tl [2981](#)
 \l__enumext_label_copy_X_tl [162](#)
 \l__enumext_label_fill_left_v_tl [3365](#)
 \l__enumext_label_fill_left_X_tl [100](#)
 \l__enumext_label_fill_right_v_tl [3372](#)
 \l__enumext_label_fill_right_X_tl [100](#)
 \l__enumext_label_font_style_v_tl [3366](#), [3381](#),
 [4079](#)
 \l__enumext_label_font_style_vii_tl ... [4654](#)
 \l__enumext_label_font_style_viii_tl .. [4912](#)
 \l__enumext_label_i_tl [607](#)
 \l__enumext_label_ii_tl [607](#)
 \l__enumext_label_iii_tl [607](#)
 \l__enumext_label_iv_tl [607](#)
 __enumext_label_style:Nnn 28, [39](#), [518](#), [518](#), [533](#),
 [612](#), [659](#), [730](#), [734](#)
 \l__enumext_label_v_tl 87, [727](#), [2932](#), [3008](#), [3078](#),
 [3118](#), [3326](#), [3331](#), [3808](#), [3950](#), [4074](#), [4076](#)
 \l__enumext_label_vi_tl 87, [727](#), [2929](#), [3005](#), [4074](#),
 [4076](#), [4080](#)
 \l__enumext_label_vii_tl . [654](#), [4585](#), [4608](#), [4615](#)
 \l__enumext_label_viii_tl [654](#), [4835](#), [4866](#), [4870](#)
 \l__enumext_label_width_by_box .. [67](#), [514](#), [515](#)
 __enumext_label_width_by_box:Nn [39](#), [512](#), [512](#),
 [517](#), [529](#), [795](#)
 \l__enumext_labelsep_i_dim ... [3076](#), [3081](#), [3089](#),
 [3121](#), [4878](#), [4893](#)
 \l__enumext_labelsep_v_dim [3740](#)
 \l__enumext_labelsep_vii_dim . [2551](#), [3076](#), [3089](#),
 [4152](#), [4162](#), [4246](#), [4552](#), [4606](#), [4661](#), [4670](#)
 \l__enumext_labelsep_viii_dim [4183](#), [4193](#), [4295](#),
 [4802](#), [4919](#), [4928](#), [4946](#)
 \l__enumext_labelwidth_i_dim . [3075](#), [3081](#), [3088](#),
 [3121](#), [4878](#), [4893](#)
 \l__enumext_labelwidth_v_dim [3379](#), [3740](#)
 \l__enumext_labelwidth_vii_dim ... [2551](#), [3075](#),
 [3088](#), [4152](#), [4161](#), [4246](#), [4552](#), [4652](#), [4669](#)
 \l__enumext_labelwidth_viii_dim .. [4183](#), [4192](#),
 [4295](#), [4802](#), [4910](#), [4927](#)
 \l__enumext_leftmargin_tmp_v_bool . 106, [3944](#)
 \l__enumext_leftmargin_tmp_X_bool [71](#)
 \l__enumext_leftmargin_tmp_X_dim [71](#)
 \l__enumext_leftmargin_X_dim [71](#)
 __enumext_level: 213, [213](#), [636](#), [639](#), [640](#), [649](#), [651](#),
 [945](#), [949](#), [953](#), [1023](#), [1027](#), [1031](#), [1035](#), [1118](#), [1120](#),
 [1122](#), [1124](#), [1166](#), [1168](#), [1170](#), [1172](#), [1177](#), [1212](#), [1218](#),
 [1223](#), [1225](#), [1228](#), [1231](#), [1244](#), [1247](#), [1555](#), [1559](#), [1565](#),
 [1628](#), [1630](#), [1632](#), [1635](#), [1642](#), [1644](#), [1646](#), [1649](#), [2244](#),
 [2246](#), [2248](#), [2276](#), [2277](#), [2279](#), [2335](#), [2343](#), [2347](#), [2351](#),
 [2555](#), [2556](#), [3133](#), [3134](#), [3138](#), [3139](#), [3140](#), [3148](#), [3156](#),
 [3157](#), [3160](#), [3167](#), [3168](#), [3172](#), [3175](#), [3177](#), [3203](#), [3204](#),
 [3205](#), [3208](#), [3211](#), [3220](#), [3221](#), [3223](#), [3224](#), [3227](#), [3538](#),
 [3551](#), [3558](#), [3566](#), [3569](#), [3571](#), [3573](#), [3574](#), [3575](#), [3576](#),
 [3579](#), [3584](#), [3590](#), [3596](#), [3603](#), [3616](#), [3618](#), [3621](#), [3622](#),
 [3624](#), [3628](#), [3634](#), [3659](#), [3664](#), [3675](#), [3677](#)
 \l__enumext_level_h_int 116, [28](#), [283](#), [306](#), [320](#), [675](#),
 [712](#), [1537](#), [2060](#), [2080](#), [2445](#), [2679](#), [2691](#), [3546](#), [4491](#),
 [4492](#)
 \l__enumext_level_int . 97, [28](#), [215](#), [293](#), [305](#), [321](#),
 [415](#), [1130](#), [1257](#), [1536](#), [2054](#), [2086](#), [2422](#), [2432](#), [2438](#),
 [2444](#), [2451](#), [2460](#), [2465](#), [2678](#), [2690](#), [2906](#), [3457](#), [3502](#),
 [3503](#), [3514](#), [3522](#), [3536](#), [3549](#), [3580](#), [3705](#), [4028](#), [4534](#),
 [4544](#), [4776](#), [5632](#), [5636](#), [5642](#), [5646](#)
 __enumext_list_arg_two_i: [3422](#)
 __enumext_list_arg_two_ii: [3422](#)
 __enumext_list_arg_two_iii: [3422](#)
 __enumext_list_arg_two_iv: [3422](#)
 __enumext_list_arg_two_v: . [94](#), [3422](#), [3810](#), [3945](#)
 __enumext_list_arg_two_vii: [3463](#), [4471](#)
 __enumext_list_arg_two_viii: [3463](#), [4739](#)
 \l__enumext_listoffset_v_dim . [3742](#), [3796](#), [3799](#)
 \l__enumext_listparindent_vii_dim [4685](#)
 \l__enumext_listparindent_viii_dim ... [4951](#)
 __enumext_log_answer_vars: . 35, [387](#), [395](#), [2913](#)

```

\__enumext_log_global_vars: . 35, 387, 387, 2912
\__enumext_make_label: . 39, 91, 3191, 3191, 3451
\__enumext_make_label_box: . . . 3191, 3195, 3215
\__enumext_make_label_std: . . . 3191, 3197, 3199
\l__enumext_mark_answer_sym_tl 75, 2198, 2401,
    2568, 3093, 3106, 4882
\l__enumext_mark_position_str 130, 2202, 2203,
    2229, 2230, 2399
\l__enumext_mark_ref_sym_tl . . 2215, 2540, 3035
\l__enumext_meta_path_tl . 126, 5220, 5221, 5223,
    5224
\c__enumext_meta_paths_prop . . . . . 131, 5196
\__enumext_mini_addvspace_vii: 57, 1516, 1516,
    4320
\__enumext_mini_addvspace_viii: 57, 1516, 1522,
    4385
__enumext_mini_env* . . . . . 413
\__enumext_mini_page 1565, 1592, 3628, 3727, 4322,
    4387, 4408
\__enumext_mini_right_cmd:n . 58, 59, 1550, 1552,
    1552
\__enumext_mini_set_vskip_vii: 56, 1459, 1459,
    1518
\__enumext_mini_set_vskip_viii: 56, 1459, 1481,
    1524
\__enumext_minipage:w 36, 402, 410, 419, 4345, 4682,
    4940
\l__enumext_minipage_active_v_bool 3724, 3747,
    3772
\g__enumext_minipage_active_vii_bool . . 113,
    4334, 4343, 4365
\l__enumext_minipage_active_vii_bool . 4316,
    4327
\g__enumext_minipage_active_viii_bool 4398,
    4406, 4425
\l__enumext_minipage_active_viii_bool 4381,
    4392
\g__enumext_minipage_active_X_bool . . . 173
\l__enumext_minipage_active_X_bool . . . . 87
\__enumext_minipage_add_space: . 53, 100, 1208,
    1234, 3626
\g__enumext_minipage_after_skip 87, 1463, 1475,
    4363, 4423
\l__enumext_minipage_after_skip . . 52, 100, 87,
    1221, 1261, 1263, 1268, 1271, 1275, 1280, 1284, 1287,
    1291, 1303, 1308, 1311, 1315, 1320, 1324, 1327, 1331,
    1342, 1347, 1350, 1354, 1359, 1363, 1366, 1370, 1382,
    1396, 1429, 1431, 1436, 1438, 1440, 1444, 1448, 1450,
    1452, 1483, 1496, 1510, 1561, 1588, 3782
\g__enumext_minipage_center_vii_bool . 4349,
    4366
\g__enumext_minipage_center_viii_bool 4410,
    4426
\g__enumext_minipage_center_X_bool . . . 173
\l__enumext_minipage_hsep_v_dim . . . . . 3722
\l__enumext_minipage_hsep_vii_dim . . . . 4314
\l__enumext_minipage_hsep_viii_dim . . . 4379
\l__enumext_minipage_left_skip 87, 1383, 1461,
    1466, 1470, 1484, 1488, 1502, 1520, 1526
\l__enumext_minipage_left_v_dim . . 3720, 3727
\l__enumext_minipage_left_vii_dim 4310, 4322
\l__enumext_minipage_left_viii_dim 4375, 4387
\l__enumext_minipage_left_X_dim . . . . . 87
\g__enumext_minipage_right_skip 87, 1462, 1467,
    1471, 4348, 4409
\l__enumext_minipage_right_skip . 52, 87, 1210,
    1216, 1221, 1223, 1225, 1384, 1385, 1391, 1396, 1397,
    1398, 1403, 1485, 1492, 1506, 1567, 1594
\l__enumext_minipage_right_v_dim . 1583, 1592,
    3718, 3722
\g__enumext_minipage_right_vii_dim 113, 4318,
    4345, 4368
\l__enumext_minipage_right_vii_dim 113, 4308,
    4313, 4319
\g__enumext_minipage_right_viii_dim . . 4383,
    4408, 4428
\l__enumext_minipage_right_viii_dim . . 4373,
    4378, 4384
\g__enumext_minipage_right_X_dim . . . . . 173
\g__enumext_minipage_right_X_skip . . . . 173
\__enumext_minipage_set_skip: . 52, 1208, 1208,
    1236
\g__enumext_minipage_stat_int . . 100, 87, 1572,
    1599, 3625, 3636, 3641, 3725, 3774, 3779
\l__enumext_minipage_temp_skip 87, 1282, 1292,
    1295, 1322, 1332, 1335, 1361, 1371, 1374, 1446, 1453,
    1455
\l__enumext_miniright_code_vii_box 4356, 4360
\g__enumext_miniright_code_vii_tl 114, 4351,
    4358, 4367
\l__enumext_miniright_code_viii_box . . 4417,
    4421
\g__enumext_miniright_code_viii_tl 4412, 4419,
    4427
\l__enumext_miniright_code_X_box . . . . . 173
\__enumext_multi_addvspace: . 51, 99, 1161, 1161,
    3587
\__enumext_multi_set_vskip: 50, 1116, 1116, 1163
\l__enumext_multicols_above_ii_skip . . . 1135
\l__enumext_multicols_above_iii_skip . . 1144
\l__enumext_multicols_above_iv_skip . . . 1153
\l__enumext_multicols_above_v_skip 1181, 1195,
    1206, 1397
\l__enumext_multicols_above_X_skip . . . . 79
\l__enumext_multicols_below_ii_skip . . 1264,
    1273, 1277, 1289, 1294
\l__enumext_multicols_below_iii_skip . 1304,
    1313, 1317, 1329, 1334
\l__enumext_multicols_below_iv_skip . . 1343,
    1352, 1356, 1368, 1373
\l__enumext_multicols_below_v_skip 1185, 1199,
    1398, 1432, 1439, 1441, 1451, 1454, 3764
\l__enumext_multicols_below_X_skip . . . . 79
\g__enumext_multicols_right_X_skip . . . . 79
\__enumext_multicols_start: 99, 100, 3563, 3563,
    3630
\__enumext_multicols_stop: 99, 1557, 3593, 3593,
    3646
\__enumext_nested_base_line_fix: . 45, 98, 869,
    876, 3518
\__enumext_newlabel:nn 31, 37, 77, 469, 469, 2476,
    2994
\l__enumext_newlabel_arg_one_tl 31, 37, 77, 86,
    162, 2469, 2477, 2539, 2983, 2995, 3033
\l__enumext_newlabel_arg_two_tl 31, 37, 76, 162,
    2425, 2435, 2448, 2463, 2478, 2970, 2975, 2980, 2996
\__enumext_parse_foreach_keys:n . . 5245, 5261,
    5278

