

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-07-15^{*}

©2024 by Pablo González[†]

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

Abstract

This package provides “*enumerated list*” environments for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the `<answers>` to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	11
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	13
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	14
2	The environments provided	4	6.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	6.6	Printing stored content	16
2.3.1	Keys for <code>\item*</code>	5	6.6.1	The command <code>\getkeyans</code>	16
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6.6.2	The command <code>\foreachkeyans</code>	16
3	The command <code>\setenumext</code>	6	6.6.3	The command <code>\printkeyans</code>	16
4	The command <code>\setenumextmeta</code>	6	7	Full examples	17
5	The <code>keyval</code> system	6	8	The way of non-enumerated lists	20
5.1	Keys for label and ref	6	9	References	22
5.2	Keys for spaces	7	10	Change history	22
5.2.1	Vertical spaces	7	11	Index of Documentation	23
5.2.2	Horizontal spaces	8	12	Implementation	25
5.3	Keys for add code	9	13	Index of Implementation	131
5.4	Keys for start, series and resume	9			
5.5	Keys for <code>multicols</code>	10			
5.6	Keys for <code>minipage</code>	10			
5.6.1	The command <code>\miniright</code>	10			
5.6.2	The key <code>mini-right</code>	10			

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 2024-07-15.

[†]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/lpp1.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

C) value

B) correct

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

D) I and III only

B) II only

E) I, II, and III

C) I and II only

★ 3. Third type of questions

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

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

A) value

D) value

B) value

E) value

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

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B

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

1. B), $x = 5$

2. D)

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

* 5. D), “other note”

*

These “*simple worksheets*” or “*multiple choice questions*” appear to be easy to obtain using a combination of the `enumerate`, `minipage` and `multicols` environments, but like many things, what “*looks simple*” is not so simple.

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

1.1 Description and usage

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

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

enumext.sty

enumext.pdf

README.md

enumext.dtx

»

»

»

»

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

\usepackage{enumext}

1.2 The concept of left margin

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

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



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

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

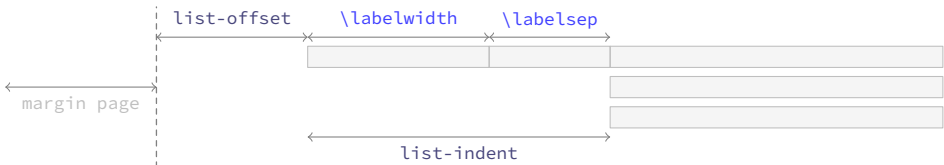


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

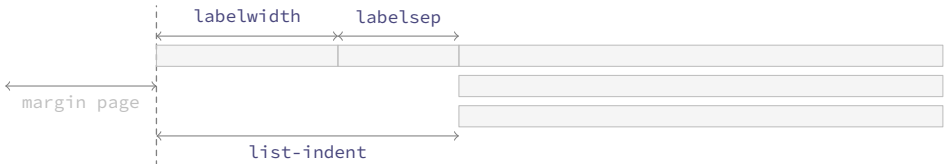


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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



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

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem`[6] package to customize the references which is activated by the `ref` key (§5.1), the standard \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.

- The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[8] package.

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*}
```

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 `TeX`, `\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.

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.
- ★ 3. This text is in the first level.

2.3 The command \item*

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

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

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

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

2.3.1 Keys for `\item*`

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

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

2.4 The command `\item` in `enumext*`

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

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

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

3 The command `\setenumext`

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

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

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

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

4 The command `\setenumextmeta`

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

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

The starred version `*` will create the new “meta-key” for the environment `enumext*` and for all levels of the environment `enumext`.

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 reference*” 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` 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* = {⟨code {#1} more code⟩}`

default: *empty*

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

5.2 Keys for spaces

`show-length = {⟨true | false⟩}`

default: *false*

Displays on the terminal the values for *all list parameters* at the current level. For *vertical spaces* show the values of `\topsep`, `\itemsep`, `\parsep` and `\partopsep`. For *horizontal spaces* show the values of `\labelwidth`, `\labelsep`, `\itemindent`, `\listparindent` and `\leftmargin`.

5.2.1 Vertical spaces

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

default: *by levels*

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

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

default: *by levels*

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

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

default: *by levels*

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

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

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

default: *by levels*

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

`noitemsep` *⟨value forbidden⟩*

default: *not used*

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

`nosep` *⟨value forbidden⟩*

default: *not used*

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

`base-fix` *⟨value forbidden⟩*

default: *not used*

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

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

default: *not used*

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

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

default: *not used*

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

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

default: *not used*

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

`below*` = { $\langle rubber\ length \mid rigid\ length \rangle$ } 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 \LaTeX which is defined (simplified) as plain form `\list{\langle arg one \rangle}{\langle arg two \rangle}`. Using the `before*` key does not allow access to the `list` parameters defined by $[\langle key = val \rangle]$.

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for multicols

`columns = {⟨integer⟩}` default: `1`

Set the *number of columns* to be used by the `multicols` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep = {⟨rigid length⟩}` default: *by level*

Set the *space between columns* used by the `multicols` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

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

5.6 Keys for minipage

`mini-env = {⟨rigid length⟩}` default: *not used*

Sets the *width* of the `minipage` environment on the “right side”. This value added to the value set by the `mini-sep` key to determines the *width* of the `minipage` environment on the “left side”, taking `\linewidth` as the maximum reference value.

`mini-sep = {⟨rigid length⟩}` default: `0.3333em`

Sets the *space between* the `minipage` environment on the “left side” and the `minipage` environment on the “right side”. This separation is applied together with `\hfill`.

5.6.1 The command `\miniright`

```
\miniright \begin{enumext}[mini-env=⟨rigid length⟩] ⟨item's before⟩ \item \miniright ⟨content⟩ \end{enumext}
\begin{enumext}[mini-env=⟨rigid length⟩] ⟨item's before⟩ \item \miniright*⟨content⟩ \end{enumext}
```

The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”.

The *starred argument* “*” inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `minipage` on the “right side”.

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

5.6.2 The key `mini-right`

In the horizontal list environments `enumext*` and `keyans*` it is not possible to use the `\miniright` command and the `mini-right` key must be used instead.

`mini-right = {⟨content⟩}` default: *not used*

Set the *content* for the drawing or tabular to be placed in the `minipage` environment on the “right side” by starting it with `\centering`. The `{⟨content⟩}` must be passed between braces.

`mini-right* = {⟨content⟩}` default: *not used*

Same as above, but *without* starting with `\centering`.

6 The storage system

The entire mechanism for “*storing content*” it is activated according to `save-ans` key on the “*first level*” of `enumext` or `enumext*` environments and it is ignored if they are established when they are nested inside each other. Only when this $\langle key \rangle$ 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 arguments passed to the inner levels or the environment nested in it, along with the content passed to `\anskey`, the current $\langle labels \rangle$ for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be stored in a $\langle sequence \rangle$ and at the same time will be stored (without the environment structure or optional arguments) in a $\langle prop list \rangle$.

The optional arguments of the inner levels or the nested environment are filtered by excluding all $\langle keys \rangle$ related to the “*stored system*” along with the keys `series`, `resume` and `resume*` when storing in $\langle sequence \rangle$.

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 $\langle sequence \rangle$ and $\langle prop list \rangle$ in which the contents will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the $\langle sequence \rangle$ or $\langle prop list \rangle$ 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 arguments of the inner levels or nested environment that will be passed to the $\langle sequence \rangle$. The $\langle key list \rangle$ passed to this key ignores any $\langle keys \rangle$ in the “*stored system*” 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 $\langle keys \rangle$ that will be stored by default in the $\langle sequence \rangle$ would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the $\langle sequence \rangle$ only the key `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 `keyans`, `keyans*` and `keyanspic` environments and storing them in the $\langle store name \rangle$ defined by the `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 $\langle store name \rangle$ 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 $\langle store name \rangle$ 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` 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` = { $\langle code \rangle$ { $\#1$ } *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 { $\langle code \rangle$ } must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{ $\#1$ }’.
- `wrap-opt` = { $\langle code \rangle$ { $\#1$ } *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 { $\langle code \rangle$ } must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{ $\#1$ }’.
- `show-ans` = { $\langle true | false \rangle$ } default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the $\langle label \rangle$ 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` = { $\langle symbol \rangle$ } 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` = { $\langle left | right \rangle$ } 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` = { $\langle true | false \rangle$ } default: `false`
 Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* { $\langle store name \rangle$ } 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` = { $\langle true | false \rangle$ } 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` $\langle value forbidden \rangle$ default: `not used`
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* { $\langle store name \rangle$ } set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey`, “without” use `anskey*`, “without” interfering with the `check-ans` key and “without” storing an unwanted structure in the *sequence* { $\langle store name \rangle$ }.

6.2 The command `\anskey`

`\anskey` $\langle anskey [\langle keys \rangle] \{ \langle content \rangle \}$

The command `\anskey` takes a mandatory non empty argument { $\langle content \rangle$ } and “stores” it in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. By design the command cannot be nested or passed *verbatim material* in the argument and it is assumed that each *numbered* `\item` or `\item*` within the environment in which it is active it has a “single execution” of `\anskey` unless `\item` or `\item*` open a nested level or use the `no-store` key.

If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L^AT_EX will be used.

The `\anskey` command is available for all levels of the `enumext` environment and the `enumext*` environment, but is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

6.2.1 Keys for `\anskey`

By default the { $\langle content \rangle$ } passed to `\anskey` when “storing” in the *sequence* { $\langle store name \rangle$ } has the form `\item \langle content \rangle`, the following $\langle keys \rangle$ allow modifying the way in which it is “stored” in the *sequence*.

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

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

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

Example

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

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

6.3 The environment `anskey*`

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

The environment `anskey*` takes a mandatory { $\langle body content \rangle$ } and “stores” it in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperLink` and `\hypertarget` will be used, otherwise the usual “*label and ref*” system provided by \LaTeX will be used.

By design the environment cannot be nested but full supports “*verbatim material*” in the 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 { $\langle keys \rangle$ } must be passed separated by commas and “without separation” of the start of the environment. Comments “%” or “any character” after `\begin{anskey*}` or [$\langle key = val \rangle$] on the same line are NOT supported, 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 { $\langle keys \rangle$ } 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*}
\end{enumext}
```

```

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

```

- | | |
|---|--|
| <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 `\keys` 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 `\keys` 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 `\label` 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 `\label` set by `label` key next to the `\content` (if it is present) 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
  \end{keyans*}
\end{enumext}

```




```
\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
C) Choice
E) Choice

* B) Correct choice
D) Choice
2. Text containing a question and image.

A) Choice
B) Choice
C) Choice
D) Choice
* E) [note] Correct choice


Some text

6.5 The environment keyanspic

```
keyanspic \begin{keyanspic}[\langle n^{\circ} above, n^{\circ} below \rangle]{\langle drawing \rangle}{\langle content \rangle}{\langle drawing \rangle}
```

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

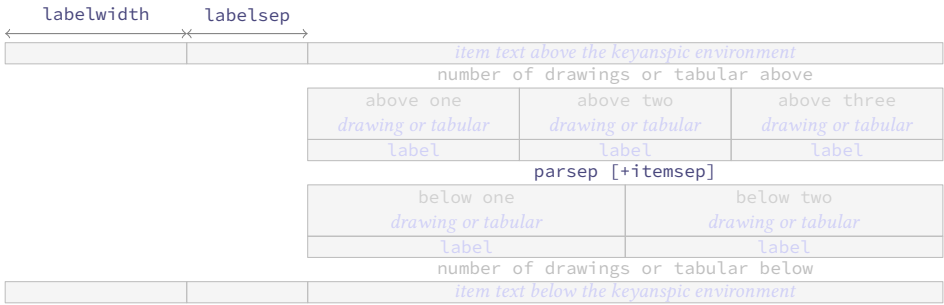


Figure 6: Representation of the `keyanspic` environment with optional argument `[3,2]` in `enumext`.

The optional argument determines the number drawings or tabular “above” and “below” within the environment. The vertical separation between “above” and “below” is controlled by the values set by `parsep` and `itemsep` keys passed to `keyans` environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

6.5.1 The command \anspic

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

The `\anspic` command take three arguments, the *starred argument* ‘`*`’ store the current `\label` next to the `\content` (if it is present) in *sequence* and *prop list* `{\langle store name \rangle}` set by `save-ans` key.

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

Example

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

1. Question with images.



A)



B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

```
\foreachkeyans <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 increment (`<step>`) applied to the value set by key `start` for each element 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> [<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 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* $\{\langle store\ name\rangle\}$ it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

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

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

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[\langle print\rangle,\langle i\rangle]{\langle keys\rangle}` and `\setenumext[\langle print*\rangle]{\langle keys\rangle}`. If we need to set the $\langle keys\rangle$ for the environment `enumext` “saved” in the *sequence* $\{\langle store\ name\rangle\}$ we will use `\setenumext[\langle print\rangle,\langle level\rangle]{\langle keys\rangle}` and if we need to set the $\langle keys\rangle$ for the environment `enumext*` “saved” in the *sequence* $\{\langle store\ name\rangle\}$ we will use `\setenumext[\langle print\rangle,\langle *\rangle]{\langle keys\rangle}`.

Example

```
\begin{enumext}[save-ans=sample,columns=2,show-pos=true,nosep,save-ref=true]
  \item Factor  $3x+3y+3z$ . \anskey{$3(x+y+z)$}
  \item True False

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]

2. True False

(a)

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*


7 Full examples

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

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

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

Example 1

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D


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

2. A

3. B

4. A

Example 2

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

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

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

A

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

B

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

C

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

D

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

2. A

3. B

4. A


*

*

*

*

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

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

- 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 B C D E
5. Question with image on left side:

A value

B value

C value

D correct

E value
-
- Test keys
1. B, $x = 5$

2. D

3. C, some note

* 4. E, A duck

* 5. D, other note

*
- Example 4
- A “simple worksheet” using ducks :) 🦆
- Factor $x^2 - 2x + 1$
- Factor $3x + 3y + 3z$
- The following questions need to be cuaqtified :)
- True False
- (a) $\alpha > \delta$

(b) ~~ETX~~e 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!

(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.

2
- ©2024 by Pablo González L
- 19 / 145

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

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

- | | |
|---------------------|---------------------|
| • First level item | * First level item |
| – Second level item | ◇ Second level item |
| * Third level item | ○ Third level item |
| · Fourth level item | ★ Fourth level item |
| • First level item | * First level item |

Fake description environment

Here we set `label={}` and `list-indent=2.5em`, `font=\bfseries`.

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

If we add `list-indent=0pt` you get *widest style*:

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- 🟢 The small space at the beginning of the “*unlabeled entry*” corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.

Description indented by label

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

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


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

- Something

A short one-line description.
This is an entry *without* a label.
- Something

A short one-line description.
- Something long

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

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

- Something

A short one-line description.
This is an entry *without* a label.
- Something

A short one-line description.
- Something long

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

If we add `align=right` it will look like this:

- Something

A short one-line description.
This is an entry *without* a label.
- Something

A short one-line description.
- Something long

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

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

Description with multi-line labels

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

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

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

- Something

A short one-line description.
This is an entry *without* a label.
- Something

A short one-line description.
- Something long

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

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of `enumext` were some macros using the `enumerate[5]` package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in [Create a fake label ref using list](#) and the answer given by David Carlisle in [Change the use of label ref by data save in an array \(list\)](#) I managed to create a more solid code than the original version, now using the `l3prop[11]` and `l3seq[11]` modules together with the `hyperref[8]` and `enumitem[6]` packages, which did the job, but with some limitations.

As time went by I took these limitations as a personal challenge which I called “*reinventing the wheel*”, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This “*reinventing the wheel*” finally ended up becoming `enumext`.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have

in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with \LaTeX , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

9 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ $\text{\LaTeX} 2_{\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The $\text{\LaTeX} 3$ Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The $\text{\LaTeX} 2_{\epsilon}$ sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.
- [13] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2024.
- [14] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [15] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [16] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

10 Change history

v1.0 2024-07-15 – First public release.