```

```

\__enumext_parse_foreach_keys:nn . 5245, 5268,
    5280
\__enumext_parse_keys:n 45, 62, 3509, 3509, 3671
\__enumext_parse_keys_vii:n . 45, 62, 4466, 4504,
    4504
\__enumext_parse_keys_viii:n . 4735, 4781, 4781
\__enumext_parse_save_key:n 73, 2269, 2274, 2274
\__enumext_parse_save_key_vii:n 73, 2264, 2274,
    2282
\__enumext_parse_series:n 62, 98, 116, 1757, 1757,
    3517, 4510
\__enumext_parse_store_keys:n . . . . . 98
\l__enumext_parsep_i_skip . . . . . 1133, 1137
\l__enumext_parsep_ii_skip . . . . . 1142, 1146
\l__enumext_parsep_iii_skip . . . . . 1151, 1155
\l__enumext_parsep_vii_skip . . . . . 4686
\l__enumext_parsep_viii_skip . . . . . 4952
\l__enumext_partopsep_v_skip . 1197, 1201, 1393,
    1416
\l__enumext_partopsep_viii_skip . . . . . 1494
\__enumext_phantomsection: 37, 431, 462, 466, 482
\__enumext_pre_itemsep_skip: . . 53, 1226, 1255,
    1255
\__enumext_print_footnote: . . . 4430, 4453, 4699,
    4965
\__enumext_print_keyans_box:NN 75, 2393, 2393,
    2406, 2550, 2554, 3080, 3120, 4878, 4893
\l__enumext_print_keyans_i_tl . . . . 5031, 5053
\l__enumext_print_keyans_ii_tl . . . 5035, 5054
\l__enumext_print_keyans_iii_tl . . 5039, 5055
\l__enumext_print_keyans_iv_tl . . . 5043, 5056
\l__enumext_print_keyans_star_bool 875, 881,
    887, 5076
\l__enumext_print_keyans_starred_tl 128, 129,
    130, 5027, 5074
\l__enumext_print_keyans_vii_tl 128, 5047, 5057
\l__enumext_print_keyans_X_tl . . . . . 130
\__enumext_printkeyans:nnn 128, 5058, 5061, 5061
\__enumext_redefine_item: . 90, 3180, 3180, 3450
\l__enumext_ref_key_arg_tl 41, 50, 228, 629, 630,
    643, 674, 677, 688, 694, 705, 746, 747, 758
\l__enumext_ref_the_count_tl . 41, 50, 636, 639,
    642, 682, 684, 687, 699, 701, 704, 752, 754, 757
\__enumext_regex_counter_style: . . 32, 41, 223,
    223, 637, 683, 700, 753
\__enumext_register_counter_style:Nn . . 502,
    502, 507, 508, 509, 510, 511
\__enumext_remove_extra_parsep_vii: . . 4484,
    4712, 4712
\__enumext_remove_extra_parsep_viii: . 4753,
    4978, 4978
\__enumext_renew_footnote: . . . 4430, 4434, 4676,
    4934
\l__enumext_renew_the_count_v_tl 755, 764, 766
\l__enumext_renew_the_count_vii_tl 685, 714,
    716
\l__enumext_renew_the_count_viii_tl 702, 721,
    723
\l__enumext_renew_the_count_X_tl . . . . . 50
\__enumext_rescan_anskey_env:n . . 83, 84, 2788,
    2883, 2891
\__enumext_reset_global_bool: . . 363, 366, 375
\__enumext_reset_global_int: . . . 363, 365, 369
\__enumext_reset_global_tl: . . . . 363, 367, 381
\__enumext_reset_global_vars: . 35, 85, 363, 363,
    2921
\l__enumext_resume_active_bool 62, 65, 61, 1761,
    1881
\__enumext_resume_counter: . . 64, 65, 1879, 1885,
    1892
\__enumext_resume_counter:n . 62, 65, 1850, 1855,
    1879, 1879, 1949, 1957
\__enumext_resume_counter_save_ans: 65, 1879,
    1890, 1922
\__enumext_resume_counter_series: . 65, 1879,
    1888, 1905
\g__enumext_resume_int . . . 61, 1802, 1896, 1897
\__enumext_resume_last:n 62, 63, 1757, 1763, 1776
\l__enumext_resume_name_tl 61, 1798, 1806, 1809,
    1825, 1833, 1836, 1882, 1883, 1911, 1918
\__enumext_resume_save_counter: . 63, 100, 117,
    1789, 1789, 3652, 4528
\__enumext_resume_series:n . 64, 1725, 1846, 1846
\__enumext_resume_starred: . 66, 1726, 1943, 1943
\g__enumext_resume_vii_int 61, 1829, 1901, 1902
\l__enumext_rightmargin_vii_dim . 4164, 4168,
    4173
\l__enumext_rightmargin_viii_dim . 4195, 4199,
    4204
\__enumext_safe_exec: . . 36, 97, 3498, 3498, 3670
\__enumext_safe_exec_vii: . 36, 4465, 4487, 4487
\__enumext_safe_exec_viii: 122, 4734, 4757, 4757
\__enumext_second_part: . . 100, 3632, 3632, 3684
\__enumext_second_part_v: . . . 3714, 3770, 3818
\l__enumext_series_name_tl . . . . . 65
\l__enumext_series_str . . 63, 98, 116, 1723, 1759,
    1767, 1768, 1770, 1772, 1793, 1796, 1800, 1820, 1823,
    1827, 3513, 4508
\__enumext_set_error:nn . . . . . 5182, 5192, 5194
\__enumext_set_item_width: 101, 3654, 3654, 3680
\__enumext_set_parse:n . . . . . 5166, 5182, 5182
\l__enumext_setkey_tmpa_int . . . 121, 5159, 5163
\l__enumext_setkey_tmpa_seq . . 121, 5157, 5167,
    5173, 5175, 5177, 5189
\l__enumext_setkey_tmpa_tl . . . . 121, 5165, 5169
\l__enumext_setkey_tmpb_seq . . 121, 5158, 5161,
    5165, 5166
\l__enumext_setkey_tmpb_tl 121, 5184, 5186, 5187
\l__enumext_show_answer_bool . 2209, 2233, 2562,
    3050, 3064, 4071, 4876
\__enumext_show_length:nnn . . 48, 231, 231, 5403,
    5404, 5405, 5406, 5407, 5408, 5409, 5410, 5411, 5412,
    5418, 5419, 5420, 5421, 5422, 5423, 5424, 5425, 5426,
    5427
\l__enumext_show_position_bool . . . 2212, 2236,
    2566, 3054, 3065, 4072, 4880
\g__enumext_standar_bool 33, 97, 34, 282, 285, 304,
    378, 1791, 1856, 1868, 1894, 1907, 1945, 2085, 2099,
    2430, 2443, 2458, 3533
\l__enumext_standar_bool 97, 100, 34, 2431, 3505,
    3651, 4501
\l__enumext_standar_first_bool 34, 97, 34, 309,
    1778, 1925, 1987, 1994
\__enumext_standar_item_vii:w . 118, 4572, 4574,
    4574
\__enumext_standar_item_viii:w 124, 4822, 4824,
    4824
\__enumext_standar_ref: . . . . 41, 627, 647, 3452

```

```

\__enumext_standar_ref:n . . . . 41, 619, 627, 627
\g__enumext_standar_series_tl . 61, 1780, 1781,
    1947, 1950
\__enumext_standar_unknown_keys:n 3290, 3294,
    3298
\__enumext_standar_unknown_keys:nn 3290, 3300,
    3302
\g__enumext_starred_bool 33, 116, 34, 292, 295, 319,
    379, 1818, 1861, 1872, 1899, 1914, 1953, 2059, 2105,
    2421, 2964, 4369
\l__enumext_starred_bool 116, 117, 122, 34, 1542,
    2459, 2494, 2500, 2548, 2837, 2842, 3073, 3086, 3506,
    4500, 4527, 4769, 4773
\__enumext_starred_columns_set_vii: . . 4146,
    4146, 4475
\__enumext_starred_columns_set_viii: . 4146,
    4177, 4744
\l__enumext_starred_first_bool 34, 116, 34, 324,
    880, 886, 1783, 1934, 1987, 1994
\__enumext_starred_item:nn . . . 3143, 3143, 3186
\__enumext_starred_item_exec: . 125, 4868, 4868,
    4908
\__enumext_starred_item_vii:w . 118, 119, 4571,
    4588, 4588
\__enumext_starred_item_vii_aux_i:w . . 4588,
    4593, 4596
\__enumext_starred_item_vii_aux_ii:w . 4588,
    4594, 4599, 4601
\__enumext_starred_item_vii_aux_iii:w 4588,
    4604, 4611
\__enumext_starred_item_viii:w 124, 4821, 4838,
    4838
\__enumext_starred_item_viii_aux_i:w . . 124,
    4838, 4843, 4846
\__enumext_starred_item_viii_aux_ii:w . 124,
    4838, 4844, 4861, 4863
\__enumext_starred_joined_item_vii:n 111, 118,
    4208, 4208, 4569
\__enumext_starred_joined_item_viii:n . 111,
    124, 4208, 4257, 4819
\__enumext_starred_ref: . . . . 42, 672, 710, 3483
\__enumext_starred_ref:n . . . . 42, 666, 672, 672
\g__enumext_starred_series_tl . 61, 1785, 1786,
    1955, 1958
\__enumext_starred_unknown_keys:n 3272, 3274,
    3276
\__enumext_starred_unknown_keys:nn 3272, 3278,
    3280
\__enumext_start_from:NNn 43, 769, 769, 782, 804,
    810
\l__enumext_start_i_int . . . . 1897, 1909, 1928
\__enumext_start_item_tmp_vii: 116, 4478, 4554,
    4554
\__enumext_start_item_tmp_viii: . . 122, 4747,
    4804, 4804
\__enumext_start_item_vii:w 118, 120, 4580, 4585,
    4608, 4615, 4663, 4663
\__enumext_start_item_viii:w . 124, 4830, 4835,
    4866, 4921, 4921
\g__enumext_start_line_tl 34, 34, 312, 327, 384,
    2129, 2134, 2139, 2153, 2158, 2163
\__enumext_start_list:nn . 36, 95, 402, 404, 3674,
    3807, 4469, 4737
\__enumext_start_list_tag:n . . 3820, 3844, 4679,
    4937
\__enumext_start_mini_vii: 117, 4306, 4306, 4519
\__enumext_start_mini_viii: . . 123, 4371, 4371,
    4792
\__enumext_start_save_ans_msg: . . 66, 67, 1971,
    1971, 1996
\__enumext_start_store_level: . 98, 3527, 3527,
    3673
\__enumext_start_store_level_vii: 117, 4468,
    4530, 4530
\l__enumext_start_vii_int . . . 1902, 1916, 1937
\l__enumext_start_X_int . . . . . 100
\__enumext_stop_item_tmp_vii: . . 116, 118, 120,
    4477, 4483, 4556, 4665
\__enumext_stop_item_tmp_viii: 122, 123, 4746,
    4752, 4806, 4923
\__enumext_stop_item_vii: 120, 121, 4665, 4688,
    4688
\__enumext_stop_item_viii: 126, 4923, 4954, 4954
\__enumext_stop_list: 36, 113, 117, 402, 405, 3598,
    3606, 3760, 3767, 4329, 4337, 4394, 4401
\__enumext_stop_list_tag:n . . . 3820, 3860, 4691,
    4957
\__enumext_stop_mini_vii: 113, 117, 4306, 4325,
    4523
\__enumext_stop_mini_viii: 123, 4371, 4390, 4796
\__enumext_stop_save_ans_msg: . 66, 1971, 1976,
    2910
\__enumext_stop_start_list_tag: . . 3820, 3852,
    4681, 4939
\__enumext_stop_store_level: 98, 99, 3556, 3556,
    3599, 3607
\__enumext_stop_store_level_vii: . . 113, 117,
    4330, 4338, 4530, 4540
\l__enumext_store_active_bool 30, 67, 112, 1926,
    1935, 2003, 2635, 3531, 3544, 3692, 3700, 4020, 4024,
    4532, 4542, 4759, 4775
\__enumext_store_active_keys:n 72, 73, 98, 2242,
    2242, 3524
\__enumext_store_active_keys_vii:n 72, 73, 116,
    2242, 2252, 4511
\__enumext_store_addto_prop:n 74, 86, 2317, 2317,
    2325, 2485, 2945, 4871
\__enumext_store_addto_seq:n 74, 87, 2326, 2326,
    2330, 2337, 2351, 2359, 2368, 2382, 2390, 2543, 3038
\l__enumext_store_anskey_arg_tl 30, 77, 78, 112,
    2491, 2496, 2498, 2503, 2510, 2513, 2523, 2528, 2531,
    2537, 2543
\__enumext_store_anskey_code:n 77, 80, 84, 2482,
    2482, 2628, 2881, 2889
\l__enumext_store_anskey_env_tl . . 30, 83, 112,
    2811, 2815, 2821, 2883, 2891
\l__enumext_store_anskey_opt_tl . . 30, 84, 112,
    2812, 2839, 2845, 2852, 2858, 2868, 2878, 2887
\__enumext_store_anskey_safe_outer: . . . . 80
\g__enumext_store_columns_break_bool . 2735,
    2836, 2898
\l__enumext_store_columns_break_bool . 2493,
    2584
\l__enumext_store_current_label_tl 30, 86, 87,
    124, 112, 2926, 2929, 2932, 2938, 2943, 2945, 3002,
    3005, 3008, 3014, 3019, 3029, 3038, 4848, 4853, 4857,
    4870, 4871, 4873
\l__enumext_store_current_label_tmp_tl . 30,
    112, 3326, 3331