11 Index of Documentation

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

C

Document class:

article 2

book 2

exam 2

letter 2

report 2

\columnbreak 4, 12

\columnsep 10

Commands provide by enumext:

\anskey 11–13

\anspic 11, 12, 15

\foreachkeyans 16

\getkeyans 12, 16

\item* 5–7, 11, 12, 14, 15

\item 5–10, 12, 14

\miniright 10

\printkeyans 6, 11, 16

\setenumextmeta 6

\setenumext 5–7, 11, 12, 14, 17

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

anskey* 11–13

enumext* 4–14, 16, 17

enumext 4–14, 16, 17, 20

keyans* 4–14

keyanspic 4, 7, 8, 11–13, 15, 20

keyans 4–9, 11–15, 20

Environments:

Verbatim 13

enumerate 1, 3, 5, 21

figure 5

list 3, 9, 22

minipage 3–5, 10, 22

multicols 3, 4, 10

table 5

task 5

F

\footnote 5

I

\itemsep 8

K

Keys for \anskey provide by enumext:

break-col 12

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

Keys for \foreachkeyans provide by enumext:

after 16

before 16

sep 16

start 16

step 16

stop 16

wrapper 16

Keys for anskey* provide by enumext:

break-col 12

force-eol 13

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

overwrite 13

write-env 13

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 21

base-fix 8

before* 9

before 9

below* 9

below 8

check-ans 12

columns-sep 4, 10

columns 4, 8, 10

first 9

font 7

item-pos* 5, 6

item-sym* 5, 6

itemindent 9

itemsep 8, 15

labelsep 3–7, 9, 10, 12, 20, 21

labelwidth 3, 4, 6, 7, 9, 10, 12, 20, 21

labelwith 5

label 7, 9, 14, 20, 21

list-indent 3, 9

list-offset 3, 9, 21

listparindent 9

mark-ans 12

mark-pos 12

mark-ref 11

mini-env 4, 5, 8, 10

mini-right* 7, 10

mini-right 7, 10

mini-sep 4, 10

no-store 11–13

noitemsep 8

nosep 8, 20

overwrite 13

parsep 8, 15

partopsep 8

ref 4, 7

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–16

©2024 by Pablo González L

23 / 145

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

12 Implementation

The most recent publicly released version of `enumext` is available at CTAN: <https://www.ctan.org/pkg/enumext>. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: <https://github.com/pablgonz/enumext/issues>.

- The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a TeXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

12.1 General conventions

Variables containing `i`, `ii`, `iii` and `iv` are associated by level with the `enumext` environment, variables containing `v` are associated with the `keyans` environment, variables containing `vi` are associated with the `keyanspic` environment, variables containing `vii` are associated with the `enumext*` environment and variables containing `viii` are associated with the `keyans*` environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital “X”.

The temporary function `__enumext_tmp:n` is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

12.2 Initial set up

Start the DocStrip guards.

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

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

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

12.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L^AT_EX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-07-15}
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 `\l__enumext_current_widest_dim` and others.)

```
\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim
```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§12.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace`:NNNNNNNNNNNN (§12.37.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72 {
73   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
74   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
75   \dim_new:c { \l__enumext_leftmargin_#1_dim }
76   \dim_new:c { \l__enumext_itemindent_#1_dim }
77 }
78 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
```

Internal variables used by `columns` key §12.21).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80 {
81   \skip_new:c { \l__enumext_multicols_above_#1_skip }
82   \skip_new:c { \l__enumext_multicols_below_#1_skip }
83   \skip_new:c { \g__enumext_multicols_right_#1_skip }
84 }
85 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(End of definition for `\l__enumext_multicols_above_X_skip`, `\l__enumext_multicols_below_X_skip`, and `\g__enumext_multicols_right_X_skip`.)

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

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

```
86 \int_new:N \g__enumext_minipage_stat_int
87 \skip_new:N \l__enumext_minipage_left_skip
88 \skip_new:N \l__enumext_minipage_right_skip
89 \skip_new:N \l__enumext_minipage_after_skip
90 \skip_new:N \g__enumext_minipage_right_skip
91 \skip_new:N \g__enumext_minipage_after_skip
92 \cs_set_protected:Npn \__enumext_tmp:n #1
93 {
94   \dim_new:c { \l__enumext_minipage_left_#1_dim }
95   \bool_new:c { \l__enumext_minipage_active_#1_bool }
96 }
97 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§12.12), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§12.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§12.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§12.12). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§12.19).

```
98 \cs_set_protected:Npn \__enumext_tmp:n #1
99 {
100   \bool_new:c { \l__enumext_wrap_label_#1_bool }
101   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
102   \int_new:c { \l__enumext_start_#1_int }
103   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
104   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
105   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
106   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
107   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
108 }
109 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```

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

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§12.25.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§12.29) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the $\langle body\rangle$ and the $\langle keys\rangle$ of the environment `anskey*` (§12.30).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§12.36) and `\anspic*` (§12.40.1) 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.

```

110 \bool_new:N \l__enumext_store_active_bool
111 \tl_new:N \l__enumext_store_name_tl
112 \tl_new:N \g__enumext_store_name_tl
113 \tl_new:N \l__enumext_store_anskey_arg_tl
114 \tl_new:N \l__enumext_store_anskey_env_tl
115 \tl_new:N \l__enumext_store_anskey_opt_tl
116 \tl_new:N \l__enumext_store_current_label_tl
117 \tl_new:N \l__enumext_store_current_opt_arg_tl
118 \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).

```

119 \tl_new:N \l__enumext_setkey_tmpa_tl
120 \tl_new:N \l__enumext_setkey_tmpb_tl
121 \int_new:N \l__enumext_setkey_tmpa_int
122 \seq_new:N \l__enumext_setkey_tmpa_seq
123 \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
\l__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command (§12.49).

```

124 \tl_new:N \l__enumext_meta_path_tl
125 \seq_new:N \l__enumext_foreach_print_seq
126 \tl_new:N \l__enumext_foreach_name_prop_tl
127 \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*”.

```

128 \tl_new:N \l__enumext_print_keyans_starred_tl
129 \str_new:N \l__enumext_mark_position_str
130 \tl_new:N \g__enumext_item_symbol_aux_tl
131 \cs_set_protected:Npn \__enumext_tmp:n #1
132 {
133   \tl_new:c { \l__enumext_print_keyans_#1_tl }
134   \tl_new:c { \l__enumext_store_save_key_#1_tl }
135   \bool_new:c { \l__enumext_store_save_key_#1_bool }
136   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
137 }
138 \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_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

```

Internal variables used by `keyanspic` environment (§12.40.2).

```

139 \seq_new:N \l__enumext_keyans_pic_body_seq
140 \dim_new:N \l__enumext_keyans_pic_width_dim
141 \int_new:N \l__enumext_keyans_pic_above_int
142 \int_new:N \l__enumext_keyans_pic_below_int
143 \skip_new:N \l__enumext_keyans_pic_above_skip

```

(End of definition for `\l__enumext_keyans_pic_body_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.

```

144 \bool_new:N \l__enumext_check_answers_bool
145 \bool_new:N \g__enumext_check_ans_key_bool
146 \tl_new:N \l__enumext_check_start_line_env_tl
147 \int_new:N \g__enumext_check_starred_cmd_int
148 \int_new:N \g__enumext_item_anskey_int
149 \int_new:N \g__enumext_item_number_int
150 \bool_new:N \l__enumext_item_number_bool
151 \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`.

```

152 \bool_new:N \l__enumext_hyperref_bool
153 \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 `⟨labels⟩` defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§12.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

154 \tl_new:N \l__enumext_newlabel_arg_one_tl
155 \tl_new:N \l__enumext_newlabel_arg_two_tl
156 \tl_new:N \l__enumext_write_aux_file_tl
157 \cs_set_protected:Npn \__enumext_tmp:n #1
158 {
159   \tl_new:c { \l__enumext_label_copy_#1_tl }
160 }
161 \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).

```

162 \int_new:N \g__enumext_footnote_int
163 \seq_new:N \g__enumext_footnote_arg_seq
164 \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.

```

165 \cs_set_protected:Npn \__enumext_tmp:n #1
166 {
167   \bool_new:c { \l__enumext_item_starred_#1_bool }
168   \int_new:c { \l__enumext_item_column_pos_#1_int }
169   \int_new:c { \g__enumext_item_count_all_#1_int }
170   \int_new:c { \l__enumext_joined_item_#1_int }
171   \int_new:c { \l__enumext_joined_item_aux_#1_int }
172   \int_new:c { \l__enumext_tmpa_#1_int }
173   \dim_new:c { \l__enumext_tmpa_#1_dim }
174   \box_new:c { \l__enumext_item_text_#1_box }
175   \dim_new:c { \l__enumext_joined_width_#1_dim }
176   \dim_new:c { \l__enumext_item_width_#1_dim }
177   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
178   \str_new:c { \l__enumext_align_label_#1_str }
179   \bool_new:c { \g__enumext_minipage_active_#1_bool }
180   \box_new:c { \l__enumext_miniright_code_#1_box }
181   \bool_new:c { \g__enumext_minipage_center_#1_bool }
182   \dim_new:c { \g__enumext_minipage_right_#1_dim }
183   \skip_new:c { \g__enumext_minipage_right_#1_skip }
184 }
185 \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`.

```

186 \clist_const:Nn \c__enumext_all_envs_clist
187 {
188   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
189   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
190 }

```

(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).

```

191 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
192 \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.

```

193 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
194 {
195   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
196 }

```

(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`

```

197 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
198 {
199   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
200 }
201 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
202 {
203   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
204 }

```

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

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

```

205 \cs_new:Nn \__enumext_level:
206 {
207   \int_to_roman:n { \__enumext_level_int }
208 }

```

(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`

```

209 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
210 {
211   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
212   { \prg_return_true: }
213   { \prg_return_false: }
214 }

```

(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.

```

215 \cs_new_protected:Nn \__enumext_regex_counter_style:
216 {
217   \tl_map_inline:Nn \c__enumext_counter_style_tl
218   {
219     \regex_replace_once:nnN { \c{##1}\* }
220     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
221   }
222 }

```

(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.

```

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

```

(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.

```

229 \cs_new_protected:Npn \__enumext_unskip_unkern:
230 {
231     \int_case:nnT { \lastnodetype }
232     {
233         { 11 }
234         {
235             \typeout{SKIIIIIIIIIIIIIIIP}
236             \typeout{\the\lastskip}
237             \unskip
238         }
239         { 12 }
240         {
241             \typeout{KERRRRRRRRRRRRRRRRRN}
242             \typeout{\the\lastkern}
243             \unkern
244         }
245     }
246 }

```

(End of definition for `__enumext_unskip_unkern:`.)

12.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`

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

`__enumext_is_on_first_level:`

```

247 \cs_new_protected:Nn \__enumext_is_not_nested:
248 {
249     \str_case:en { \@currentenv }
250     {
251         {enumext}
252         {
253             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
254             \bool_lazy_and:nnT
255             { \bool_not_p:n { \g__enumext_standar_bool } }
256             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
257             {
258                 \bool_gset_true:N \g__enumext_standar_bool
259             }
260         }
261         {enumext*}
262         {
263             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
264             \bool_lazy_and:nnT
265             { \bool_not_p:n { \g__enumext_starred_bool } }
266             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
267             {
268                 \bool_gset_true:N \g__enumext_starred_bool
269             }
270         }
271     }
272 }

```

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.

```

273 \cs_new_protected:Nn \__enumext_is_on_first_level:
274 {
275   \bool_lazy_all:nT
276   {
277     { \bool_if_p:N \g__enumext_standar_bool }
278     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
279     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
280   }
281   {
282     \bool_set_true:N \l__enumext_standar_first_bool
283     \bool_set_true:N \l__enumext_anskey_env_bool
284     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
285     \tl_gset:Ne \g__enumext_start_line_tl
286     {
287       on ~ line ~ \exp_not:V \inputlineno
288     }
289   }
290   \bool_lazy_all:nT
291   {
292     { \bool_if_p:N \g__enumext_starred_bool }
293     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
294     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
295   }
296   {
297     \bool_set_true:N \l__enumext_starred_first_bool
298     \bool_set_true:N \l__enumext_anskey_env_bool
299     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
300     \tl_gset:Ne \g__enumext_start_line_tl
301     {
302       on ~ line ~ \exp_not:V \inputlineno
303     }
304   }
305 }

```

(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.

```

306 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
307 {
308   \str_case:en { \@currenvir }
309   {
310     {keyans}
311     {
312       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
313       \tl_set:Ne \l__enumext_check_start_line_env_tl
314       {
315         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
316       }
317     }
318     {keyans*}
319     {
320       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
321       \tl_set:Ne \l__enumext_check_start_line_env_tl
322       {
323         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
324       }
325     }
326     {keyanspic}
327     {
328       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
329       \tl_set:Ne \l__enumext_check_start_line_env_tl
330       {
331         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
332       }
333     }
334   }
335 }

```


(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:
336 \cs_new_protected:Nn \__enumext_reset_global_vars:
337 {
338   \__enumext_reset_global_int:
339   \__enumext_reset_global_bool:
340   \__enumext_reset_global_tl:
341 }
342 \cs_new_protected:Nn \__enumext_reset_global_int:
343 {
344   \int_gzero:N \g__enumext_item_number_int
345   \int_gzero:N \g__enumext_item_anskey_int
346   \int_gzero:N \g__enumext_item_answer_diff_int
347 }
348 \cs_new_protected:Nn \__enumext_reset_global_bool:
349 {
350   \bool_gset_false:N \g__enumext_check_ans_key_bool
351   \bool_gset_false:N \g__enumext_standar_bool
352   \bool_gset_false:N \g__enumext_starred_bool
353 }
354 \cs_new_protected:Nn \__enumext_reset_global_tl:
355 {
356   \tl_gclear:N \g__enumext_store_name_tl
357   \tl_gclear:N \g__enumext_start_line_tl
358   \tl_gclear:N \g__enumext_envir_name_tl
359 }
```

(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.

```
360 \cs_new_protected:Nn \__enumext_log_global_vars:
361 {
362   \msg_log:nneeee { enumext } { prop-seq-int-hook }
363   { \g__enumext_store_name_tl }
364   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
365   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
366   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
367 }
```

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.

```
368 \cs_new_protected:Nn \__enumext_log_answer_vars:
369 {
370   \msg_log:nneee { enumext } { item-answer-hook }
371   { \int_use:N \g__enumext_item_number_int }
372   { \int_use:N \g__enumext_item_anskey_int }
373   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
374 }
```

(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 L^AT_EX has the following plain form:

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

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

The functions `__enumext_start_list:nn`, `__enumext_stop_list:` and `__enumext_item_std:w` correspond to copies of `\list`, `\endlist` and `\item` from plain definition of `list` environment.

```
375 \__enumext_at_begin_document:n
376 {
```

```

377     \cs_new_eq:NN \__enumext_start_list:nn \list
378     \cs_new_eq:NN \__enumext_stop_list: \endlist
379     \cs_new_eq:NN \__enumext_item_std:w \item
380 }

```

(End of definition for `__enumext_start_list:nn`, `__enumext_stop_list:`, and `__enumext_item_std:w`.)

The `minipage` environment provided by L^AT_EX 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 `minipage` environment or a related command.

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

```

381 \__enumext_at_begin_document:n
382 {
383     \cs_new_eq:NN \__enumext_minipage:w \minipage
384     \cs_new_eq:NN \__enumext_endminipage: \endminipage
385 }

```

(End of definition for `__enumext_minipage:w` and `__enumext_endminipage:`.)

12.7 The internal minipage environment

`__enumext_internal_mini_page:` The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` 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)

```

386 \cs_new_protected:Nn \__enumext_internal_mini_page:
387 {
388     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
389     {
390         \DeclareDocumentEnvironment{\__enumext_mini_env*}{ m }
391         {
392             \__enumext_minipage:w [ t ] { ##1 }
393             \legacy_if_gset_false:n { @minipage }
394             \skip_vertical:N \c_zero_skip
395         }
396         {
397             \skip_vertical:N \c_zero_skip
398             \__enumext_endminipage:
399         }
400     }
401 }

```

(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.

```

402 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
403 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

`__enumext_after_hyperref:` The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `__enumext_footnotes_key_bool` to “true”.

```

404 \cs_new_protected:Nn \__enumext_after_hyperref:
405 {
406     \IfPackageLoadedTF { hyperref }
407     {
408         \msg_info:nnn { enumext } { package-load } { hyperref }
409         \bool_set_true:N \l__enumext_hyperref_bool
410         \IfHyperBoolean{hyperfootnotes}
411         {

```

```

412         \typeout{hyperfootnotes=true}
413         \bool_set_true:N \l__enumext_footnotes_key_bool
414     }
415     { \typeout{hyperfootnotes=false} }
416 }
417 { }

```

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`.

```

418 \bool_if:NT \l__enumext_footnotes_key_bool
419 {
420     \IfPackageLoadedTF { footnotehyper }
421     {
422         \msg_info:nnn { enumext } { package-load } { footnotehyper }
423     }
424     {
425         \typeout{No ~ footnotehyper ~ load}
426         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
427         \bool_set_false:N \l__enumext_footnotes_key_bool
428     }
429 }

```

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.

```

430 \bool_if:NTF \l__enumext_hyperref_bool
431 {
432     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
433     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
434 }
435 {
436     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
437     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
438 }
439 }

```

(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.

```

440 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
441 {
442     \protected@write \@auxout { }
443     {
444         \token_to_str:N \newlabel {#1}
445         {
446             {#2}
447             \bool_if:NT \l__enumext_hyperref_bool
448             { { \thepage } {#2} {#1} }
449             { }
450         }
451     }
452     \__enumext_hypertarget:nn {#1} { }
453     \__enumext_phantomsection:
454 }

```

(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.