```


`\l__enumext_store_current_opt_arg_tl` 30, 124, 112, 3048, 3061, 3067, 4859
`__enumext_store_internal_ref:` .. 76, 77, 2407, 2407, 2488
`\g__enumext_store_item_join_int` .. 2738, 2843, 2847, 2899
`\l__enumext_store_item_join_int` .. 2501, 2505, 2587
`\g__enumext_store_item_star_bool` . 2740, 2850, 2900
`\l__enumext_store_item_star_bool` . 2508, 2589
`\g__enumext_store_item_symbol_sep_dim` 2745, 2865, 2870, 2902
`\l__enumext_store_item_symbol_sep_dim` 2520, 2525, 2594
`\g__enumext_store_item_symbol_tl` . 2743, 2856, 2860, 2901
`\l__enumext_store_item_symbol_tl` . 2511, 2515, 2592
`\l__enumext_store_keyans_item_opt_sep_tl` 2195, 2936, 2940, 3012, 3016, 4851, 4855
`__enumext_store_level_close:` . 74, 2331, 2355, 3560
`__enumext_store_level_close_vii:` . 75, 2362, 2386, 4546
`__enumext_store_level_open:` 74, 98, 2331, 2331, 3539, 3552
`__enumext_store_level_open_vii:` .. 75, 2362, 2362, 4536
`\g__enumext_store_name_tl` 30, 67, 112, 383, 390, 391, 392, 393, 1979, 2005, 2128, 2133, 2138, 2152, 2157, 2162, 2908
`\l__enumext_store_name_tl` 30, 67, 68, 112, 1812, 1815, 1839, 1842, 1930, 1939, 1974, 1983, 1984, 2005, 2006, 2007, 2009, 2010, 2012, 2014, 2015, 2017, 2019, 2020, 2044, 2319, 2321, 2328, 2471, 2472, 2574, 2817, 2985, 2986, 3099, 3112, 4888
`\l__enumext_store_ref_key_bool` 77, 2218, 2486, 2534, 2949, 3026
`\l__enumext_store_save_key_vii_bool` .. 2254, 2284
`\l__enumext_store_save_key_vii_tl` 2256, 2257, 2285, 2286, 2366, 2374, 2378, 2382
`\l__enumext_store_save_key_X_bool` .. 72, 130
`\l__enumext_store_save_key_X_tl` .. 72, 73, 130
`\l__enumext_store_upper_level_X_bool` .. 130
`__enumext_storing_exec:` 67, 82, 1981, 1997, 2001
`__enumext_storing_set:n` 66, 67, 1966, 1981, 1981
`\l__enumext_the_counter_v_tl` 754
`\l__enumext_the_counter_vii_tl` 684
`\l__enumext_the_counter_viii_tl` 701
`\l__enumext_the_counter_X_tl` 50
`__enumext_tmp:n` 45, 49, 54, 60, 71, 78, 79, 86, 94, 99, 100, 111, 133, 140, 165, 169, 173, 193, 1719, 1730, 1962, 1970, 2023, 2041, 2182, 2223, 2224, 2241, 2260, 2273, 2409, 2416, 2417, 2438, 2451, 2454, 2465, 2951, 2958, 3250, 3257, 3290, 3297, 3422, 3462, 3463, 3497
`__enumext_tmp:nn` 534, 555, 556, 590, 591, 606, 799, 824, 898, 920, 921, 941, 997, 1005, 1006, 1020, 1085, 1101, 1102, 1115, 1608, 1624, 3234, 3249
`__enumext_tmp:nnn` 607, 623, 624, 625, 626, 654, 670, 671
`__enumext_tmp:nnnnnn` 825, 850, 853, 856, 858, 860, 863, 866
`__enumext_tmp:w` 5006, 5009
`\l__enumext_tmpa_vii_int` 4156, 4159, 4168, 4199
`\l__enumext_tmpa_viii_int` 4187, 4190
`\l__enumext_tmpa_X_dim` 173
`\l__enumext_tmpa_X_int` 173
`\l__enumext_topsep_v_skip` ... 1183, 1187, 1387
`\l__enumext_topsep_vii_skip` .. 1464, 1473, 1477
`\l__enumext_topsep_viii_skip` . 1486, 1508, 1512
`__enumext_undefine_anskey_env:` . 81, 85, 2668, 2668, 2919
`__enumext_unskip_unkern:` .. 33, 237, 237, 1175, 1204, 1237, 1409, 3601, 3602, 3642, 3762, 3763, 3780
`__enumext_unskip_unkern_new:` 254
`\l__enumext_vspace_a_star_v_bool` 1657
`\l__enumext_vspace_a_star_vii_bool` ... 1679
`\l__enumext_vspace_a_star_viii_bool` ... 1690
`\l__enumext_vspace_a_star_X_bool` 100
`__enumext_vspace_above:` 60, 99, 1625, 1625, 3612
`__enumext_vspace_above_v:` . 60, 1653, 1653, 3716
`\l__enumext_vspace_above_v_skip` .. 1655, 1659, 1661
`__enumext_vspace_above_vii:` 61, 117, 1675, 1675, 4516
`\l__enumext_vspace_above_vii_skip` 1677, 1681, 1683
`__enumext_vspace_above_viii:` . 61, 1675, 1686, 4790
`\l__enumext_vspace_above_viii_skip` 1688, 1692, 1694
`\l__enumext_vspace_b_star_v_bool` 1668
`\l__enumext_vspace_b_star_vii_bool` ... 1701
`\l__enumext_vspace_b_star_viii_bool` ... 1712
`\l__enumext_vspace_b_star_X_bool` 100
`__enumext_vspace_below:` 60, 100, 1639, 1639, 3650
`__enumext_vspace_below_v:` . 60, 1664, 1664, 3789
`\l__enumext_vspace_below_v_skip` .. 1666, 1670, 1672
`__enumext_vspace_below_vii:` 61, 117, 1697, 1697, 4526
`\l__enumext_vspace_below_vii_skip` 1699, 1703, 1705
`__enumext_vspace_below_viii:` . 61, 1697, 1708, 4798
`\l__enumext_vspace_below_viii_skip` 1710, 1714, 1716
`__enumext_widest_from:nnn` .. 44, 783, 783, 798, 817
`\g__enumext_widest_label_tl` 28, 39, 67, 522, 526, 530
`\l__enumext_wrap_label_opt_v_bool` 3320
`\l__enumext_wrap_label_opt_vii_bool` 118, 4579
`\l__enumext_wrap_label_opt_viii_bool` .. 124, 4829
`\l__enumext_wrap_label_opt_X_bool` 100
`\l__enumext_wrap_label_v_bool` 3316, 3320, 3328, 3367, 3382
`\l__enumext_wrap_label_vii_bool` .. 118, 4579, 4583, 4591, 4655
`\l__enumext_wrap_label_viii_bool` . 124, 4829, 4833, 4841, 4913
`\l__enumext_wrap_label_X_bool` 100
`__enumext_wrapper_label_v:n` . 3369, 3384, 4080
`__enumext_wrapper_label_vii:n` 4657
`__enumext_wrapper_label_viii:n` 4915
`\l__enumext_write_aux_file_tl` . 31, 77, 87, 162, 2474, 2480, 2992, 2998