```

455 \dim_zero_new:N \itemwidth

```

12.10 Definition of counters

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

To create the necessary “counters” we must first make sure that they are not already defined by the user or a package such as **enumitem**, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

- #1 : A token list `__enumext_counter_X_tl` for “store” the counter’s name.
- #2 : The counter’s name.

```
456 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
457 {
458   \cs_if_exist:cTF { c@ #2 }
459   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
460   {
461     \tl_set:Nn #1 { #2 }
462     \newcounter { #2 }
463   }
464 }
```

(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    465 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii   466 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii  467 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv   468 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv    469 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi   470 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii  471 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii 472 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
```

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

12.11 Definition of labels

This part of the code is inspired by the **enumitem** package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```
\__enumext_register_counter_style:Nn
```

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

```
473 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
474 {
475   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
476   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
477 }
478 \__enumext_register_counter_style:Nn \arabic { 0 }
479 \__enumext_register_counter_style:Nn \Alph { M }
480 \__enumext_register_counter_style:Nn \alph { m }
481 \__enumext_register_counter_style:Nn \Roman { VIII }
482 \__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.

```
483 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
484 {
485   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
486   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
487 }
488 \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.

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

```

490 {
491   \tl_clear_new:N #1
492   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
493   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
494   \tl_map_inline:Nn \g__enumext_counter_styles_tl
495   {
496     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
497     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
498     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
499   }
500   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
501   { \tl_use:N \g__enumext_widest_label_tl }
502   \tl_set_eq:cN { the #2 } #1
503 }
504 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

12.12 Setting keys associated with label

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

```

505 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
506 {
507   \keys_define:nn { enumext / #1 }
508   {
509     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
510     font .value_required:n = true,
511     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
512     labelsep .initial:n = {0.3333em},
513     labelsep .value_required:n = true,
514     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
515     labelwidth .value_required:n = true,
516     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
517     wrap-label .initial:n = {##1},
518     wrap-label .value_required:n = true,
519     wrap-label* .code:n = {
520       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
521       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
522     },
523     wrap-label* .value_required:n = true,
524   }
525 }
526 \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.

```

527 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
528 {
529   \keys_define:nn { enumext / #1 }
530   {
531     align .choice:,
532     align / left .code:n =
533     {
534       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
535       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
536     },
537     align / right .code:n =
538     {
539       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
540       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
541     },
542     align / center .code:n =
543     {
544       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
545       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
546     },

```

```

547         align / unknown .code:n =
548             \msg_error:nneee { enumext } { unknown-choice }
549             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
550         align .initial:n = left,
551         align .value_required:n = true,
552     }
553 }
554 \clist_map_inline:nn
555 {
556     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
557 }
558 { \__enumext_tmp:nn #1 }

559 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
560 {
561     \keys_define:nn { enumext / #1 }
562     {
563         align .choice:,
564         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
565         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
566         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
567         align / unknown .code:n =
568             \msg_error:nneee { enumext } { unknown-choice }
569             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
570         align .initial:n = left,
571         align .value_required:n = true,
572     }
573 }
574 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

12.13 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for `\label`, the value of the variables `\l__enumext_label_X_tl`, the default values for `\labelwidth` and the “*label and ref*” system.

12.13.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

575 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
576 {
577     \keys_define:nn { enumext / #1 }
578     {
579         label .code:n = {
580             \__enumext_label_style:cnv { l__enumext_label_#2_tl }
581             { l__enumext_counter_#2_tl } {##1}
582             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
583             \l__enumext_current_widest_dim
584         },
585         label .initial:n = #3,
586         label .value_required:n = true,
587         ref .code:n = \__enumext_standar_ref:n {##1},
588         ref .value_required:n = true,
589     }
590 }
591 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
592 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
593 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
594 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

The `__enumext_standar_ref:n` first we will pass the key argument to `\l__enumext_ref_key_arg_tl` and we will analyze its state, if it is not *empty* we will make a copy of the current counter in `\l__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will return the modified `\l__enumext_ref_key_arg_tl` and we make the value of `\l__enumext_ref_the_count_tl` the same as that `\l__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `\l__enumext_renew_the_count_X_tl` with the renewed command.

```

595 \cs_new_protected:Npn \__enumext_standar_ref:n #1

```



```

596 {
597   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
598   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
599   {
600     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
601   }
602   {
603     \tl_set_eq:Nc
604       \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
605     \__enumext_regex_counter_style:
606     \tl_set_eq:Nc
607       \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
608     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
609     {
610       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
611       { \exp_not:V \l__enumext_ref_key_arg_tl }
612     }
613   }
614 }

```

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

```

615 \cs_new_protected:Nn \__enumext_standar_ref:
616 {
617   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
618   {
619     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
620   }
621 }

```

(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
622 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
623 {
624   \keys_define:nn { enumext / #1 }
625   {
626     label .code:n = {
627       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
628       { \l__enumext_counter_#2_tl } {##1}
629       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
630       \l__enumext_current_widest_dim
631     },
632     label .initial:n = #3,
633     label .value_required:n = true,
634     ref .code:n = \__enumext_starred_ref:n {##1},
635     ref .value_required:n = true,
636   }
637 }
638 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
639 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

640 \cs_new_protected:Npn \__enumext_starred_ref:n #1
641 {
642   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
643   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
644   {
645     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
646     {
647       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
648     }
649     {
650       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
651       \__enumext_regex_counter_style:
652       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
653       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl

```

```

654         {
655             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
656             { \exp_not:V \l__enumext_ref_key_arg_tl }
657         }
658     }
659 }
660 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
661 {
662     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
663     {
664         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
665     }
666     {
667         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
668         \__enumext_regex_counter_style:
669         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
670         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
671         {
672             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
673             { \exp_not:V \l__enumext_ref_key_arg_tl }
674         }
675     }
676 }
677 }

```

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.

```

678 \cs_new_protected:Nn \__enumext_starred_ref:
679 {
680     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
681     {
682         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
683         {
684             \tl_use:N \l__enumext_renew_the_count_vii_tl
685         }
686     }
687     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
688     {
689         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
690         {
691             \tl_use:N \l__enumext_renew_the_count_viii_tl
692         }
693     }
694 }

```

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

12.13.3 Define and set label and ref keys for keyans and keyanspic environments

Here we set the default `<label>` for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` and `ref` key. The `keyanspic` environment use the same `<label>` as the `keyans` environment.

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
695 \keys_define:nn { enumext / keyans }
696 {
697     label .code:n = {
698         \__enumext_label_style:cvn { \__enumext_label_v_tl }
699         { \__enumext_counter_v_tl } {#1}
700         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
701         \__enumext_current_widest_dim
702         \__enumext_label_style:cvn { \__enumext_label_vi_tl }
703         { \__enumext_counter_vi_tl } {#1}
704         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
705         \__enumext_current_widest_dim
706     },
707     label .initial:n = \Alph*,
708     label .value_required:n = true,
709     ref .code:n = \__enumext_keyans_ref:n {#1},
710     ref .value_required:n = true,
711 }

```

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

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

```

712 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
713 {
714   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
715   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
716   {
717     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
718   }
719   {
720     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
721     \__enumext_regex_counter_style:
722     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
723     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
724     {
725       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
726       { \exp_not:V \l__enumext_ref_key_arg_tl }
727     }
728   }
729 }

```

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

```

730 \cs_new_protected:Npn \__enumext_keyans_ref:
731 {
732   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
733   {
734     \tl_use:N \l__enumext_renew_the_count_v_tl
735   }
736 }

```

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

12.14 Setting start, start* and widest keys

```

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

```

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

```

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

```

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

```

737 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
738 {
739   \__enumext_if_is_int:nTF { #3 }
740   {
741     \int_set:Nn #2 {#3}
742   }
743   {
744     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
745     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
746     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
747     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
748   }
749 }
750 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

751 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
752 {
753   \__enumext_if_is_int:nTF {#4}
754   {
755     \setcounter{enumX#1} { #4 }

```

```

756     }
757     {
758         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
759         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
760         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
761         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
762     }
763     \__enumext_label_width_by_box:cv
764     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
765 }
766 \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.

```

767 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
768 {
769     \keys_define:nn { enumext / #1 }
770     {
771         start* .code:n = {
772             \__enumext_start_from:ccn
773             { l__enumext_label_#2_tl }
774             { l__enumext_start_#2_int } {##1}
775         },
776         start* .value_required:n = true,
777         start .code:n = {
778             \__enumext_start_from:cce
779             { l__enumext_label_#2_tl }
780             { l__enumext_start_#2_int } { \int_eval:n {##1} }
781         },
782         start .initial:n = 1,
783         start .value_required:n = true,
784         widest .code:n = {
785             \__enumext_widest_from:nccn {#2}
786             { l__enumext_label_#2_tl }
787             { l__enumext_labelwidth_#2_dim } {##1}
788         },
789         widest .value_required:n = true,
790     }
791 }
792 \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.

```

793 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
794 {
795     \keys_define:nn { enumext / #1 }
796     {
797         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
798         topsep .initial:n = {#3},
799         topsep .value_required:n = true,
800         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
801         partopsep .initial:n = {#4},
802         partopsep .value_required:n = true,
803         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
804         parsep .initial:n = {#5},
805         parsep .value_required:n = true,
806         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
807         itemsep .initial:n = {#6},
808         itemsep .value_required:n = true,
809         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
810         noitemsep .value_forbidden:n = true,
811         nosepp .meta:n = {
812             itemsep = 0pt, parsep = 0pt,
813             topsep = 0pt, partopsep = 0pt,
814         },

```

```

815         nosep      .value_forbidden:n = true,
816     }
817 }

```

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

```

818 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
819 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
820 { 4.0pt plus 2.0pt minus 1.0pt }
821 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
822 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
823 { 2.0pt plus 1.0pt minus 1.0pt }
824 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
825 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
826 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
827 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
828 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
829 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
830 { 2.0pt plus 1.0pt minus 1.0pt }
831 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
832 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
833 { 4.0pt plus 2.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
835 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
836 { 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` or `enumext*` environments.

```

base-fix \__enumext_nested_base_line_fix:
837 \cs_set_protected:Npn \__enumext_tmp:n #1
838 {
839     \keys_define:nn { enumext / #1 }
840     {
841         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
842         base-fix .initial:n = false,
843         base-fix .value_forbidden:n = true,
844     }
845 }
846 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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)

```

847 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
848 {
849     \bool_lazy_and:nnT
850     { \bool_if_p:N \l__enumext_standar_first_bool }
851     { \bool_if_p:N \l__enumext_base_line_fix_bool }
852     {
853         \mode_leave_vertical:
854         \vspace { -\baselineskip }
855         \keys_set:nn { enumext / level-1 }
856         {
857             topsep = 0pt, above = 0pt, above* = 0pt,
858         }
859     }
860     \bool_lazy_and:nnT
861     { \bool_if_p:N \l__enumext_starred_first_bool }
862     { \bool_if_p:N \l__enumext_base_line_fix_bool }
863     {
864         \mode_leave_vertical:
865         \vspace { -\baselineskip }
866         \keys_set:nn { enumext / enumext* }
867         {
868             topsep = 0pt, above = 0pt, above* = 0pt,

```

```

869     }
870   }
871   \bool_set_false:N \__enumext_base_line_fix_bool
872 }

```

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

(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.

```

873 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
874 {
875   \keys_define:nn { enumext / #1 }
876   {
877     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
878     itemindent .value_required:n = true,
879     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
880     rightmargin .value_required:n = true,
881     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
882     listparindent .value_required:n = true,
883     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
884     list-offset .value_required:n = true,
885     list-indent .code:n =
886       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
887       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
888     list-indent .value_required:n = true,
889   }
890 }
891 \clist_map_inline:nn
892 {
893   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
894 }
895 { \__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.

```

896 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
897 {
898   \keys_define:nn { enumext / #1 }
899   {
900     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
901     itemindent .value_required:n = true,
902     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
903     rightmargin .value_required:n = true,
904     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
905     listparindent .value_required:n = true,
906     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
907     list-offset .value_required:n = true,
908     list-indent .meta:n = { list-offset = ##1 },
909     list-indent .value_required:n = true,
910   }
911 }
912 \clist_map_inline:nn
913 {
914   {enumext*}{vii}, {keyans*}{viii}
915 }
916 { \__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.

```

917 \cs_set_protected:Nn \__enumext_fake_item:
918 {
919   \dim_compare:nNt
920   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
921   >

```



```

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

```

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

12.18 Setting show-length key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to “true” and calls the function `__enumext_show_length:nnn` which prints all the values of the “vertical” and “horizontal” parameters calculated and used.

```

969 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
970 {
971   \keys_define:nn { enumext / #1 }
972   {
973     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
974     show-length .initial:n = false,
975   }
976 }
977 \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.

```

978 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
979 {
980   \keys_define:nn { enumext / #1 }
981   {
982     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
983     before .value_required:n = true,
984     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
985     before* .value_required:n = true,
986     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
987     after .value_required:n = true,
988     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
989     first .value_required:n = true,
990   }
991 }
992 \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}`.

```

993 \cs_new_protected:Nn \__enumext_before_args_exec:
994 {
995   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
996 }

```

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}`

```

997 \cs_new_protected:Nn \__enumext_before_keys_exec:
998 {
999   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1000 }

```

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

```

1001 \cs_new_protected:Nn \__enumext_after_stop_list:
1002 {
1003   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1004 }

```

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.`

```

1005 \cs_new_protected:Nn \__enumext_after_args_exec:
1006 {
1007   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1008 }

```

(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:
1009 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1010 {
1011   \tl_use:N \l__enumext_before_starred_key_v_tl
1012 }
1013 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1014 {
1015   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1016 }
1017 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1018 {
1019   \tl_use:N \l__enumext_after_stop_list_v_tl
1020 }
1021 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1022 {
1023   \tl_use:N \l__enumext_after_list_args_v_tl
1024 }

```

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

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

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

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

```
1025 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1026 {
1027   \tl_use:N \l__enumext_before_starred_key_vii_tl
1028 }
1029 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1030 {
1031   \tl_use:N \l__enumext_before_starred_key_viii_tl
1032 }
1033 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1034 {
1035   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1036 }
1037 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1038 {
1039   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1040 }
1041 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1042 {
1043   \tl_use:N \l__enumext_after_stop_list_vii_tl
1044 }
1045 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1046 {
1047   \tl_use:N \l__enumext_after_stop_list_viii_tl
1048 }
1049 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1050 {
1051   \tl_use:N \l__enumext_after_list_args_vii_tl
1052 }
1053 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1054 {
1055   \tl_use:N \l__enumext_after_list_args_viii_tl
1056 }
```

(End of definition for __enumext_before_args_exec_vii: and others.)

12.20 Setting keys for multicol and minipage

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

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

```
1057 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1058 {
1059   \keys_define:nn { enumext / #1 }
1060   {
1061     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1062     mini-env .value_required:n = true,
1063     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1064     mini-sep .initial:n = 0.3333em,
1065     mini-sep .value_required:n = true,
1066     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1067     columns-sep .value_required:n = true,
1068     columns .int_set:c = { l__enumext_columns_#2_int },
1069     columns .initial:n = 1,
1070     columns .value_required:n = true,
1071   }
1072 }
1073 \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.

```
1074 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1075 {
1076   \keys_define:nn { enumext / #1 }
1077   {
1078     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1079     mini-right .value_required:n = true,
1080     mini-right* .code:n = {
```

```

1081 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1082 \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1083 },
1084 mini-right* .value_required:n = true,
1085 }
1086 }
1087 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

12.21 Adjustment of vertical spaces for multicol

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



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

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

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

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

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

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

```

1088 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1089 {
1090   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1091   {
1092     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1093   }
1094   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1095   {
1096     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1097   }
1098   \__enumext_add_pre_parsep:
1099 }

```

(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*.