enumext*	6, 4463
enumXi	494
enumXii	494
enumXiii	494
enumXiv	494
enumXv	494
enumXvi	494
enumXvii	494
enumXviii	494
Environments provide by enumext :	
anskey*	30 , 67 , 73 , 76 , 78 , 81–83 , 85 , 98 , 117 , 128 , 134 , 136
enumext*	27 , 28 , 31–33 , 36 , 38 , 41 , 42 , 44–46 , 48–50 , 56 , 57 , 61–64 , 66–69 , 71–77 , 79 , 81 , 84–86 , 91 , 92 , 97 , 98 , 103 , 104 , 110 , 111 , 113–115 , 117 , 120–123 , 125 , 126 , 128 , 129 , 131 , 135 , 138 , 139
enumext	27 , 28 , 32 , 33 , 36 , 38–46 , 48–52 , 55 , 58–64 , 66–69 , 71–74 , 76 , 77 , 79 , 81 , 84–86 , 89–93 , 95 , 98 , 101 , 102 , 106 , 110 , 113 , 116 , 117 , 122 , 128 , 129 , 131 , 134 , 136 , 138
keyans*	27 , 28 , 30–34 , 38 , 41–46 , 48–50 , 56 , 57 , 61 , 67 , 68 , 71 , 72 , 74 , 81 , 86 , 92 , 97 , 103 , 104 , 111 , 112 , 115 , 122 , 135 , 137 , 139
keyanspic	27 , 28 , 30 , 31 , 34 , 38 , 39 , 43 , 67 , 68 , 71 , 74 , 81 , 86–88 , 103–110 , 137
keyans	27 , 28 , 30 , 31 , 33 , 34 , 38 , 39 , 43–46 , 48 , 49 , 52 , 55 , 58–60 , 67 , 68 , 71 , 72 , 74 , 81 , 86–88 , 92–95 , 101–103 , 105 , 106 , 109 , 113 , 123 , 135 , 137
Environments:	
center	110
description	110
enumerate	110
flushleft	110
flushright	110
itemize	110
list	32 , 36 , 79 , 95 , 99 , 101 , 103–107 , 110 , 113
lrbox	120
minipage	32 , 36 , 50 , 52 , 53 , 105 , 106 , 110 , 113 , 114 , 120 , 121 , 126
multicols	50–53 , 58 , 99 , 100
quotation	110
quote	110
scontents	82 , 84
tabbing	110
trivlist	110
verbatim	110
verse	110
exp commands:	
\exp_after:wN	5009
\exp_args:Ne	2880 , 2888 , 3521 , 4997
\exp_args:NV	2600 , 2755 , 3260 , 3278 , 3300 , 5280
\exp_not:N	58 , 525 , 642 , 687 , 704 , 757 , 951 , 965 , 966 , 978 , 979 , 991 , 992 , 2539 , 2571 , 2572 , 3031 , 3096 , 3097 , 3109 , 3110 , 4885 , 4886 , 5006
\exp_not:n	314 , 329 , 342 , 350 , 358 , 581 , 601 , 642 , 643 , 687 , 688 , 704 , 705 , 757 , 758 , 952 , 1746 , 1755 , 2206 , 2303 , 2315 , 2477 , 2505 , 2515 , 2525 , 2539 , 2540 , 2847 , 2860 , 2870 , 2995 , 3033 , 3035 , 5109 , 5119 , 5312 , 5317
F	
\fbbox	2189
\fbboxrule	2189
\fbboxsep	2189
file commands:	
\file_input_stop:	5716
first	1006
font	534

\footnote	115
\footnote	115 , 4438
\footnotemark	4448
\footnotesize	2572 , 3097 , 3110 , 4886
\footnotetext	4432
\foreachkeyans	17 , 132 , 5245
G	
\getkeyans	17 , 127 , 4995
group commands:	
\group_begin:	2570 , 2615 , 2790 , 2877 , 3095 , 3108 , 4884 , 5052
\group_end:	2577 , 2631 , 2894 , 3102 , 3115 , 4891 , 5059
H	
\hbadness	4693 , 4959
hbox commands:	
\hbox_overlap_left:n	3176 , 4648
\hbox_set:Nn	514 , 3950
\hbox_set_end:	4692 , 4958
\hbox_set_to_wd:Nnw	4666 , 4924
\hfill	564 , 569 , 575 , 576 , 1564 , 1591 , 2539 , 3031 , 4333 , 4397
hook commands:	
\hook_gput_code:nnn	9 , 203 , 207 , 211 , 429
\hook_gremove_code:nn	84 , 2806
\hook_gset_rule:nnnn	430
\hook_if_empty:nTF	2804
\hyperlink	78 , 87
\hyperlink	2539 , 3031
\hypertarget	37
\hypertarget	461
I	
\IfDocumentMetadataTF	3193 , 3355 , 3846 , 3854 , 3862 , 3896 , 3904 , 3912 , 3973 , 3983 , 3991 , 4001 , 4006 , 4045 , 4054 , 4131 , 4139 , 4331 , 4395 , 4474 , 4482 , 4624 , 4672 , 4695 , 4743 , 4751 , 4930 , 4961
\IfHyperBoolean	437
\IfPackageLoadedTF	11 , 19 , 433 , 449
\ignorespaces	954 , 967 , 980 , 993 , 4479 , 4661 , 4748 , 4919
\inputlineno	314 , 329 , 342 , 350 , 358
int commands:	
\int_add:Nn	4241 , 4290
\int_case:nn	1130 , 1257 , 2054 , 2080 , 2119 , 2143
\int_case:nnTF	239 , 256
\int_compare:nNnTF	415 , 675 , 692 , 712 , 719 , 1227 , 1246 , 1400 , 1418 , 1530 , 1546 , 1558 , 1586 , 2167 , 2173 , 2639 , 2643 , 2647 , 2655 , 2701 , 2705 , 2709 , 2906 , 2927 , 2968 , 2973 , 2978 , 3003 , 3091 , 3503 , 3514 , 3536 , 3549 , 3565 , 3580 , 3595 , 3636 , 3701 , 3705 , 3733 , 3758 , 3774 , 3921 , 4028 , 4032 , 4211 , 4221 , 4237 , 4260 , 4270 , 4286 , 4492 , 4496 , 4534 , 4544 , 4702 , 4714 , 4764 , 4776 , 4968 , 4980 , 5163 , 5295
\int_compare_p:nNn	283 , 293 , 305 , 306 , 320 , 321 , 1536 , 1537 , 2060 , 2086 , 2422 , 2432 , 2444 , 2445 , 2460 , 2501 , 2678 , 2679 , 2690 , 2691 , 2843 , 3546
\int_decr:N	4240 , 4289
\int_eval:n	400 , 812 , 2321 , 2472 , 2572 , 2986 , 3097 , 3110 , 3437 , 3482 , 4229 , 4278 , 4886
\int_from_alph:n	777 , 791
\int_from_roman:n	779 , 793
\int_gadd:Nn	4242 , 4291
\int_gdecr:N	2063 , 2068 , 2072 , 2076 , 2089
\int_gincr:N	1896 , 1901 , 2484 , 3041 , 3130 , 3164 , 3335 , 3625 , 3725 , 4069 , 4558 , 4634 , 4808 , 4875
\int_gset:Nn	2112 , 4446

`\int_gset_eq:NN` 1795, 1802, 1808, 1814, 1822, 1829, 1835, 1841, 4443

`\int_gzero:N` . 371, 372, 373, 1572, 1599, 2179, 2899, 3641, 3779, 4725, 4992

`\int_if_exist:NTF` 1770, 1806, 1812, 1833, 1839, 2017

`\int_incr:N` 2654, 3502, 3696, 3920, 4491, 4557, 4763, 4807

`\int_mod:nn` 4716, 4982

`\int_new:N` . 28, 29, 30, 31, 32, 33, 61, 62, 87, 104, 123, 143, 144, 155, 156, 157, 159, 170, 176, 177, 178, 179, 180, 1772, 2020