```

1100 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1101 {
1102   \int_case:nn { \l__enumext_level_int }
1103   {
1104     { 2 }{
1105       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1106       {
1107         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1108       }
1109     }
1110     { 3 }{
1111       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1112       {

```

```

1113         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1114     }
1115 }
1116 { 4 }{
1117     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1118     {
1119         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1120     }
1121     }
1122 }
1123 }

```

(End of definition for `\l__enumext_add_pre_parsep:`)

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

```

1124 \cs_new_protected:Nn \l__enumext_multi_addvspace:
1125 {
1126     \l__enumext_multi_set_vskip:
1127     \mode_if_vertical:T
1128     {
1129         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1130         {
1131             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1132         }
1133         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1134         {
1135             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1136         }
1137     }
1138     %%\l__enumext_unskip_unkern:
1139     \par\nopagebreak
1140     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1141 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

12.21.2 Adjustment of vertical spaces for multicols in keyans

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

`\l__enumext_keyans_multi_addvspace:`

```

1142 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
1143 {
1144     \skip_set:Nn \l__enumext_multicols_above_v_skip
1145     {
1146         \l__enumext_topsep_v_skip
1147     }
1148     \skip_set:Nn \l__enumext_multicols_below_v_skip
1149     {
1150         \l__enumext_topsep_v_skip
1151     }
1152 }
1153 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1154 {
1155     \l__enumext_keyans_multi_set_vskip:
1156     \mode_if_vertical:T
1157     {
1158         \skip_add:Nn \l__enumext_multicols_above_v_skip
1159         {
1160             \skip_use:N \l__enumext_partopsep_v_skip
1161         }
1162         \skip_add:Nn \l__enumext_multicols_below_v_skip
1163         {
1164             \skip_use:N \l__enumext_partopsep_v_skip
1165         }
1166     }
1167     \l__enumext_unskip_unkern:
1168     \par\nopagebreak
1169     \addvspace{ \l__enumext_multicols_above_v_skip }
1170 }

```

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

12.22 Adjustment of vertical spaces for minipage

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

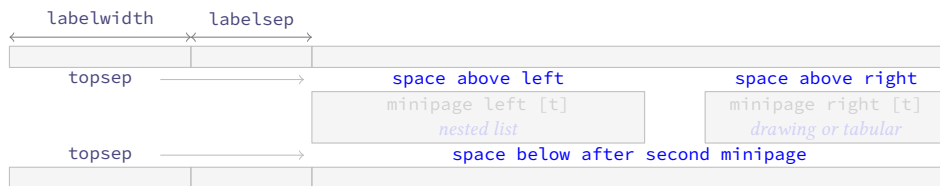


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `minipage` environment is quite complicated, even more if `multicols` is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[14] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

12.22.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

First we will set the value of `__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 `__enumext_minipage_after_skip`.

```

1171 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1172 {
1173   \skip_set:Nn \__enumext_minipage_right_skip
1174   {
1175     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1176   }
1177   \mode_if_vertical:T
1178   {
1179     \skip_add:Nn \__enumext_minipage_right_skip
1180     {
1181       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1182     }
1183   }
1184   \skip_set_eq:NN \__enumext_minipage_after_skip \__enumext_minipage_right_skip

```

We will adjust the values `__enumext_multicols_above_X_skip` and `__enumext_multicols_below_X_skip` and call the function `__enumext_pre_itemsep_skip:`.

```

1185   \skip_set_eq:cN
1186   { l__enumext_multicols_above_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1187   \skip_set_eq:cN
1188   { l__enumext_multicols_below_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1189   \__enumext_pre_itemsep_skip:

```

If the environment `multicols` is active, we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `__enumext_multicols_above_X_skip`.

```

1190   \int_compare:nNnT
1191   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1192   {
1193     \skip_zero:N \topskip
1194     \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1195   }
1196 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_mini_env*` 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*.

```

1197 \cs_new_protected:Nn \__enumext_minipage_add_space:
1198 {
1199   \__enumext_minipage_set_skip:
1200   \__enumext_unskip_unkern:
1201   \mode_if_vertical:TF
1202   {
1203     \nopagebreak\nointerlineskip
1204   }
1205   {
1206     \par\nopagebreak\nointerlineskip
1207     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1208   }
1209   \int_compare:nNnTF
1210   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1211   {
1212     \addvspace{ 0.445\box_ht:N \strutbox }
1213   }
1214   {
1215     \addvspace{ 0.250\box_ht:N \strutbox }
1216   }
1217 }

```

(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.

```

1218 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1219 {
1220   \int_case:nn { \l__enumext_level_int }
1221   {
1222     { 2 }{
1223       \skip_if_eq:nnTF
1224       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1225       {
1226         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1227         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1228       }
1229       {
1230         \dim_compare:nNnT
1231         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1232         {
1233           \skip_sub:Nn
1234           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1235           \skip_sub:Nn
1236           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1237           \skip_add:Nn
1238           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1239           \skip_add:Nn
1240           \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1241         }
1242         \dim_compare:nNnT
1243         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1244         {
1245           \skip_set:Nn \l_tmpa_skip
1246           {
1247             \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1248           }
1249           \skip_sub:Nn
1250           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1251           \skip_sub:Nn
1252           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1253           \skip_add:Nn
1254           \l__enumext_minipage_after_skip
1255           { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1256           \skip_add:Nn
1257           \l__enumext_multicols_below_ii_skip
1258           { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1259         }
1260       }
1261     }
1262   }

```

```

1260         }
1261     }
1262     { 3 }{
1263         \skip_if_eq:nnTF
1264         { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1265         {
1266             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1267             \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1268         }
1269         {
1270             \dim_compare:nnNT
1271             { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1272             {
1273                 \skip_sub:Nn
1274                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1275                 \skip_sub:Nn
1276                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1277                 \skip_add:Nn
1278                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1279                 \skip_add:Nn
1280                 \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1281             }
1282             \dim_compare:nnNT
1283             { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1284             {
1285                 \skip_set:Nn \l_tmpa_skip
1286                 {
1287                     \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1288                 }
1289                 \skip_sub:Nn
1290                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1291                 \skip_sub:Nn
1292                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1293                 \skip_add:Nn
1294                 \l__enumext_minipage_after_skip
1295                 { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1296                 \skip_add:Nn
1297                 \l__enumext_multicols_below_iii_skip
1298                 { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1299             }
1300         }
1301     }
1302     { 4 }{
1303         \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1304         {
1305             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1306             \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1307         }
1308         {
1309             \dim_compare:nnNT
1310             { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1311             {
1312                 \skip_sub:Nn
1313                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1314                 \skip_sub:Nn
1315                 \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1316                 \skip_add:Nn
1317                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1318                 \skip_add:Nn
1319                 \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1320             }
1321             \dim_compare:nnNT
1322             { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1323             {
1324                 \skip_set:Nn \l_tmpa_skip
1325                 {
1326                     \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1327                 }
1328                 \skip_sub:Nn
1329                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1330                 \skip_sub:Nn

```

```

1331         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1332     \skip_add:Nn
1333     \l__enumext_minipage_after_skip
1334     { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1335     \skip_add:Nn
1336     \l__enumext_multicols_below_iv_skip
1337     { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1338 }
1339 }
1340 }
1341 }
1342 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

12.22.2 Adjustment of vertical spaces for minipage in keyans

`__enumext_keyans_minipage_set_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_env*` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

```

1343 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1344 {
1345     \skip_zero:N \l__enumext_minipage_after_skip
1346     \skip_zero:N \l__enumext_minipage_left_skip
1347     \skip_zero:N \l__enumext_minipage_right_skip
1348     \skip_set:Nn \l__enumext_minipage_right_skip
1349     {
1350         \l__enumext_topsep_v_skip
1351     }
1352     \mode_if_vertical:T
1353     {
1354         \skip_add:Nn \l__enumext_minipage_right_skip
1355         {
1356             \l__enumext_partopsep_v_skip
1357         }
1358     }
1359     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1360     %% prev level
1361     \skip_if_eq:nnF { \l__enumext_minipage_after_skip } { \c_zero_skip }
1362     {
1363         \skip_if_eq:nnTF { \l__enumext_itemsep_i_skip } { \c_zero_skip }
1364         {
1365             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1366         }
1367         {
1368             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1369         }
1370     }
1371     %% columns
1372     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1373     {
1374         \skip_zero:N \topskip
1375         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1376     }
1377 }

```

(End of definition for `__enumext_keyans_minipage_set_skip:`)

`__enumext_keyans_minipage_add_space:`

The function `__enumext_keyans_minipage_add_space:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether TeX is in `(horizontal mode)` or `(vertical mode)`. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```

1378 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1379 {
1380     \__enumext_keyans_minipage_set_skip:
1381     \mode_if_vertical:TF
1382     {
1383         \nopagebreak\nointerlineskip
1384     }
1385     {

```

```

1386         \par\nopagebreak\nointerlineskip
1387         \skip_zero:N \l__enumext_partopsep_v_skip
1388     }
1389     \addvspace{ 0.245\box_ht:N \strutbox }
1390 }

```

(End of definition for \l__enumext_keyans_minipage_add_space:.)

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_env*` environment in `enumext*` and `keyans*`.

```

1391 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1392 {
1393     \skip_zero_new:N \l__enumext_minipage_left_skip
1394     \skip_gzero_new:N \g__enumext_minipage_right_skip
1395     \skip_gzero_new:N \g__enumext_minipage_after_skip
1396     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1397     {
1398         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1399         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1400     }
1401     {
1402         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1403         \skip_gset:Nn \g__enumext_minipage_right_skip
1404         {
1405             \l__enumext_topsep_vii_skip
1406         }
1407         \skip_gset:Nn \g__enumext_minipage_after_skip
1408         {
1409             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1410         }
1411     }
1412 }
1413 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1414 {
1415     \skip_zero_new:N \l__enumext_minipage_after_skip
1416     \skip_zero_new:N \l__enumext_minipage_left_skip
1417     \skip_zero_new:N \l__enumext_minipage_right_skip
1418     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1419     {
1420         \skip_set:Nn \l__enumext_minipage_left_skip
1421         {
1422             0.5\box_dp:N \strutbox
1423         }
1424         \skip_set:Nn \l__enumext_minipage_right_skip
1425         {
1426             \l__enumext_partopsep_viii_skip
1427         }
1428         \skip_set:Nn \l__enumext_minipage_after_skip
1429         {
1430             1.6\box_dp:N \strutbox
1431         }
1432     }
1433     {
1434         \skip_set:Nn \l__enumext_minipage_left_skip
1435         {
1436             0.5875\box_dp:N \strutbox
1437         }
1438         \skip_set:Nn \l__enumext_minipage_right_skip
1439         {
1440             \l__enumext_topsep_viii_skip
1441         }
1442         \skip_set:Nn \l__enumext_minipage_after_skip
1443         {
1444             0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1445         }
1446     }
1447 }

```

(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_env*` environment on the *left side* when the `mini-right` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether T_EX is in *(horizontal mode)* or *(vertical mode)*, since `\partopsep` is equal to `0pt` in both environments.

```

1448 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1449 {
1450   \__enumext_mini_set_vskip_vii:
1451   \par\nopagebreak
1452   \addvspace { \__enumext_minipage_left_skip }
1453 }
1454 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1455 {
1456   \__enumext_mini_set_vskip_viii:
1457   \par\nopagebreak
1458   \addvspace { \__enumext_minipage_left_skip }
1459 }

```

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

12.22.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_env*` environment on the “left side”, open the `__enumext_mini_env*` 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_env*` 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.

```

1460 \NewDocumentCommand \miniright { s }
1461 {
1462   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1463   {
1464     \msg_error:nnn { enumext } { wrong-miniright-place }
1465   }
1466   % outside
1467   \bool_lazy_and:nnT
1468   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1469   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1470   {
1471     \msg_error:nnn { enumext } { wrong-miniright-place }
1472   }
1473   % starred env
1474   \bool_if:NT \__enumext_starred_bool
1475   {
1476     \msg_error:nnn { enumext } { wrong-miniright-starred }
1477   }
1478   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1479   {
1480     \__enumext_keyans_mini_right_cmd:n {#1}
1481   }
1482   { \__enumext_mini_right_cmd:n {#1} }
1483 }

```

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

`__enumext_mini_right_cmd:n`

The function `__enumext_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `__enumext_minipage_right_X_dim`, if so we close the `\multicols` environment with the `__enumext_mini_env*` environment on the “left side”, then we open the `__enumext_mini_env*` 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.

```

1484 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1485 {
1486   \dim_compare:nNtF
1487   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1488   {
1489     \__enumext_multicols_stop:
1490     \int_compare:nNtF

```

```

1491         { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1492     {
1493         %%\skip_vertical:N \__enumext_minipage_after_skip
1494         %%\__enumext_unskip_unkern: % remove previous
1495         \par\addvspace{ \__enumext_minipage_after_skip }
1496     }
1497 \end{__enumext_mini_env*}
1498 \hfill
1499 \begin{__enumext_mini_env*}
1500 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1501 % Add vertical space above
1502 \par\nointerlineskip
1503 \addvspace { \__enumext_minipage_right_skip }
1504 \bool_if:nF {#1}
1505 {
1506     \centering
1507 }
1508 \int_gzero:N \g__enumext_minipage_stat_int
1509 }
1510 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1511 % paranoia
1512 \RenewDocumentCommand \miniright { s }
1513 {
1514     \msg_error:nn { enumext } { many-miniright-used }
1515 }
1516 }

```

(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.

```

1517 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1518 {
1519     \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1520     {
1521         \__enumext_keyans_multicols_stop:
1522         \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1523         {
1524             \skip_vertical:N \__enumext_minipage_after_skip
1525         }
1526         \end{__enumext_mini_env*}
1527         \hfill
1528         \begin{__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1529             % Add vertical space above
1530             \par\nointerlineskip
1531             \addvspace { \__enumext_minipage_right_skip }
1532             \bool_if:nF {#1}
1533             {
1534                 \centering
1535             }
1536             \int_gzero:N \g__enumext_minipage_stat_int
1537         }
1538         { \msg_error:nnn { enumext } { wrong-miniright-use } }
1539     % paranoia
1540     \RenewDocumentCommand \miniright { s }
1541     {
1542         \msg_error:nn { enumext } { many-miniright-used }
1543     }
1544 }

```

(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.

```

1545 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1546 {
1547   \keys_define:nn { enumext / #1 }
1548   {
1549     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1550     above .value_required:n = true,
1551     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1552               \keys_set:nn { enumext / #1 } { above = {##1} },
1553     above* .value_required:n = true,
1554     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1555     below .value_required:n = true,
1556     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1557               \keys_set:nn { enumext / #1 } { below = {##1} },
1558     below* .value_required:n = true,
1559   }
1560 }
1561 \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.

```

1562 \cs_new_protected:Nn \__enumext_vspace_above:
1563 {
1564   \skip_if_eq:nnF
1565   { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1566   {
1567     \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1568     {
1569       \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1570     }
1571     {
1572       \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1573     }
1574   }
1575 }

```

(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.

```

1576 \cs_new_protected:Nn \__enumext_vspace_below:
1577 {
1578   \skip_if_eq:nnF
1579   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1580   {
1581     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1582     {
1583       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1584     }
1585     {
1586       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1587     }
1588   }
1589 }

```

(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.

```

1590 \cs_new_protected:Nn \__enumext_vspace_above_v:
1591 {
1592   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1593   {
1594     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1595     {
1596       \vspace*{ \l__enumext_vspace_above_v_skip }

```

```

1597     }
1598     { \vspace { \l__enumext_vspace_above_v_skip } }
1599   }
1600 }

```

(End of definition for \l__enumext_vspace_above_v:.)

\l__enumext_vspace_below_v:

The function \l__enumext_vspace_below_v: apply the *vertical space below* the **keyans** environment set by the **below*** and **below** keys.

```

1601 \cs_new_protected:Nn \l__enumext_vspace_below_v:
1602 {
1603   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1604   {
1605     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1606     {
1607       \vspace*{ \l__enumext_vspace_below_v_skip }
1608     }
1609     { \vspace { \l__enumext_vspace_below_v_skip } }
1610   }
1611 }

```

(End of definition for \l__enumext_vspace_below_v:.)

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

\l__enumext_vspace_above_vii:

The functions \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii: apply the *vertical space above* the **enumext*** and **keyans*** environments set by the **above** and **above*** keys.

\l__enumext_vspace_above_viii:

```

1612 \cs_new_protected:Nn \l__enumext_vspace_above_vii:
1613 {
1614   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1615   {
1616     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1617     {
1618       \vspace*{ \l__enumext_vspace_above_vii_skip }
1619     }
1620     { \vspace { \l__enumext_vspace_above_vii_skip } }
1621   }
1622 }
1623 \cs_new_protected:Nn \l__enumext_vspace_above_viii:
1624 {
1625   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1626   {
1627     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1628     {
1629       \vspace*{ \l__enumext_vspace_above_viii_skip }
1630     }
1631     { \vspace { \l__enumext_vspace_above_viii_skip } }
1632   }
1633 }

```

(End of definition for \l__enumext_vspace_above_vii: and \l__enumext_vspace_above_viii:.)

\l__enumext_vspace_below_vii:

The functions \l__enumext_vspace_below_vii: and \l__enumext_vspace_below_viii: apply the *vertical space below* the **enumext*** and **keyans*** environments set by the **below*** and **below** keys.

\l__enumext_vspace_below_viii:

```

1634 \cs_new_protected:Nn \l__enumext_vspace_below_vii:
1635 {
1636   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1637   {
1638     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1639     {
1640       \vspace*{ \l__enumext_vspace_below_vii_skip }
1641     }
1642     { \vspace { \l__enumext_vspace_below_vii_skip } }
1643   }
1644 }
1645 \cs_new_protected:Nn \l__enumext_vspace_below_viii:
1646 {
1647   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1648   {
1649     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1650     {
1651       \vspace*{ \l__enumext_vspace_below_viii_skip }

```

```

1652     }
1653     { \vspace { \l__enumext_vspace_below_viii_skip } }
1654   }
1655 }

```

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

12.24 Setting series, resume and resume* keys

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

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

```

series
resume
resume*
1656 \cs_set_protected:Npn \__enumext_tmp:n #1
1657 {
1658   \keys_define:nn { enumext / #1 }
1659   {
1660     series .str_set:N = \__enumext_series_str,
1661     series .value_required:n = true,
1662     resume .code:n = \__enumext_resume_series:n {##1},
1663     resume* .code:n = \__enumext_resume_starred:,
1664     resume* .value_forbidden:n = true,
1665   }
1666 }
1667 \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

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1668 \cs_new:Npn \__enumext_filter_series:n #1
1669 {
1670   \use:e
1671   {
1672     \keyval_parse:NNn
1673     \__enumext_filter_series_key:n
1674     \__enumext_filter_series_pair:nn {#1}
1675   }
1676 }

```

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

```

1677 \cs_new:Npn \__enumext_filter_series_key:n #1
1678 {
1679   \str_case:nnF {#1}
1680   {
1681     { resume } {} { resume* } {} { base-fix } {}
1682   }
1683   { , { \exp_not:n {#1} } }
1684 }

```

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

```

1685 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1686 {
1687   \str_case:nnF {#1}
1688   {
1689     { series } {} { resume } {} { start } {}
1690     { start* } {} { save-ans } {} { save-key } {}
1691   }
1692   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1693 }

```

(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).

```

1694 \cs_new_protected:Npn __enumext_parse_series:n #1
1695 {
1696   \str_if_empty:NTF \l__enumext_series_str
1697   {
1698     \bool_if:NF \l__enumext_resume_active_bool
1699     {
1700       __enumext_resume_last:n {#1}
1701     }
1702   }
1703   {
1704     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1705     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1706       { __enumext_filter_series:n {#1} }
1707     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1708     {
1709       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1710     }
1711   }
1712 }

```

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.

```

1713 \cs_new_protected:Npn __enumext_resume_last:n #1
1714 {
1715   \bool_if:NT \l__enumext_standar_first_bool
1716   {
1717     \tl_gclear:N \g__enumext_standar_series_tl
1718     \tl_gset:Ne \g__enumext_standar_series_tl { __enumext_filter_series:n {#1} }
1719   }
1720   \bool_if:NT \l__enumext_starred_first_bool
1721   {
1722     \tl_gclear:N \g__enumext_starred_series_tl
1723     \tl_gset:Ne \g__enumext_starred_series_tl { __enumext_filter_series:n {#1} }
1724   }
1725 }

```

(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).

```

1726 \cs_new_protected:Nn __enumext_resume_save_counter:
1727 {
1728   \bool_if:NT \g__enumext_standar_bool
1729   {
1730     \tl_if_empty:NF \l__enumext_series_str
1731     {
1732       \int_gset_eq:cN

```

```

1733         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1734     }
1735     \tl_if_empty:NTF \l__enumext_resume_name_tl
1736     {
1737         \str_if_empty:NT \l__enumext_series_str
1738         {
1739             \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1740         }
1741     }
1742     {
1743         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1744         {
1745             \int_gset_eq:cN
1746             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1747         }
1748     }
1749     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1750     {
1751         \int_gset_eq:cN
1752         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1753     }
1754 }
1755 \bool_if:NT \g__enumext_starred_bool
1756 {
1757     \tl_if_empty:NF \l__enumext_series_str
1758     {
1759         \int_gset_eq:cN
1760         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1761     }
1762     \tl_if_empty:NTF \l__enumext_resume_name_tl
1763     {
1764         \str_if_empty:NT \l__enumext_series_str
1765         {
1766             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1767         }
1768     }
1769     {
1770         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1771         {
1772             \int_gset_eq:cN
1773             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1774         }
1775     }
1776     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1777     {
1778         \int_gset_eq:cN
1779         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1780     }
1781 }
1782 }

```

(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.

```

1783 \cs_new_protected:Npn \__enumext_resume_series:n #1
1784 {
1785     \tl_if_empty:NTF {#1}
1786     {
1787         \__enumext_resume_counter:n { }
1788     }
1789     {
1790         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1791         {
1792             \__enumext_resume_counter:n {#1}

```

```

1793         \bool_if:NT \g__enumext_standar_bool
1794         {
1795             \keys_set:nv { enumext / level-1 }
1796             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1797         }
1798         \bool_if:NT \g__enumext_starred_bool
1799         {
1800             \keys_set:nv { enumext / enumext* }
1801             { g__enumext_series_ \tl_to_str:n {#1} _tl }
1802         }
1803     }
1804     {
1805         \bool_if:NT \g__enumext_standar_bool
1806         {
1807             \msg_error:nnn { enumext } { unknown-series } {#1}
1808         }
1809         \bool_if:NT \g__enumext_starred_bool
1810         {
1811             \msg_error:nnn { enumext } { unknown-series } {#1}
1812         }
1813     }
1814 }
1815 }

```

(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`.

```

1816 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1817 {
1818     \bool_set_true:N \l__enumext_resume_active_bool
1819     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1820     \tl_if_empty:NTF \l__enumext_resume_name_tl
1821     {
1822         \__enumext_resume_counter:
1823     }
1824     {
1825         \__enumext_resume_counter_series:
1826     }
1827     \__enumext_resume_counter_save_ans:
1828 }

```

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.

```

1829 \cs_new_protected:Nn \__enumext_resume_counter:
1830 {
1831     \bool_if:NT \g__enumext_standar_bool
1832     {
1833         \int_gincr:N \g__enumext_resume_int
1834         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1835     }
1836     \bool_if:NT \g__enumext_starred_bool
1837     {
1838         \int_gincr:N \g__enumext_resume_vii_int
1839         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1840     }
1841 }

```

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.

```

1842 \cs_new_protected:Nn \__enumext_resume_counter_series:
1843 {
1844     \bool_if:NT \g__enumext_standar_bool
1845     {
1846         \int_set:Nn \l__enumext_start_i_int

```

```

1847         {
1848             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1849         }
1850     }
1851     \bool_if:NT \g__enumext_starred_bool
1852     {
1853         \int_set:Nn \l__enumext_start_vii_int
1854         {
1855             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1856         }
1857     }
1858 }

```

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.

```

1859 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1860 {
1861     \bool_lazy_and:nnT
1862     { \bool_if_p:N \l__enumext_standar_first_bool }
1863     { \bool_if_p:N \l__enumext_store_active_bool }
1864     {
1865         \int_set:Nn \l__enumext_start_i_int
1866         {
1867             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1868         }
1869     }
1870     \bool_lazy_and:nnT
1871     { \bool_if_p:N \l__enumext_starred_first_bool }
1872     { \bool_if_p:N \l__enumext_store_active_bool }
1873     {
1874         \int_set:Nn \l__enumext_start_vii_int
1875         {
1876             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1877         }
1878     }
1879 }

```

(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.

```

1880 \cs_new_protected:Nn \__enumext_resume_starred:
1881 {
1882     \bool_if:NT \g__enumext_standar_bool
1883     {
1884         \tl_if_empty:NF \g__enumext_standar_series_tl
1885         {
1886             \__enumext_resume_counter:n { }
1887             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1888         }
1889     }
1890     \bool_if:NT \g__enumext_starred_bool
1891     {
1892         \tl_if_empty:NF \g__enumext_starred_series_tl
1893         {
1894             \__enumext_resume_counter:n { }
1895             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1896         }
1897     }
1898 }

```

(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*`.

```

1899 \cs_set_protected:Npn \__enumext_tmp:n #1
1900 {
1901   \keys_define:nn { enumext / #1 }
1902   {
1903     save-ans .code:n = \__enumext_storing_set:n {##1},
1904     save-ans .value_required:n = true,
1905   }
1906 }
1907 \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:` 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:`.

`__enumext_stop_save_ans_msg:`

```

1908 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1909 {
1910   \msg_term:nnVV { enumext } { save-ans-log }
1911   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1912 }
1913 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1914 {
1915   \msg_term:nnVV { enumext } { save-ans-log-hook }
1916   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1917 }

```

(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.

```

1918 \cs_new_protected:Npn \__enumext_storing_set:n #1
1919 {
1920   \tl_set:Nx \l__enumext_store_name_tl {#1}
1921   \tl_if_empty:NTF \l__enumext_store_name_tl
1922   {
1923     \bool_lazy_or:nnT
1924     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1925     {
1926       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1927     }
1928   }
1929   {
1930     \bool_lazy_or:nnT
1931     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1932     {
1933       \__enumext_start_save_ans_msg:
1934       \__enumext_storing_exec:
1935     }
1936   }
1937 }

```

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 `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy {*⟨store name⟩*} into the global variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§12.30). 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*`.

```

1938 \cs_new_protected:Nn \__enumext_storing_exec:
1939 {

```

```

1940 \bool_set_true:N \l__enumext_store_active_bool
1941 \bool_set_true:N \l__enumext_check_answers_bool
1942 \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1943 \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1944 \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1945 {
1946   \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1947   \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1948 }
1949 \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1950 {
1951   \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1952   \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1953 }
1954 \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1955 {
1956   \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1957   \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1958 }
1959 }

```

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

12.25.3 The check answer mechanism

The mechanism for checking that all questions are answered follows this logic:

If the line begins with `\item` or `\item*` and does NOT *open a nested environment*, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

If the line begins with `\item` or `\item*` and *opens a nested environment* each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must decrementing by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a “hook” function after closing the first level of the environment.

12.25.4 Setting check-ans and no-store keys

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

check-ans

no-store

```

1960 \cs_set_protected:Npn \__enumext_tmp:n #1
1961 {
1962   \keys_define:nn { enumext / #1 }
1963   {
1964     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1965     check-ans .initial:n = false,
1966     check-ans .value_required:n = true,
1967     no-store .code:n = {
1968       \bool_set_false:N \l__enumext_check_answers_bool
1969       \bool_set_false:N \l__enumext_check_ans_key_bool
1970     },
1971     no-store .value_forbidden:n = true,
1972   }
1973 }

```

```

1974 \clist_map_inline:nn
1975 {
1976     level-1, level-2, level-3, level-4, enumext*
1977 }
1978 { \__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.

```

1979 \cs_new_protected:Nn \__enumext_check_ans_active:
1980 {
1981     \tl_if_empty:NF \l__enumext_store_name_tl
1982     {
1983         \bool_if:NT \l__enumext_check_answers_bool
1984         {
1985             \__enumext_check_ans_level:
1986         }
1987     }
1988 }

```

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*”.

```

1989 \cs_new_protected:Nn \__enumext_check_ans_level:
1990 {
1991     \int_case:nn { \l__enumext_level_int }
1992     {
1993         { 1 }{
1994             \bool_lazy_all:nT
1995             {
1996                 { \bool_if_p:N \g__enumext_starred_bool }
1997                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1998             }
1999             {
2000                 \int_gdecr:N \g__enumext_item_number_int
2001                 \bool_set_false:N \l__enumext_item_number_bool
2002             }
2003         }
2004         { 2 }{
2005             \int_gdecr:N \g__enumext_item_number_int
2006             \bool_set_false:N \l__enumext_item_number_bool
2007         }
2008         { 3 }{
2009             \int_gdecr:N \g__enumext_item_number_int
2010             \bool_set_false:N \l__enumext_item_number_bool
2011         }
2012         { 4 }{
2013             \int_gdecr:N \g__enumext_item_number_int
2014             \bool_set_false:N \l__enumext_item_number_bool
2015         }
2016     }

```

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.

```

2017     \int_case:nn { \l__enumext_level_h_int }
2018     {
2019         { 1 }{
2020             \bool_lazy_all:nT
2021             {
2022                 { \bool_if_p:N \g__enumext_standar_bool }
2023                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2024             }
2025             {
2026                 \int_gdecr:N \g__enumext_item_number_int
2027                 \bool_set_false:N \l__enumext_item_number_bool
2028             }

```

```

2029         }
2030     }
2031 }

```

(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.

```

2032 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2033 {
2034     \bool_lazy_and:nnT
2035     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2036     { \bool_if_p:N \g__enumext_standar_bool }
2037     {
2038         \bool_gset_true:N \g__enumext_check_ans_key_bool
2039     }
2040     \bool_lazy_and:nnT
2041     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2042     { \bool_if_p:N \g__enumext_starred_bool }
2043     {
2044         \bool_gset_true:N \g__enumext_check_ans_key_bool
2045     }
2046 }

```

(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:`.

```

2047 \cs_new_protected:Nn \__enumext_item_answer_diff:
2048 {
2049     \int_gset:Nn \g__enumext_item_answer_diff_int
2050     {
2051         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2052     }
2053 }

```

(End of definition for `__enumext_item_answer_diff:`)

`__enumext_check_ans_show:`

`__enumext_check_ans_msg_less:`

`__enumext_check_ans_msg_same_ok:`

`__enumext_check_ans_msg_greater:`

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

```

2054 \cs_new_protected:Nn \__enumext_check_ans_show:
2055 {
2056     \int_case:nn { \g__enumext_item_answer_diff_int }
2057     {
2058         { -1 } { \__enumext_check_ans_msg_less: }
2059         { 0 } { \__enumext_check_ans_msg_same_ok: }
2060         { 1 } { \__enumext_check_ans_msg_greater: }
2061     }
2062 }
2063 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2064 {
2065     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2066     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2067 }
2068 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2069 {
2070     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2071     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2072 }
2073 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2074 {
2075     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2076     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2077 }

```

(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:`.

```

2078 \cs_new_protected:Nn \__enumext_check_ans_log:
2079 {
2080   \int_case:nn { \g__enumext_item_answer_diff_int }
2081   {
2082     { -1 } { \__enumext_check_ans_log_msg_less: }
2083     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2084     { 1 } { \__enumext_check_ans_log_msg_greater: }
2085   }
2086 }
2087 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2088 {
2089   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2090   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2091 }
2092 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2093 {
2094   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2095   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2096 }
2097 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2098 {
2099   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2100   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2101 }

```

(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.

```

2102 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2103 {
2104   \int_compare:nNnT
2105   { \g__enumext_check_starred_cmd_int } = { 0 }
2106   {
2107     \msg_warning:nnnV
2108     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2109   }
2110   \int_compare:nNnT
2111   { \g__enumext_check_starred_cmd_int } > { 1 }
2112   {
2113     \msg_warning:nnnV
2114     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2115   }
2116   \int_gzero:N \g__enumext_check_starred_cmd_int
2117   \tl_clear:N \l__enumext_check_start_line_env_tl
2118 }

```

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

12.26 Keys and functions associated with storage

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

```

2119 \cs_set_protected:Npn \__enumext_tmp:n #1
2120 {
2121   \keys_define:nn { enumext / #1 }
2122   {
2123     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2124     wrap-ans .initial:n =
2125       {
2126         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2127       },
2128     wrap-ans .value_required:n = true,
2129     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,

```

```

2130     wrap-opt .initial:n = [{##1}],
2131     wrap-opt .value_required:n = true,
2132     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2133     save-sep .initial:n = {, ~ },
2134     save-sep .value_required:n = true,
2135     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2136     mark-ans .initial:n = \textasteriskcentered,
2137     mark-ans .value_required:n = true,
2138     mark-pos .choice:,
2139     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2140     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2141     mark-pos / unknown .code:n =
2142         \msg_error:nnee { enumext } { unknown-choice }
2143         { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2144     mark-pos .initial:n = right,
2145     mark-pos .value_required:n = true,
2146     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2147     show-ans .initial:n = false,
2148     show-ans .value_required:n = true,
2149     show-pos .bool_set:N = \l__enumext_show_position_bool,
2150     show-pos .initial:n = false,
2151     show-pos .value_required:n = true,
2152     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2153     mark-ref .initial:n = \textasteriskcentered,
2154     mark-ref .value_required:n = true,
2155     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2156     save-ref .initial:n = false,
2157     save-ref .value_required:n = true,
2158 }
2159 }
2160 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {##1} }

```

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

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

```

mark-pos 2161 \cs_set_protected:Npn \l__enumext_tmp:n #1
show-ans 2162 {
show-pos 2163     \keys_define:nn { enumext / #1 }
2164     {
2165         mark-pos .choice:,
2166         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2167         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2168         mark-pos .initial:n = right,
2169         mark-pos .value_required:n = true,
2170         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2171         show-ans .initial:n = false,
2172         show-ans .value_required:n = true,
2173         show-pos .bool_set:N = \l__enumext_show_position_bool,
2174         show-pos .initial:n = false,
2175         show-pos .value_required:n = true,
2176     }
2177 }
2178 \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*” 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 arguments passed to the levels of the environment in which it is executed and “*storing*” them.