`\int_set:Nn` 773, 777, 779, 1909, 1916, 1928, 1937, 2791, 4125, 4126, 4156, 4187, 4210, 4216, 4232, 4259, 4265, 4281, 4693, 4959, 5159, 5297

`\int_set_eq:NN` 1897, 1902, 4239, 4288

`\int_sign:n` 2114

`\int_step_function:nnN` 2438, 2451, 2465

`\int_step_function:nnnN` 5301

`\int_step_inline:nn` 5211

`\int_step_inline:nnn` 4127

`\int_to_roman:n` 215, 2418, 2455

`\int_use:N` 393, 398, 399, 1228, 1247, 1559, 1911, 1918, 1930, 1939, 3437, 3457, 3482, 3522, 3566, 3575, 3590, 3596, 4214, 4215, 4227, 4263, 4264, 4276, 5632, 5636, 5642, 5646

`\int_zero:N` 4706, 4972

`\item` . 89, 93, 118, 120, 123, 126, 406, 2339, 2345, 2370, 2376, 2498, 3005, 3008, 3182, 3339, 3977, 3979, 4476, 4478, 4745, 4747, 4873

`\item*` 6, 15, 71, 3337

`item-pos*` 3234

`item-sym*` 3234

`\itemindent` 96

`\itemindent` 95

`itemindent` 898

`\itemsep` 3967

`\itemwidth` . 484, 2189, 3656, 3662, 3793, 3799, 4250, 4254, 4299, 4303

K

`keyans` 15, 3802

`keyans*` 15, 4732

`keyanspic` 16, 3969

Keys for `\anskey` provide by [enumext](#):

`break-col` 77, 79, 82–84

`item-join` 77, 79, 82–84

`item-pos*` 78, 79, 82–84

`item-star` 78, 79, 82–84

`item-sym*` 78, 79, 82–84

Keys for `anskey*` provide by [enumext](#):

`break-col` 77, 79, 82–84

`item-join` 77, 79, 82–84

`item-pos*` 78, 79, 82–84

`item-star` 78, 79, 82–84

`item-sym*` 78, 79, 82–84

Keys for environments provide by [enumext](#):

`above*` 29, 59–61, 99, 117

`above` 29, 59–61, 99, 117, 123

`after` 48, 100, 117, 123

`align` 29, 40, 91, 94, 119, 133

`base-fix` 45, 62, 73, 98, 129

`before*` 48, 99, 117, 123

`before` 48

`below*` 29, 59–61, 100, 117

`below` 29, 59–61, 100, 117, 123

`check-ans` 31, 32, 34, 66–71, 74, 85, 88, 100, 101, 117, 121, 135

`columns-sep` 49, 99

`columns` 29, 49, 59, 99

`first` 48, 120

`font` 39, 91, 94, 108, 119

`item-pos*` 90, 92

`item-sym*` 30, 90, 92

`itemindent` 29, 46, 47, 89, 90, 93, 94, 120

`itemsep` 45, 97

`labelsep` 39, 96, 119

`labelwidth` 38–41, 43, 44, 96, 119

`label` 28, 38–40, 43, 44, 106, 110

`lisparindent` 97

`list-indent` 29, 46, 106

`list-offset` 46, 101, 103

`listparindent` 46, 120

`mark-ans` 71, 74, 78

`mark-pos` 71, 72, 133

`mark-ref` 71, 74, 76, 78

`mini-env` 29, 36, 49, 58, 59, 74, 100, 110, 113, 114, 117, 123

`mini-right*` 29, 32, 50, 74, 114, 117

`mini-right` 29, 32, 50, 57, 74, 114, 117

`mini-sep` 29, 49, 74, 100

`no-store` 31, 66–68, 73, 79, 89, 90

`noitemsep` 45

`nosep` 45

`parindent` 97

`parsep` 45, 97, 120

`partopsep` 45

`ref` 28, 32, 40, 41, 43, 135

`resume*` 28, 61, 62, 66, 67, 73, 100, 117, 129

`resume` 28, 35, 61–67, 73, 74, 100, 117, 129, 130

`rightmargin` 46, 111

`save-ans` 30, 35, 62–68, 70, 72–74, 79–82, 85–87, 93, 101, 108, 122, 123, 125, 127, 128, 130, 135

`save-key` 30, 62, 73, 98, 116

`save-pos` 74

`save-ref` 31, 37, 71, 74, 76–78, 86, 87, 94, 125

`save-sep` 71, 74, 86, 124

`series` 28, 61–66, 74, 98, 100, 116, 117, 130

`show-ans` 71, 72, 74, 75, 77, 78, 93, 108, 125

`show-length` 33, 48, 134, 135

`show-pos` 30, 71, 72, 75, 77, 78, 88, 93, 108, 125

`start*` 29, 43, 44, 62

`start` 29, 32, 43, 44, 62

`store-key` 72

`topsep` 45

`widest` 28, 32, 44

`wrap-ans` 38, 71, 74, 75, 78

`wrap-label*` 29, 39, 89, 91, 93, 94, 118, 119, 124

`wrap-label` 29, 39, 89–91, 93, 94, 106, 108, 118, 119, 124

`wrap-opt` 71, 74, 93, 108

keys commands:

`\keys_define:nn` 536, 558, 593, 609, 656, 727, 801, 827, 869, 900, 923, 999, 1008, 1087, 1104, 1610, 1721, 1964, 2025, 2184, 2226, 2262, 2267, 2582, 2733, 2769, 3236, 3252, 3272, 3292, 5023, 5121, 5237, 5245

`\keys_if_exist_p:nn` 5233, 5234

`\l_keys_key_str` 79, 82, 2600, 2755, 3260, 3278, 3300, 5280, 5388

`\keys_precompile:nnN` . . 128, 199, 199, 5025, 5029, 5033, 5037, 5041, 5045, 5263

`\keys_set:nn` . 550, 892, 1110, 1615, 1620, 1858, 1863, 1950, 1958, 2620, 3516, 3521, 3712, 4509, 4785, 5125,

5130, 5131, 5132, 5133, 5136, 5141, 5142, 5143, 5144, 5145, 5146, 5147, 5179, 5289	
\keys_set_known:nn	2887
keyval commands:	
\keyval_parse:NNn	1735, 2292, 5097

L

label	607, 654, 727
Labels provide by enumext:	
\Alph*	38, 39
\Roman*	38, 39
\alph*	38, 39
\arabic*	32, 38, 39
\roman*	38, 39
\labelsep	3961, 3965
labelsep	534
\labelwidth	39
\labelwidth	3961, 3963
labelwidth	534
\lastkern	248, 265
\lastnodetype	239, 256
\lastskip	243, 260
\leftmargin	96
\leftmargin	95, 3961
legacy commands:	
\legacy_if:nTF	4619, 4622, 4898, 4901
\legacy_if_gset_false:n	420, 4346
\legacy_if_set_false:n	4621, 4900
\legacy_if_set_true:n	4584, 4607, 4614, 4628, 4834, 4865
\linewidth	100
\linewidth	3620, 3656, 3722, 3793, 4124, 4159, 4190, 4312, 4377
\list	404
list-indent	898
list-offset	898
\listparindent	3964
listparindent	898

M

\makebox	110
\makebox	2397, 2399, 3219, 3379, 4062, 4652, 4910
\makelabel	89, 91, 94, 110
\makelabel	89, 93, 3201, 3217, 3363, 3377
\makesavenoteenv	455
mark-ans	2182
mark-pos	2182, 2224
mark-ref	2182
mini-env	1085
mini-sep	1085
\minipage	410
\miniright	11, 58, 1528, 1576, 1603, 3639, 3777
mode commands:	
\mode_if_math:TF	2663, 2717
\mode_if_vertical:TF	1164, 1193, 1214, 1238, 1389, 1410
\mode_leave_vertical:	878, 951, 965, 978, 991, 2395, 3174, 4646
msg commands:	
\msg_error:nn	1578, 1605, 2624, 2657, 2661, 2715, 2823, 3703, 3707, 3923, 3981, 4030, 4494, 4766, 4778, 5148, 5207
\msg_error:nnn	632, 679, 696, 749, 1532, 1539, 1544, 1574, 1601, 1870, 1874, 1989, 2606, 2665, 2683, 2695,