```

\__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 “*storing*” the *keys* filtered from the optional arguments 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*.

```

2179 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2180 {
2181     \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2182     {

```

```

2183         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2184         \tl_set:ce
2185             { l__enumext_store_save_key_ \__enumext_level: _tl }
2186             { \__enumext_filter_save_key:n {#1} }
2187     }
2188 }
2189 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2190 {
2191     \bool_if:NF \__enumext_store_save_key_vii_bool
2192     {
2193         \tl_clear:N \__enumext_store_save_key_vii_tl
2194         \tl_set:Nc \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2195     }
2196 }

```

(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 list structure will be stored in the *sequence* established by the `save-ans` key when executing `\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 list stored in the *sequence*.

`save-key` The values set by this key passed in the optional arguments 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`.

Define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2197 \cs_set_protected:Npn \__enumext_tmp:n #1
2198 {
2199     \keys_define:nn { enumext / enumext* }
2200     {
2201         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2202         save-key .value_required:n = true,
2203     }
2204     \keys_define:nn { enumext / #1 }
2205     {
2206         save-key .code:n = \__enumext_parse_save_key:n {##1},
2207         save-key .value_required:n = true,
2208     }
2209 }
2210 \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 the filtered *keys* in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2211 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2212 {
2213     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2214     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2215     \tl_set:ce
2216         { l__enumext_store_save_key_ \__enumext_level: _tl }
2217         { \__enumext_filter_save_key:n {#1} }
2218 }
2219 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2220 {
2221     \bool_set_true:N \__enumext_store_save_key_vii_bool
2222     \tl_clear:N \__enumext_store_save_key_vii_tl
2223     \tl_set:Nc \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2224 }

```

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

```

\__enumext_filter_save_key:n
\__enumext_filter_save_key_key:n
\__enumext_filter_save_key_pair:nn
2225 \cs_new:Npn \__enumext_filter_save_key:n #1
2226 {
2227     \use:e
2228     {

```



```

2229         \keyval_parse:NNn
2230         \__enumext_filter_save_key_key:n
2231         \__enumext_filter_save_key_pair:nn {#1}
2232     }
2233 }

```

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

```

2234 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2235 {
2236     \str_case:nnF {#1}
2237     {
2238         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2239     }
2240     { , { \exp_not:n {#1} } }
2241 }

```

The function `__enumext_filter_save_key_pair:nn` will be responsible for filtering the *⟨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.

```

2242 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2243 {
2244     \str_case:nnF {#1}
2245     {
2246         { series } {} { resume } {} { save-ans } {} { save-ref } {}
2247         { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2248         { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2249         { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2250         { mini-right* } {}
2251     }
2252     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2253 }

```

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

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

```

2254 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2255 {
2256     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2257     {
2258         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2259     }
2260     { #1 }
2261 }
2262 \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 content in *⟨sequence⟩* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

The form in which the content is stored in *⟨sequence⟩* is in a internal `enumext` or `enumext*` environments with the *same structure* in which the command was executed.

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

```

2263 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2264 {
2265     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2266 }
2267 \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

The memorization structure of the list is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```

2268 \cs_new_protected:Nn \__enumext_store_level_open:
2269 {
2270   \bool_if:NT \l__enumext_check_answers_bool
2271   {
2272     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2273     {
2274       \__enumext_store_addto_seq:n
2275       {
2276         \item \begin{enumext}
2277       }
2278     }
2279     {
2280       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2281       {
2282         \item \begin{enumext} [
2283         ]
2284       }
2285       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2286       {
2287         ]
2288       }
2289       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2290     }
2291   }
2292 \cs_new_protected:Nn \__enumext_store_level_close:
2293 {
2294   \bool_if:NT \l__enumext_check_answers_bool
2295   {
2296     \__enumext_store_addto_seq:n { \end{enumext} }
2297   }
2298 }

```

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

The memorization structure of the list is handled by the functions `__enumext_store_level_open_vii:` and `__enumext_store_level_close_vii:` which are executed in the `enumext*` environment.

```

2299 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2300 {
2301   \bool_if:NT \l__enumext_check_answers_bool
2302   {
2303     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2304     {
2305       \__enumext_store_addto_seq:n
2306       {
2307         \item \begin{enumext*}
2308       }
2309     }
2310     {
2311       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2312       {
2313         \item \begin{enumext*}[
2314         ]
2315       }
2316       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2317       {
2318         ]
2319       }
2320       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2321     }
2322   }
2323 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2324 {
2325   \bool_if:NT \l__enumext_check_answers_bool
2326   {
2327     \__enumext_store_addto_seq:n { \end{enumext*} }
2328   }
2329 }

```

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

12.26.7 Function for show marks and position

`__enumext_print_keyans_box:NN`
`__enumext_print_keyans_box:cc`

The function `__enumext_print_keyans_box:NN` print a box in the left margin with `\l__enumext_mark_answer_sym_tl` used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

#1: `\l__enumext_labelwidth_X_dim`
 #2: `\l__enumext_labelsep_X_dim`

```
2330 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2331 {
2332   \mode_leave_vertical:
2333   \skip_horizontal:n { -\dim_use:N #2 }
2334   \makebox[0pt][ r ]
2335   {
2336     \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2337     {
2338       \tl_use:N \l__enumext_mark_answer_sym_tl
2339     }
2340   }
2341   \skip_horizontal:n { \dim_use:N #2 }
2342 }
2343 \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`.

```
2344 \cs_new_protected:Nn \__enumext_store_internal_ref:
2345 {
2346   \cs_set_protected:Npn \__enumext_tmp:n ##1
2347   {
2348     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2349     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2350     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2351     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2352   }
2353   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2354   \cs_set:Npn \__enumext_tmp:n ##1
2355   { . \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.

```
2356 \bool_lazy_all:nT
2357 {
2358   { \bool_if_p:N \g__enumext_starred_bool }
2359   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2360 }
2361 {
2362   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2363   { \tl_use:N \l__enumext_label_copy_vii_tl }
2364 }
2365 \bool_lazy_all:nT
2366 {
2367   { \bool_not_p:n { \g__enumext_standar_bool } }
2368   { \bool_if_p:N \l__enumext_standar_bool }
2369   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2370 }
2371 {
2372   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2373   {
2374     \tl_use:N \l__enumext_label_copy_vii_tl
2375     \int_step_function:nnN { 1 } { { \l__enumext_level_int } \__enumext_tmp:n
2376   }
2377 }
```

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.

```

2378 \bool_lazy_all:nT
2379 {
2380   { \bool_if_p:N \g__enumext_standar_bool }
2381   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2382   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2383 }
2384 {
2385   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2386   {
2387     \tl_use:N \l__enumext_label_copy_i_tl
2388     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2389   }
2390 }
2391 \cs_set:Npn \l__enumext_tmp:n ##1
2392 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2393 \bool_lazy_all:nT
2394 {
2395   { \bool_if_p:N \g__enumext_standar_bool }
2396   { \bool_if_p:N \l__enumext_starred_bool }
2397   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2398 }
2399 {
2400   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2401   {
2402     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2403     \tl_use:N \l__enumext_label_copy_vii_tl
2404   }
2405 }

```

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

```

2406 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2407 {
2408   \l__enumext_store_name_tl \c_colon_str
2409   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2410 }

```

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

```

2411 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2412 {
2413   \l__enumext_newlabel:nn
2414   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2415   { \l__enumext_newlabel_arg_two_tl }
2416 }
2417 \l__enumext_write_aux_file_tl
2418 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

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

`\l__enumext_store_anskey_code:n`

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

```

2419 \cs_new_protected:Npn \l__enumext_store_anskey_code:n #1
2420 {
2421   \int_gincr:N \g__enumext_item_anskey_int
2422   \l__enumext_store_addto_prop:n {#1}
2423   \bool_if:NT \l__enumext_store_ref_key_bool
2424   {
2425     \l__enumext_store_internal_ref:
2426   }
2427   \l__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 $\langle \textit{sequence} \rangle$. 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`.

```

2428 \tl_clear:N \l__enumext_store_anskey_arg_tl

```

```

2429 \bool_lazy_and:nnT
2430 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2431 { \bool_not_p:n { \l__enumext_starred_bool } }
2432 {
2433   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2434 }
2435 \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 (*number*) to `\l__enumext_store_anskey_arg_tl`.

```

2436 \bool_lazy_and:nnT
2437 { \bool_not_p:n { \l__enumext_starred_bool } }
2438 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2439 {
2440   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2441   {
2442     ( \exp_not:V \l__enumext_store_item_join_int )
2443   }
2444 }

```

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 (*argument*) for `\anskey` or (*body*) for `anskey*`.

```

2445 \bool_if:NTF \l__enumext_store_item_star_bool
2446 {
2447   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2448   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2449   {
2450     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2451     {
2452       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2453     }
2454   }
2455   \dim_compare:nT
2456   {
2457     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2458   }
2459   {
2460     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2461     {
2462       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2463     }
2464   }
2465   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2466 }
2467 {
2468   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2469 }

```

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*).

```

2470 \bool_lazy_and:nnT
2471 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2472 { \bool_if_p:N \l__enumext_hyperref_bool }
2473 {
2474   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2475   {
2476     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2477     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2478   }
2479 }
2480 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2481 }

```

(End of definition for `__enumext_store_anskey_code:n`)

`__enumext_anskey_show_wrap_arg:n`

The function `__enumext_anskey_show_wrap_arg:n` “wraps” the (*argument*) passed to `\anskey` and the (*body*) for `anskey*` when using the `wrap-ans` key.

```

2482 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2483 {
2484   \par
2485   \bool_if:NTF \l__enumext_starred_bool
2486   {

```

```

2487     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_d
2488   }
2489   {
2490     \__enumext_print_keyans_box:cc
2491     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2492     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2493   }
2494   \__enumext_anskey_wrapper:n { #1 }
2495 }

```

(End of definition for __enumext_anskey_show_wrap_arg:n.)

__enumext_anskey_show_wrap_left:n

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

```

2496 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2497 {
2498   \bool_if:NT \l__enumext_show_answer_bool
2499   {
2500     \__enumext_anskey_show_wrap_arg:n { #1 }
2501   }
2502   \bool_if:NT \l__enumext_show_position_bool
2503   {
2504     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2505     {
2506       \group_begin:
2507       \exp_not:N \normalfont
2508       \exp_not:N \footnotesize [ \int_eval:n
2509       {
2510         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2511       }
2512       ]
2513       \group_end:
2514     }
2515     \__enumext_anskey_show_wrap_arg:n { #1 }
2516   }
2517 }

```

(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>}`.

__enumext_anskey_unknown:n
 __enumext_anskey_unknown:n

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

```

2518 \keys_define:nn { enumext / anskey }
2519 {
2520   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2521   break-col .default:n = true,
2522   break-col .value_forbidden:n = true,
2523   item-join .int_set:N = \l__enumext_store_item_join_int,
2524   item-join .value_required:n = true,
2525   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2526   item-star .default:n = true,
2527   item-star .value_forbidden:n = true,
2528   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2529   item-sym* .value_required:n = true,
2530   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2531   item-pos* .value_required:n = true,
2532   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2533 }

```

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`.

```

2534 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2535 {
2536   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2537 }

```

```

2538 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2539 {
2540   \tl_if_blank:nTF {#2}
2541   {
2542     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2543   }
2544   {
2545     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2546   }
2547 }

```

(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`.

```

2548 \NewDocumentCommand \anskey { o +m }
2549 {
2550   \__enumext_anskey_safe_outer:
2551   \group_begin:
2552     \bool_if:NT \l__enumext_check_answers_bool
2553     {
2554       \tl_if_novalue:nF {#1}
2555       {
2556         \keys_set:nn { enumext / anskey } {#1}
2557       }
2558       \tl_if_blank:nTF {#2}
2559       {
2560         \msg_error:nn { enumext } { anskey-empty-arg }
2561       }
2562       {
2563         \__enumext_anskey_safe_inner:
2564         \__enumext_store_anskey_code:n {#2}
2565       }
2566     }
2567   \group_end:
2568 }

```

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

12.29.1 Internal functions for the command

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

`__enumext_anskey_safe_inner:`

```

2569 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2570 {
2571   \bool_if:NF \l__enumext_store_active_bool
2572   {
2573     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2574   }
2575   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2576   {
2577     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2578   }
2579   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2580   {
2581     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2582   }
2583   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2584   {
2585     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2586   }
2587 }

```

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.

```

2588 \cs_new_protected:Nn \__enumext_anskey_safe_inner:

```



```

2589 {
2590   \int_incr:N \l__enumext_anskey_level_int
2591   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2592   {
2593     \msg_error:nn { enumext } { anskey-nested }
2594   }
2595   \bool_if:NF \l__enumext_item_number_bool
2596   {
2597     \msg_error:nn { enumext } { anskey-unnumber-item }
2598   }
2599   \mode_if_math:T
2600   {
2601     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2602   }
2603 }

```

(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.

```

2604 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2605 {
2606   \cs_undefine:c { anskey* }
2607   \cs_undefine:c { endanskey* }
2608   \cs_undefine:c { __scontents_anskey*_env_begin: }
2609   \cs_undefine:c { __scontents_anskey*_env_end: }
2610 }

```

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

```

2611 \__enumext_before_env:nn { enumext }
2612 {
2613   \bool_lazy_and:nnT
2614   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2615   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2616   {
2617     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2618     {
2619       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2620     }
2621   }
2622 }
2623 \__enumext_before_env:nn { enumext* }
2624 {
2625   \bool_lazy_and:nnT
2626   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2627   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2628   {
2629     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2630     {
2631       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2632     }
2633   }
2634 }

```

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.

```

2635 \__enumext_before_env:nn { anskey* }
2636 {
2637   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2638   {
2639     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2640   }
2641   \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2642   {
2643     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2644   }
2645   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2646   {
2647     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2648   }
2649   \bool_if:NF \__enumext_item_number_bool
2650   {
2651     \msg_error:nn { enumext } { anskey-unnumber-item }
2652   }
2653   \mode_if_math:T
2654   {
2655     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2656   }
2657 }

```

(End of definition for `__enumext_undefine_anskey_env:.`)

`anskey*`

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

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec: ($12.25.1)` and we will execute it only if the variable `__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2658 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2659 {
2660   \bool_if:NT \__enumext_anskey_env_bool
2661   {
2662     \newenvsc{anskey*}[store-env=#1,print-env=false]
2663     \__enumext_anskey_env_exec:
2664   }
2665 }
2666 \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`.

```

2667 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2668 {
2669   \keys_define:nn { scontents / scontents }
2670   {
2671     break-col .bool_gset:N = \__enumext_store_columns_break_bool,
2672     break-col .default:n   = true,
2673     break-col .value_forbidden:n = true,
2674     item-join .int_gset:N   = \__enumext_store_item_join_int,
2675     item-join .value_required:n = true,
2676     item-star .bool_gset:N = \__enumext_store_item_star_bool,
2677     item-star .default:n   = true,
2678     item-star .value_forbidden:n = true,
2679     item-sym* .tl_gset:N   = \__enumext_store_item_symbol_tl,
2680     item-sym* .value_required:n = true,
2681     item-pos* .dim_gset:N   = \__enumext_store_item_symbol_sep_dim,
2682     item-pos* .value_required:n = true,
2683     print-env .undefine:,
2684     store-env .undefine:,
2685     write-out .undefine:,
2686     unknown .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2687   }
2688 }

```

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`.

```

2689 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2690 {
2691   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2692 }
2693 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2694 {
2695   \tl_if_blank:nTF {#2}
2696   {
2697     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2698   }
2699   {
2700     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2701   }
2702 }

```

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`.

```

2703 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2704 {
2705   \keys_define:nn { scontents / scontents }
2706   {
2707     break-col .undefine:,
2708     item-join .undefine:,
2709     item-star .undefine:,
2710     item-sym* .undefine:,
2711     item-pos* .undefine:,
2712     write-out .code:n = {
2713       \bool_set_false:N \l__scontents_storing_bool
2714       \bool_set_true:N \l__scontents_writing_bool
2715       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2716     },
2717     write-out .value_required:n = true,
2718     print-env .meta:nn = { scontents } { print-env = ##1 },
2719     print-env .default:n = true,
2720     store-env .meta:nn = { scontents } { store-env = ##1 },
2721     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2722   }
2723 }

```

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*.

```

2724 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2725 {
2726   \group_begin:
2727   \int_set:Nn \tex_newlinechar:D { ``^^J }
2728   \__scontents_rescan_tokens:x
2729   {
2730     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2731     #1
2732   }
2733 }

```

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

`__enumext_anskey_env_exec:`

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

```

2734 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2735 {
2736   \__enumext_before_env:nn { anskey* }
2737   {
2738     \__enumext_anskey_env_define_keys:
2739   }

```

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.