2703, 2707, 2711, 2719, 2761, 3266, 3284, 3306, 4498, 4771, 5011, 5020, 5090, 5195, 5226, 5235, 5272, 5293	
\msg_error:nnnn	2609, 2637, 2641, 2645, 2649, 2764, 3269, 3287, 3309, 3694, 4026, 4034, 4761, 5071, 5275
\msg_error:nnnnn	580, 600, 2205
\msg_fatal:nn	3504
\msg_fatal:nnn	488
\msg_info:nnn	13, 16, 21, 24, 435, 451
\msg_line_context:	5353, 5358, 5363, 5392, 5397, 5402, 5417, 5432, 5436, 5440, 5444, 5448, 5452, 5459, 5466, 5472, 5486, 5490, 5495, 5499, 5503, 5507, 5512, 5516, 5520, 5524, 5529, 5564, 5568, 5573, 5578, 5582, 5587, 5663, 5667, 5672, 5677, 5682, 5686, 5690, 5694, 5698, 5702, 5706, 5710, 5714
\msg_log:nnn	2009, 2014, 2019
\msg_log:nnnnn	397, 2152, 2157, 2162
\msg_log:nnnnnn	389
\msg_new:nnn	5320, 5324, 5328, 5332, 5337, 5350, 5355, 5360, 5365, 5374, 5382, 5386, 5390, 5395, 5400, 5415, 5430, 5434, 5438, 5442, 5446, 5450, 5454, 5463, 5469, 5475, 5479, 5483, 5488, 5493, 5497, 5501, 5505, 5510, 5514, 5518, 5522, 5527, 5562, 5566, 5571, 5576, 5580, 5585, 5661, 5665, 5670, 5675, 5680, 5684, 5688, 5692, 5696, 5700, 5704, 5708, 5712
\msg_new:nnnn	5341, 5532, 5541, 5550, 5556, 5589, 5599, 5609, 5619, 5629, 5639, 5649, 5655
\msg_term:nnnn	1973, 1978, 3446, 3456, 3488, 3493
\msg_term:nnnnn	2133
\msg_warning:nn	3638, 3776
\msg_warning:nnnn	2170, 2176, 3394, 3399, 4213, 4226, 4262, 4275
\msg_warning:nnnnn	2128, 2138
\multicolsep	99
\multicolsep	1231, 1403, 3586, 3749

N

\NeedsTeXFormat	3
\NewCommandCopy	406
\newcounter	491
\NewDocumentCommand	1528, 2612, 4022, 4995, 5050, 5155, 5204, 5282
\NewDocumentEnvironment	3668, 3802, 3969, 4463, 4732
\newenvsc	2726
\newlabel	37
\newlabel	473
no-store	2023
\noindent	3627, 4321, 4386, 4705, 4971
\nointerlineskip	1240, 1243, 1412, 1415, 1566, 1593, 4321, 4386
noitemsep	825
\nopagebreak	1176, 1205, 1240, 1243, 1412, 1415, 1519, 1525
\normalfont	2571, 3096, 3109, 4885
nosep	825

P

Packages:	
caption	114
enumext	27, 38, 40, 66, 95, 105, 133, 134
enumitem	38
expl3	110
footnotehyper	37
hyperref	31, 32, 37, 78, 87, 119, 133
ltxcmd	36
lua-visual-debug	52
multicol	27, 133
scontents	27, 81, 82

shortlst	110, 116, 120
\par ..	1176, 1205, 1243, 1415, 1519, 1525, 1561, 1566, 1588, 1593, 2547, 3603, 3764, 3782, 4015, 4018, 4144, 4348, 4363, 4409, 4423, 4705, 4971
para commands:	
\para_end:	4722, 4989
\parbox	2189
\parindent	4685, 4951
\parsep	51, 106
\parsep	883, 3479, 3946, 3955, 3959
parsep	825
\parskip	4686, 4952
\partopsep	3480, 3780, 3966
partopsep	825
peek commands:	
\peek_meaning:N	4563, 4577, 4592, 4603, 4813, 4827, 4842
\peek_meaning_remove:N	4570, 4820
\peek_remove_spaces:n	3343
\phantomsection	37
\phantomsection	462
prg commands:	
\prg_do_nothing:	466
\prg_new_protected_conditional:Npnn	217
\prg_replicate:nn	234
\prg_return_false:	221
\prg_return_true:	220
\printkeyans	18, 128, 5050
prop commands:	
\prop_const_from_keyval:Nn	5196
\prop_count:N	391, 2321, 2472, 2574, 2986, 3099, 3112, 4888, 5298
\prop_get:NnNTF	5222
\prop_gput_if_not_in:Nnn	2319
\prop_if_exist:N	2007, 5015, 5291
\prop_item:Nn	5017, 5315
\prop_new:N	2010
\ProvidesExplPackage	4

R

\raggedcolumns	3589, 3752
\raisebox	4093
\ref	76, 86
ref	607, 654, 727
\refstepcounter	4631, 4903
regex commands:	
\regex_match:nnTF	219, 776, 778, 790, 792, 2819
\regex_replace_once:nnN	227
\renewcommand	642, 687, 704, 757
\RenewDocumentCommand	1576, 1603, 3182, 3201, 3217, 3339, 3363, 3377, 3979, 4438
\RequirePackage	17, 25
resume	1719
resume*	1719
rightmargin	898
\Roman	39, 43, 44
\Roman	510
\roman	39, 43, 44
\roman	511, 625, 5040

S

\s	2820
save-ans	1962
save-key	2260
save-ref	2182

save-sep	2182
scan commands:	
\scan_stop:	3977, 4476, 4745, 5006, 5009
scontents internal commands:	
\l__scontents_fname_out_tl	2779
__scontents_parse_environment_keys:n	2785
__scontents_rescan_tokens:n	2792
\l__scontents_storing_bool	2777
\l__scontents_writing_bool	2778
seq commands:	
\seq_clear:N	5157, 5300
\seq_const_from_clist:Nn	5150
\seq_count:N	392, 3998, 5161
\seq_gclear:N	4436, 4437
\seq_gput_right:Nn	2328, 4449, 4450
\seq_if_empty:N	4455, 5065, 5175
\seq_if_exist:N	2012, 5063
\seq_if_in:NnTF	5069
\seq_item:Nn	2817, 4137
\seq_map_function:NN	5166
\seq_map_inline:Nn	5078, 5084, 5176, 5177
\seq_map_pairwise_function:NNN	4457
\seq_new:N	124, 125, 127, 141, 171, 172, 2015
\seq_pop_left:NN	5165
\seq_put_right:Nn	4036, 5173, 5189, 5310
\seq_set_from_clist:Nn	5158
\seq_set_map_e:NNn	5167
\seq_use:Nn	199, 200, 5306
series	1719
\setcounter	787, 791, 793, 3437, 3482, 4012
\setenumext	7, 129, 5155
\setenumextmeta	7, 131, 5196
show-ans	2182, 2224
show-length	997
show-pos	2224
skip commands:	
\skip_add:Nn	1135, 1144, 1153, 1166, 1170, 1195, 1199, 1216, 1274, 1276, 1290, 1293, 1314, 1316, 1330, 1333, 1353, 1355, 1369, 1372, 1391, 1440, 1441, 1452, 1454, 3955, 3962
\skip_gset:Nn	1467, 1471, 1475
\skip_gset_eq:NN	3959
\skip_gzero_new:N	1462, 1463
\skip_horizontal:N	966, 979, 992, 4649, 4661, 4709, 4919, 4975
\skip_horizontal:n	952, 2396, 2404, 3175, 3177, 4552, 4647, 4802, 4946
\skip_if_eq:nnTF	1133, 1142, 1151, 1260, 1300, 1340, 1428, 1464, 1486, 1627, 1641, 1655, 1666, 1677, 1688, 1699, 1710
\skip_new:N	81, 82, 83, 88, 89, 90, 91, 92, 93, 147, 191
\skip_set:Nn	1118, 1122, 1181, 1185, 1210, 1263, 1264, 1282, 1303, 1304, 1322, 1342, 1343, 1361, 1385, 1431, 1432, 1446, 1466, 1470, 1488, 1492, 1496, 1502, 1506, 1510, 3939
\skip_set_eq:NN	1221, 1222, 1224, 1231, 1396, 1397, 1398, 1403, 3435, 3478, 3479, 4686, 4952
\skip_sub:Nn	1270, 1272, 1286, 1288, 1310, 1312, 1326, 1328, 1349, 1351, 1365, 1367, 1438, 1439, 1450, 1451
\skip_use:N	1120, 1124, 1168, 1172, 1177, 1197, 1201, 1212, 1218, 1628, 1632, 1635, 1642, 1646, 1649, 3603
\skip_vertical:N	421, 424, 890, 4347, 4361, 4724, 4991
\skip_vertical:n	889, 4723, 4990