```

2740   \hook_if_empty:nF {env/anskey*/after}

```

```

2741     {
2742         \hook_gremove_code:nn {env/anskey*/after} { * }
2743     }
2744     \__enumext_after_env:nn { anskey* }
2745     {
2746         \__enumext_anskey_env_save_keys:
2747         \tl_clear:N \l__enumext_store_anskey_env_tl
2748         \tl_clear:N \l__enumext_store_anskey_opt_tl
2749         \bool_if:NT \l__enumext_check_answers_bool
2750         {
2751             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2752             {
2753                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2754             }
2755             \regex_match:nVTF
2756             { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2757             \l__enumext_store_anskey_env_tl
2758             {
2759                 \msg_error:nn { enumext } { anskey-empty-arg }
2760             }
2761             {
2762                 \__enumext_anskey_env_store:
2763             }
2764         }
2765         \__enumext_anskey_env_clean_vars:
2766         \__enumext_anskey_env_reset_keys:
2767     }
2768 }

```

• 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`.

```

2769 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2770 {
2771     \bool_lazy_and:nnT
2772     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2773     { \bool_not_p:n { \l__enumext_starred_bool } }
2774     {
2775         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2776     }

```

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

```

2777     \bool_lazy_and:nnT
2778     { \bool_not_p:n { \l__enumext_starred_bool } }
2779     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2780     {
2781         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2782         {
2783             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2784         }
2785     }

```

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

```

2786     \bool_if:NT \g__enumext_store_item_star_bool
2787     {
2788         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2789         {
2790             ,item-star,
2791         }
2792         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2793         {
2794             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2795             {
2796                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,

```

```

2797         }
2798     }
2799     \dim_compare:nT
2800     {
2801         \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2802     }
2803     {
2804         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2805         {
2806             ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2807         }
2808     }
2809 }
2810 }

```

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`.

```

2811 \cs_new_protected:Nn \__enumext_anskey_env_store:
2812 {
2813     \group_begin:
2814     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2815     {
2816         \exp_args:Ne
2817         \__enumext_store_anskey_code:n
2818         {
2819             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2820         }
2821     }
2822     {
2823         \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2824         \exp_args:Ne
2825         \__enumext_store_anskey_code:n
2826         {
2827             \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2828         }
2829     }
2830     \group_end:
2831 }

```

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

```

2832 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2833 {
2834     \bool_gset_false:N \g__enumext_store_columns_break_bool
2835     \int_gzero:N \g__enumext_store_item_join_int
2836     \bool_gset_false:N \g__enumext_store_item_star_bool
2837     \tl_gclear:N \g__enumext_store_item_symbol_tl
2838     \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2839 }

```

(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.

```

2840 \cs_new_protected:Nn \__enumext_execute_after_env:
2841 {
2842     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2843     {
2844         \tl_if_empty:NF \g__enumext_store_name_tl
2845         {
2846             \__enumext_stop_save_ans_msg:
2847             \__enumext_item_answer_diff:
2848             \__enumext_log_global_vars:

```

```

2849         \__enumext_log_answer_vars:
2850         \bool_if:NTF \__enumext_check_ans_key_bool
2851         {
2852             \__enumext_check_ans_show:
2853         }
2854         { \__enumext_check_ans_log: }
2855         \__enumext_undefine_anskey_env:
2856     }
2857     \__enumext_reset_global_vars:
2858 }
2859 }

```

(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 contents of the current *⟨label⟩* \l__enumext_label_v_tl for the `keyans` environment and the current *⟨label⟩* \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, which will be passed to the *⟨prop list⟩* defined by the `save-ans` key using the __enumext_store_addto_prop:V.

```

2860 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2861 {
2862     \tl_clear:N \l__enumext_store_current_label_tl
2863     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2864     {
2865         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2866     }
2867     {
2868         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2869     }
2870     \tl_if_novalue:nF { #1 }
2871     {
2872         % Set save-sep
2873         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2874         {
2875             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2876         }
2877         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2878     }
2879     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2880 }

```

(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 the `\anskey` command, basically because in this environments we are interested in the current *⟨label⟩*. The mechanism defined here will allow to execute *\ref{⟨store name : position⟩}* and will return *1*. (A).

__enumext_keyans_store_ref:
 __enumext_keyans_store_ref_aux_i:
 __enumext_keyans_store_ref_aux_ii:

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

```

2881 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2882 {
2883     \bool_if:NT \l__enumext_store_ref_key_bool
2884     {
2885         \cs_set_protected:Npn \__enumext_tmp:n ##1
2886         {
2887             \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2888             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2889             \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2890             \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2891         }
2892         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2893         \__enumext_keyans_store_ref_aux_i:

```

```

2894     }
2895 }

```

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

```

2896 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2897 {
2898   \bool_if:NT \g__enumext_starred_bool
2899   {
2900     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2901   }
2902   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2903   {
2904     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2905       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2906   }
2907   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2908   {
2909     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2910       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2911   }
2912   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2913   {
2914     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2915       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2916   }
2917   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2918   {
2919     \l__enumext_store_name_tl \c_colon_str
2920     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2921   }
2922   \__enumext_keyans_store_ref_aux_ii:
2923 }

```

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.

```

2924 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2925 {
2926   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2927   {
2928     \__enumext_newlabel:nn
2929       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2930       { \l__enumext_newlabel_arg_two_tl }
2931   }
2932   \l__enumext_write_aux_file_tl
2933 }

```

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

12.32.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2934 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2935 {
2936   \tl_clear:N \l__enumext_store_current_label_tl
2937   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2938   {
2939     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2940   }
2941   {
2942     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2943   }
2944   \tl_if_no_value:nF { #1 }
2945   {
2946     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2947     {

```



```

2948         \tl_put_right:Ne \l__enumext_store_current_label_tl
2949         {
2950             \l__enumext_store_keyans_item_opt_sep_tl
2951         }
2952     }
2953     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2954 }
2955 \__enumext_keyans_addto_seq_link:
2956 }

```

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.

```

2957 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2958 {
2959     \bool_lazy_and:nnT
2960     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2961     { \bool_if_p:N \l__enumext_hyperref_bool }
2962     {
2963         \tl_put_right:Ne \l__enumext_store_current_label_tl
2964         {
2965             \hfill \exp_not:N \hyperlink
2966             {
2967                 \exp_not:V \l__enumext_newlabel_arg_one_tl
2968             }
2969             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2970         }
2971     }
2972     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2973     \bool_if:NT \l__enumext_check_answers_bool
2974     {
2975         \int_gincr:N \g__enumext_item_anskey_int
2976     }
2977 }

```

(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.

```

2978 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2979 {
2980     \tl_if_novalue:nF { #1 }
2981     {
2982         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2983     }
2984     \bool_if:NT \l__enumext_show_answer_bool
2985     {
2986         \__enumext_keyans_show_ans:
2987     }
2988     \bool_if:NT \l__enumext_show_position_bool
2989     {
2990         \__enumext_keyans_show_pos:
2991     }
2992 }
2993 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2994 {
2995     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2996     {
2997         \bool_lazy_or:nnT
2998         { \bool_if_p:N \l__enumext_show_answer_bool }
2999         { \bool_if_p:N \l__enumext_show_position_bool }
3000         {
3001             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3002         }

```

```

3003     }
3004 }
3005 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3006 {
3007     \bool_if:NT \l__enumext_starred_bool
3008     {
3009         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3010         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3011     }
3012     \tl_put_left:Nn \l__enumext_label_v_tl
3013     {
3014         \__enumext_print_keyans_box:NN
3015         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3016     }
3017 }
3018 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3019 {
3020     \bool_if:NT \l__enumext_starred_bool
3021     {
3022         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3023         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3024     }
3025     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3026     {
3027         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3028         {
3029             \group_begin:
3030             \exp_not:N \normalfont
3031             \exp_not:N \footnotesize [ \int_eval:n
3032             {
3033                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3034             }
3035             ]
3036             \group_end:
3037         }
3038     }
3039     {
3040         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3041         {
3042             \group_begin:
3043             \exp_not:N \normalfont
3044             \exp_not:N \footnotesize [ \int_eval:n
3045             {
3046                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3047             }
3048             ]
3049             \group_end:
3050         }
3051     }
3052     \tl_put_left:Nn \l__enumext_label_v_tl
3053     {
3054         \__enumext_print_keyans_box:NN
3055         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3056     }
3057 }

```

(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`.

```

3058 \cs_new_protected:Npn \__enumext_default_item:n #1
3059 {
3060   \tl_if_novalue:nTF {#1}
3061   {
3062     \bool_if:NT \__enumext_check_answers_bool
3063     {
3064       \int_gincr:N \g__enumext_item_number_int
3065       \bool_set_true:N \__enumext_item_number_bool
3066     }
3067     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3068     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3069   }
3070   {
3071     \bool_set_eq:cc
3072     { \__enumext_wrap_label_ \__enumext_level: _bool }
3073     { \__enumext_wrap_label_opt_ \__enumext_level: _bool }
3074     \__enumext_item_std:w [#1] \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3075   }
3076 }

```

(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`.

```

3077 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3078 {
3079   \tl_if_novalue:nTF {#1}
3080   {
3081     \tl_gset_eq:Nc
3082     \g__enumext_item_symbol_aux_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
3083   }
3084   {
3085     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3086   }
3087   \tl_if_novalue:nTF {#2}
3088   {
3089     \dim_set_eq:cc
3090     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
3091     { \__enumext_labelsep_ \__enumext_level: _dim }
3092   }
3093   {
3094     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3095   }
3096   \bool_if:NT \__enumext_check_answers_bool
3097   {
3098     \int_gincr:N \g__enumext_item_number_int
3099     \bool_set_true:N \__enumext_item_number_bool
3100   }
3101   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3102   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3103 }

```

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

```

3104 \cs_new_protected:Nn \__enumext_item_star_exec:
3105 {
3106   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3107   {
3108     \mode_leave_vertical:

```

```

3109     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3110     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_tl }
3111     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3112   }
3113 }

```

(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*.

```

3114 \cs_new_protected:Nn \__enumext_redefine_item:
3115 {
3116   \RenewDocumentCommand \item { s o o }
3117   {
3118     \bool_if:nTF {##1}
3119     {
3120       \__enumext_starred_item:nn {##2} {##3}
3121     }
3122     { \__enumext_default_item:n {##2} }
3123   }
3124 }

```

The function __enumext_make_label: redefine \makeLabel for the keys align, font, wrap-label, wrap-label* and \item* for enumext environment.

```

3125 \cs_new_protected:Nn \__enumext_make_label:
3126 {
3127   \RenewDocumentCommand \makeLabel { m }
3128   {
3129     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3130     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3131     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3132     {
3133       \__enumext_item_star_exec:
3134       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3135     }
3136     { ##1 }
3137     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3138     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3139   }
3140 }

```

(End of definition for __enumext_redefine_item: and __enumext_make_label.)

🌱 These functions are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§12.38).

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*
3141 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3142 {
3143   \keys_define:nn { enumext / #1 }
3144   {
3145     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3146     item-sym* .value_required:n = true,
3147     item-sym* .initial:n = {$\star$},
3148     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3149     item-pos* .value_required:n = true,
3150   }
3151 }
3152 \clist_map_inline:nn
3153 {
3154   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3155 }
3156 { \__enumext_tmp:nn #1 }

```

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

12.35 Handling unknown keys

At this point in the code I already know that I will not add more ⟨keys⟩ and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the ⟨keys⟩ (you have to be consistent in life).

12.35.1 Handling unknown keys for keyans and keyans*

unknown

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

```

3157 \cs_set_protected:Npn \__enumext_tmp:n #1
3158 {
3159   \keys_define:nn { enumext / #1 }
3160   {
3161     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3162   }
3163 }
3164 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3165 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3166 {
3167   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3168 }
3169 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3170 {
3171   \tl_if_blank:nTF {#2}
3172   {
3173     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3174   }
3175   {
3176     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3177   }
3178 }

```

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

12.35.2 Handling unknown keys for enumext*

unknown

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

```

3179 \keys_define:nn { enumext / enumext* }
3180 {
3181   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3182 }

```

Internal functions for handling `unknown` key.

```

3183 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3184 {
3185   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3186 }
3187 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3188 {
3189   \tl_if_blank:nTF {#2}
3190   {
3191     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3192   }
3193   {
3194     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3195   }
3196 }

```

(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.

```

3197 \cs_set_protected:Npn \__enumext_tmp:n #1
3198 {
3199   \keys_define:nn { enumext / #1 }
3200   {
3201     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3202   }
3203 }
3204 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3205 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3206 {
3207   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3208 }
3209 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2

```

```

3210 {
3211   \tl_if_blank:nTF {#2}
3212   {
3213     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3214   }
3215   {
3216     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3217   }
3218 }

```

(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`.

```

3219 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3220 {
3221   \tl_if_novalue:nTF { #1 }
3222   {
3223     \bool_set_true:N \__enumext_wrap_label_v_bool
3224     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3225   }
3226   {
3227     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3228     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3229   }
3230 }

```

(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⟩`.

```

3231 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3232 {
3233   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3234   \__enumext_keyans_show_left:n { #1 }
3235   \bool_set_true:N \__enumext_wrap_label_v_bool
3236   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

3237   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3238   \__enumext_keyans_addto_prop:n { #1 }
3239   \__enumext_keyans_store_ref:
3240   \__enumext_keyans_addto_seq:n { #1 }
3241   \int_gincr:N \g__enumext_check_starred_cmd_int
3242 }

```

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

`\item*`
`__enumext_keyans_redefine_item:`
`__enumext_keyans_make_label:`

The function `__enumext_keyans_redefine_item:` is responsible for adding the *starred* 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.

```

3243 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3244 {
3245   \RenewDocumentCommand \item { s o }
3246   {
3247     \bool_if:nTF {##1}
3248     {
3249       \peek_remove_spaces:n
3250       {
3251         \__enumext_keyans_starred_item:n {##2}
3252       }
3253     }

```

```

3254     {
3255         \__enumext_keyans_default_item:n {##2}
3256     }
3257 }
3258 }

```

The function `__enumext_keyans_make_label:` redefine `\makeLabel` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `keyans` environment.

```

3259 \cs_new_protected:Nn \__enumext_keyans_make_label:
3260 {
3261     \RenewDocumentCommand \makeLabel { m }
3262     {
3263         \tl_use:N \l__enumext_label_fill_left_v_tl
3264         \tl_use:N \l__enumext_label_font_style_v_tl
3265         \bool_if:NTF \l__enumext_wrap_label_v_bool
3266         {
3267             \__enumext_wrapper_label_v:n { ##1 }
3268         }
3269         { ##1 }
3270         \tl_use:N \l__enumext_label_fill_right_v_tl
3271     }
3272 }

```

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

- This functions are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§12.37.2).

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.

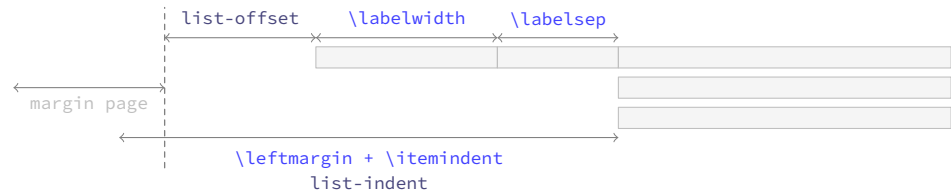


Figure 9: Representation of standard horizontal lengths in `list` environment.

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.

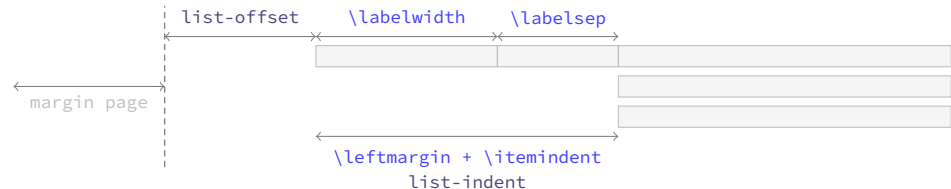


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

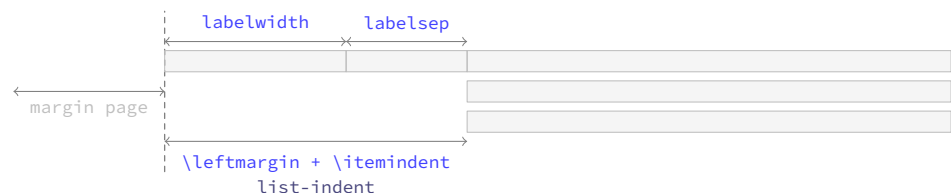


Figure 11: Default horizontal lengths in `enumext`.