\skip_zero:N	1230, 1244, 1382, 1383, 1384, 1402, 1416, 3480, 3586, 3749, 3966, 3967
\skip_zero_new:N	1461, 1483, 1484, 1485
\c_zero_skip	421, 424, 890, 1133, 1142, 1151, 1301, 1340, 1464, 1486, 1628, 1642, 1655, 1666, 1677, 1688, 1699, 1710, 4347, 4361, 4724, 4991
\small	5028, 5032, 5036, 5040, 5044, 5048
socket commands:	
\socket_assign_plug:nn	3848, 3856, 3864, 3898, 3906, 3914
\socket_new:nn	3820, 3868
\socket_new_plug:nnn	3821, 3828, 3836, 3869, 3876, 3885
\socket_use:n	3849, 3899
\socket_use:nn	3857, 3865, 3907, 3915
\star	3240
start	799
start*	799
start-list-tags	3820, 3868
\stepcounter	3949, 4086, 4442
stop-list-tags	3820, 3868
stop-start-tags	3820, 3868
str commands:	
\c_backslash_str	2665, 5353, 5358, 5363, 5368, 5370, 5372, 5377, 5379, 5477, 5481, 5485, 5495, 5499, 5507, 5508, 5512, 5524, 5525, 5529, 5530, 5551, 5553, 5557, 5559, 5587, 5650, 5652, 5656, 5658, 5667, 5668, 5672, 5677, 5678, 5682, 5686, 5690
\c_colon_str	2471, 2985, 5006
\c_left_brace_str	5458, 5465, 5471
\c_right_brace_str	5458, 5465, 5471
\str_case:nn	276, 335
\str_case:nnTF	1742, 1750, 2299, 2307, 5104, 5113
\str_clear:N	3513, 4508
\str_count:n	234
\str_if_empty:N	1759, 1800, 1827
\str_if_eq:nnTF	3438, 3484, 5206
\str_if_in:nnTF	5002
\str_new:N	84, 131, 146, 186
\str_set:Nn	565, 571, 577, 596, 597, 598, 2202, 2203, 2229, 2230, 3929, 3932
\str_use:N	3221
\string	455
\strutbox	1249, 1252, 1263, 1264, 1275, 1277, 1292, 1295, 1303, 1304, 1315, 1317, 1332, 1335, 1342, 1343, 1354, 1356, 1371, 1374, 1420, 1423, 1431, 1432, 1440, 1441, 1453, 1455, 1466, 1467, 1470, 1477, 1490, 1498, 1504, 1512, 3957, 3962, 4015, 4099

T

tag commands:	
\tag_mc_begin:n	3826, 3874, 3883
\tag_mc_end:	3830, 3878, 3887
\tag_resume:n	3823, 3871, 3985, 3993, 4056, 4141, 4331, 4395
\tag_struct_begin:n	3824, 3825, 3832, 3833, 3834, 3872, 3873, 3880, 3881, 3882, 3994
\tag_struct_end:	4008, 4009
\tag_struct_end:n	3831, 3838, 3839, 3840, 3841, 3879, 3888, 3889, 3890, 3891, 4482, 4751
\tag_suspend:n	3842, 3892, 3975, 3987, 4003, 4047, 4133, 4474, 4743
\tag_tool:n	3986
TeX and L ^A T _E X 2 _ε commands:	
\@auxout	471

\@currentvir	276, 335
\protected@write	471
tex commands:	
\tex_newlinechar:D	2791
text commands:	
\text_expand:n	4998
\textasteriskcentered	2199, 2216
\the	243, 248, 260, 265
\thepage	477
tl commands:	
\c_space_tl	3067, 5402, 5417, 5440, 5444, 5631, 5632, 5641, 5642, 5702, 5706
\tl_clear:N	563, 570, 2180, 2246, 2256, 2277, 2285, 2491, 2811, 2812, 2926, 3002, 4848
\tl_clear_new:N	520
\tl_const:Nn	50, 504
\tl_gclear:N	383, 384, 385, 1780, 1785, 2901, 3212, 3230, 4367, 4427, 4650
\tl_gclear_new:N	1767
\tl_gput_right:Nn	505
\tl_greplace_all:Nnn	526
\tl_gset:Nn	311, 312, 326, 327, 1768, 1781, 1786, 2005, 2815, 3151, 4598
\tl_gset_eq:NN	522, 3147, 4643
\tl_if_blank:nTF	2604, 2622, 2759, 3264, 3282, 3304, 4641, 5270
\tl_if_empty:nTF	630, 649, 677, 694, 714, 721, 747, 764, 1793, 1798, 1820, 1825, 1883, 1947, 1955, 1984, 2044, 2335, 2366, 2511, 2856, 2878, 2908, 2936, 3012, 3061, 3172, 4851, 5187
\tl_if_empty:nTF	1848
\tl_if_exist:nTF	1853
\tl_if_novalue:nTF	2618, 2934, 3010, 3046, 3126, 3145, 3153, 3314, 3511, 3996, 4440, 4506, 4783, 4849
\tl_map_inline:Nn	225, 523
\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, 132, 135, 136, 154, 162, 163, 164, 167, 185
\tl_put_left::Ne	2845
\tl_put_left:Nn	2343, 2374, 2496, 2839, 2852, 2858, 2868, 3078, 3118, 4351, 4412, 4870, 4873
\tl_put_right:Nn	521, 640, 685, 702, 755, 2347, 2378, 2425, 2435, 2448, 2463, 2469, 2474, 2498, 2503, 2510, 2513, 2523, 2528, 2531, 2537, 2929, 2932, 2938, 2943, 2970, 2975, 2980, 2983, 2992, 3005, 3008, 3014, 3019, 3029, 4853, 4857
\tl_remove_all:Nn	5186
\tl_remove_once:Nn	2413, 2955
\tl_replace_all:Nnn	525, 5221
\tl_reverse:N	2412, 2414, 2954, 2956
\tl_set:Nn	58, 280, 290, 339, 340, 347, 348, 355, 356, 490, 564, 569, 575, 576, 629, 674, 746, 949, 963, 976, 989, 1882, 1983, 2247, 2257, 2278, 2286, 2568, 2779, 3048, 3093, 3106, 4859, 4882, 5184, 5220, 5290
\tl_set_eq:NN	531, 635, 638, 682, 684, 699, 701, 752, 754, 2411, 2953, 2966, 3326, 3331, 4074, 4076
\tl_to_str:n	1853, 1859, 1864, 4998
\tl_trim_spaces:n	521, 5173, 5184, 5190, 5206
\tl_use:N	527, 530, 651, 716, 723, 766, 1023, 1027, 1031, 1035, 1039, 1043, 1047, 1051, 1055, 1059, 1063, 1067, 1071, 1075, 1079, 1083, 2401, 2418, 2426, 2437, 2450, 2455, 2466, 3134, 3140, 3168, 3203, 3204, 3211, 3223, 3317, 3321, 3329, 3365, 3366, 3372, 3381, 3675, 3808, 4079, 4358, 4419, 4654, 4683, 4684, 4912, 4941, 4944,

4949, 5053, 5054, 5055, 5056, 5057, 5074, 5169, 5288

token commands:

\token_to_str:N	473
\topsep	3780, 3962
topsep	825
\topskip	1230, 1402
\typeout	243, 248, 260, 265, 270, 439, 443, 454, 455

U

\u	228, 2820
\unkern	249, 266
unknown	3250, 3272, 3290
\unskip	244, 261

use commands:

\use:N	235, 3208, 3227, 3677
\use:n	1733, 2290, 5004, 5095
\use_none:nn	465, 5227
\usecounter	3436, 3481

V

\value	1796, 1802, 1809, 1815, 1823, 1829, 1836, 1842
--------	--

vbox commands:

\vbox_set:Nn	4049
\vbox_set_top:Nn	4356, 4417
\vspace	883, 1632, 1635, 1646, 1649, 1659, 1661, 1670, 1672, 1681, 1683, 1692, 1694, 1703, 1705, 1714, 1716

W

widest	799
wrap-ans	2182
wrap-label	534
wrap-label*	534
wrap-opt	2182

Z

\z	2820
----	------