```

\__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).

```

3273 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3274 {
3275   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3276   {
3277     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3278     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3279   }
3280   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3281   {
3282     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3283     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3284   }

```

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

```

3285   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3286   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3287   {
3288     \dim_set:Nn #6 { #1 + #2 - #4 }
3289     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3290   }
3291   {
3292     \dim_compare:nNnT { #4 } = { #1 + #2 }
3293     { \dim_set:Nn #6 { \c_zero_dim } }
3294     \dim_compare:nNnT { #4 } < { #1 + #2 }
3295     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3296     \dim_compare:nNnT { #4 } > { #1 + #2 }
3297     {
3298       \dim_set:Nn #6 { -#1 - #2 + #4 }
3299       \dim_set:Nn #6 { #6*-1 }
3300     }
3301     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3302   }
3303 }
3304 \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:
3305 \cs_set_protected:Npn \__enumext_tmp:n #1
3306 {
3307   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3308   {
3309     \__enumext_calc_hspace:ccccc
3310     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3311     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3312     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3313     { \__enumext_leftmargin_tmp_#1_bool }
3314     \clist_map_inline:nn
3315     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3316     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3317     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3318     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3319     \usecounter { enumX#1 }
3320     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3321     \str_if_eq:nnTF {#1} { v }
3322     {
3323       \__enumext_keyans_redefine_item:

```

```

3324     \__enumext_keyans_make_label:
3325     \__enumext_keyans_ref:
3326     \__enumext_keyans_fake_item:
3327     \bool_if:cT { \__enumext_show_length_#1_bool }
3328     {
3329         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3330     }
3331 }
3332 {
3333     \__enumext_redefine_item:
3334     \__enumext_make_label:
3335     \__enumext_standar_ref:
3336     \__enumext_fake_item:
3337     \bool_if:cT { \__enumext_show_length_#1_bool }
3338     {
3339         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_i
3340     }
3341 }
3342 }
3343 }
3344 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

3345 \cs_set_protected:Npn \__enumext_tmp:n #1
3346 {
3347     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3348     {
3349         \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3350         \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3351         \__enumext_calc_hspace:cccccc
3352         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3353         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3354         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3355         { \__enumext_leftmargin_tmp_#1_bool }
3356         \clist_map_inline:nn
3357         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3358         { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3359         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3360         { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3361         \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3362         \skip_zero:N \partopsep
3363         \usecounter { enumX#1 }
3364         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3365         \__enumext_starred_ref:
3366         \str_if_eq:nnTF {#1} { vii }
3367         {
3368             \__enumext_fake_item_vii:
3369             \bool_if:cT { \__enumext_show_length_vii_bool }
3370             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3371         }
3372         {
3373             \__enumext_fake_item_viii:
3374             \bool_if:cT { \__enumext_show_length_#1_bool }
3375             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3376         }
3377     }
3378 }
3379 \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` We create the `enumext` environment based on `list` environment by levels.

```

3380 \NewDocumentEnvironment{enumext}{0}{}
3381 {
3382     \__enumext_safe_exec:

```

```

3383     \__enumext_parse_keys:n {#1}
3384     \__enumext_before_list:
3385     \__enumext_start_store_level:
3386     \__enumext_start_list:nn
3387     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3388     {
3389         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3390         \__enumext_before_keys_exec:
3391     }
3392     \__enumext_set_item_width:
3393     \__enumext_after_args_exec:
3394 }
3395 {
3396     \__enumext_stop_list:
3397     \__enumext_stop_store_level:
3398     \__enumext_after_list:
3399 }

```

(End of definition for enumext. This function is documented on page 4.)

`__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.

```

3400 \cs_new_protected:Nn \__enumext_set_item_width:
3401 {
3402     \dim_set:Nn \itemwidth
3403     {
3404         \linewidth
3405     }
3406     \dim_compare:nT
3407     {
3408         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3409     }
3410     {
3411         \dim_sub:Nn \itemwidth
3412         {
3413             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3414         }
3415     }
3416 }

```

(End of definition for `__enumext_set_item_width:.`)

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_env*`, 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”.

```

3417 \cs_new_protected:Nn \__enumext_safe_exec:
3418 {
3419     \__enumext_internal_mini_page:
3420     \__enumext_is_not_nested:
3421     \int_incr:N \l__enumext_level_int
3422     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3423     { \msg_fatal:nn { enumext } { list-too-deep } }
3424     \bool_set_true:N \l__enumext_standar_bool
3425     \bool_set_false:N \l__enumext_starred_bool
3426     \__enumext_is_on_first_level:
3427 }

```

(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.

```

3428 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3429 {

```

```

3430 \tl_if_novalue:nF {#1}
3431 {
3432   \str_clear:N \l__enumext_series_str
3433   \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3434   {
3435     \keys_set:nn { enumext / level-1 } {#1}
3436     \__enumext_parse_series:n {#1}
3437     \__enumext_nested_base_line_fix:
3438   }
3439   {
3440     \exp_args:Ne \keys_set:nn
3441       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3442   }
3443   \__enumext_store_active_keys:n {#1}
3444 }
3445 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level:

__enumext_stop_store_level:

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

```

3446 \cs_new_protected:Nn \__enumext_start_store_level:
3447 {
3448   \bool_lazy_all:nT
3449   {
3450     { \bool_if_p:N \l__enumext_store_active_bool }
3451     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3452     { \bool_if_p:N \g__enumext_standar_bool }
3453   }
3454   {
3455     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3456     {
3457       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3458       \__enumext_store_level_open:
3459     }
3460   }

```

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

```

3461   \bool_lazy_all:nT
3462   {
3463     { \bool_if_p:N \l__enumext_store_active_bool }
3464     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3465     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3466   }
3467   {
3468     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3469     {
3470       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3471       \__enumext_store_level_open:
3472     }
3473   }
3474 }

```

Close the stored structure.

```

3475 \cs_new_protected:Nn \__enumext_stop_store_level:
3476 {
3477   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3478   {
3479     \__enumext_store_level_close:
3480   }
3481 }

```

(End of definition for __enumext_start_store_level: and __enumext_stop_store_level:.)

__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.

```

3482 \cs_new_protected:Nn \__enumext_before_list:
3483 {
3484   \__enumext_vspace_above:
3485   \__enumext_before_args_exec:
3486   \__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_env*` 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_env*` environment on the “left side”, always having a current `\linewidth` as *maximum width* between them.

```

3487 \dim_compare:nNt
3488 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3489 {
3490   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3491   {
3492     \linewidth
3493     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3494     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3495   }

```

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_env*` 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.

```

3496 \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3497 \int_gincr:N \g__enumext_minipage_stat_int
3498 \__enumext_minipage_add_space:
3499 \begin{__enumext_mini_env*}
3500 { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3501 }
3502 \__enumext_multicols_start:
3503 }

```

(End of definition for `__enumext_before_list:`)

`__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.

```

3504 \cs_new_protected:Nn \__enumext_multicols_start:
3505 {
3506   \int_compare:nNt
3507   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3508   {
3509     \dim_compare:nNt
3510     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3511     {
3512       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3513       {
3514         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3515           + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3516         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3517         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3518       }
3519     }
3520     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3521     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3522     {
3523       \dim_zero:N \columnseprule
3524     }
3525   }

```

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.

```

3525 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3526 {
3527   \skip_zero:N \multicolsep
3528   \__enumext_multi_addvspace:
3529 }
3530 \raggedcolumns
3531 \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3532 }
3533 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our “vertical adjust” spacing.

```

3534 \cs_new_protected:Nn \__enumext_multicols_stop:
3535 {
3536   \int_compare:nNt
3537     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3538   {
3539     \end{multicols}
3540     \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3541       {
3542         \__enumext_unskip_unkern:
3543         \__enumext_unskip_unkern:
3544         \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
3545       }
3546       {
3547         \__enumext_unskip_unkern:
3548         \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \__enumext_level: _skip } }
3549       }
3550   }
3551 }
```

(End of definition for `__enumext_multicols_stop:`.)

`__enumext_after_list:` The function `__enumext_after_list:` 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_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3552 \cs_new_protected:Nn \__enumext_after_list:
3553 {
3554   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3555     {
3556       \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3557       {
3558         \msg_warning:nn { enumext } { missing-miniright }
3559         \miniright
3560       }
3561       \int_gzero:N \g__enumext_minipage_stat_int
3562       \__enumext_unskip_unkern: % remove topsep + [partopsep]
3563       \end{__enumext_mini_env*}
3564     }
3565     {
3566       \__enumext_multicols_stop:
3567     }
```

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.

```

3568   \__enumext_after_stop_list:
3569   \__enumext_check_ans_key_hook:
3570   \__enumext_vspace_below:
3571   \bool_set_false:N \l__enumext_standar_bool
3572   \__enumext_resume_save_counter:
3573 }
```

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`.

```

3574 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

(End of definition for `__enumext_after_list:`.)

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”.

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

```

3575 \NewDocumentEnvironment{keyans}{0}{ }
3576 {
3577   \__enumext_keyans_safe_exec:
3578   \__enumext_keyans_parse_keys:n {#1}
3579   \__enumext_before_list_v:
3580   \__enumext_start_list:nn
3581   { \tl_use:N \__enumext_label_v_tl }
3582   {
3583     \__enumext_list_arg_two_v:
3584     \__enumext_before_keys_exec_v:
3585   }
3586   \__enumext_keyans_set_item_width:
3587   \__enumext_after_args_exec_v:
3588 }
3589 {
3590   \__enumext_check_starred_cmd:n { item }
3591   \__enumext_stop_list:
3592   \__enumext_after_list_v:
3593 }

```

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

__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.

```

3594 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3595 {
3596   \dim_set:Nn \itemwidth
3597   {
3598     \linewidth
3599   }
3600   \dim_compare:nT
3601   {
3602     \l__enumext_listoffset_v_dim != \c_zero_dim
3603   }
3604   {
3605     \dim_sub:Nn \itemwidth
3606     {
3607       \l__enumext_listoffset_v_dim
3608     }
3609   }
3610 }

```

(End of definition for **__enumext_keyans_set_item_width:**.)

__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.

```

3611 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3612 {
3613   \bool_if:NF \l__enumext_store_active_bool
3614   {
3615     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3616   }
3617   \int_incr:N \l__enumext_keyans_level_int
3618   \bool_set_true:N \l__enumext_keyans_env_bool
3619   \__enumext_keyans_name_and_start:
3620   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3621   \bool_set_false:N \l__enumext_store_active_bool
3622   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3623   {
3624     \msg_error:nn { enumext } { keyans-nested }
3625   }
3626   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3627   {
3628     \msg_error:nn { enumext } { keyans-wrong-level }
3629   }
3630 }

```

(End of definition for **__enumext_keyans_safe_exec:**.)

\\enumext_keyans_parse_keys:n

Parse [*key* = *val*] for *keyans* environment.

```

3631 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3632 {
3633   \keys_set:nn { enumext / keyans } {#1}
3634 }

```

(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_after_list_v:

```

3635 \cs_new_protected:Nn \__enumext_before_list_v:
3636 {
3637   \__enumext_vspace_above_v:
3638   \__enumext_before_args_exec_v:
3639   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3640   {
3641     \dim_set:Nn \l__enumext_minipage_left_v_dim
3642     {
3643       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3644     }
3645     \bool_set_true:N \l__enumext_minipage_active_v_bool
3646     \int_gincr:N \g__enumext_minipage_stat_int
3647     \__enumext_keyans_minipage_add_space:
3648     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3649   }
3650   \__enumext_keyans_multicols_start:
3651 }
3652 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3653 {
3654   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3655   {
3656     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3657     {
3658       \dim_set:Nn \l__enumext_columns_sep_v_dim
3659       {
3660         (
3661           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3662         ) / \l__enumext_columns_v_int
3663         - \l__enumext_listoffset_v_dim
3664       }
3665     }
3666     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3667     \dim_zero:N \columnseprule % no rule here
3668     \bool_if:NF \l__enumext_minipage_active_v_bool
3669     {
3670       \skip_zero:N \multicolsep
3671       \__enumext_keyans_multi_addvspace:
3672     }
3673     \raggedcolumns
3674     \begin{multicols}{ \l__enumext_columns_v_int }
3675   }
3676 }
3677 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3678 {
3679   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3680   {
3681     \end{multicols}
3682     \bool_if:NF \l__enumext_minipage_active_v_bool
3683     {
3684       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3685     }
3686   }
3687 }
3688 \cs_new_protected:Nn \__enumext_after_list_v:
3689 {
3690   \bool_if:NTF \l__enumext_minipage_active_v_bool
3691   {
3692     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3693     {
3694       \msg_warning:nn { enumext } { missing-miniright }
3695       \miniright
3696     }
3697   }
3698 }

```

```

3697 \int_gzero:N \g__enumext_minipage_stat_int
3698 \unskip % remove topsep + [partopsep]
3699 \end{__enumext_mini_env*}
3700 \par\addvspace{ \l__enumext_minipage_after_skip }
3701 }
3702 {
3703 \__enumext_keyans_multicols_stop:
3704 }
3705 \bool_set_false:N \l__enumext_keyans_env_bool
3706 \__enumext_after_stop_list_v:
3707 \__enumext_vspace_below_v:
3708 }

```

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

12.40 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a list-based environment that uses the same configuration for “spacing” and `\label` as the `keyans` environment, but it does not use `\item`.

The contents are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the `\label` underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to “adjust” the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

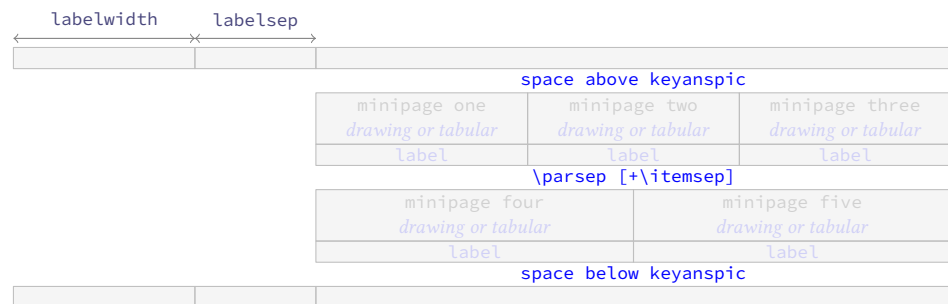


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

This implementation is adapted from the answer given by Enrico Gregorio in [How to process the body of an environment and divide it by a `\macro`?](#).

12.40.1 The command `\anspic`

`\anspic` The `\anspic` command take three arguments, the starred (*) versions `\anspic*` and `\anspic*[\langle content \rangle]` store the current `\label` next to the `[\langle content \rangle]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key. This command is used as a replacement for `\item` in the `keyanspic` environment.

```

3709 \NewDocumentCommand \anspic { s o +m }
3710 {

```

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error.

```

3711 \bool_if:NF \l__enumext_store_active_bool
3712 {
3713 \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3714 }
3715 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3716 {
3717 \msg_error:nn { enumext } { keyanspic-wrong-level }
3718 }
3719 \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3720 {
3721 \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3722 }

```

The three arguments are handled by the function `__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the `keyanspic` environment.

```

3723 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3724 {
3725 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3726 }
3727 }

```

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

`__enumext_keyans_anspic_code:nnn`

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

```

3728 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3729 {
3730   \stepcounter { enumXvi }
3731   #3 \\\
3732   \bool_if:nT { #1 }
3733   {
3734     \__enumext_keyans_addto_prop:n { #2 }
3735     \__enumext_keyans_store_ref:
3736     \__enumext_keyans_addto_seq:n { #2 }
3737     \int_gincr:N \g__enumext_check_starred_cmd_int
3738     \bool_lazy_or:nnT
3739     { \bool_if_p:N \l__enumext_show_answer_bool }
3740     { \bool_if_p:N \l__enumext_show_position_bool }
3741     {
3742       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3743       \__enumext_keyans_show_left:n { #2 }
3744       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3745     }
3746   }
3747   \tl_use:N \l__enumext_label_font_style_v_tl
3748   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3749 }

```

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

12.40.2 The environment `keyanspic`

`keyanspic`

Now we define the environment `keyanspic` based on list. The optional argument [$\langle number\ above,\ number\ below \rangle$] will determine the number of `minipage` environments that will be above and below separated by `\parsep+\itemsep` within it.

```

3750 \NewDocumentEnvironment{keyanspic}{o}
3751 {
3752   \__enumext_keyans_pic_safe_exec:
3753   \__enumext_start_list:nn
3754   { }
3755   {
3756     \__enumext_keyans_pic_arg_two:
3757   }

```

We apply the “adjusted” vertical spacing above the environment

```

3758   \vspace { \l__enumext_keyans_pic_above_skip }
3759 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_keyans_pic_body_seq` and placed in `minipage` environments on a single line. Finally we check if `\anspic*` has been used, set the counter to zero and apply our “adjusted” vertical space below the environment.

```

3760 {
3761   \tl_if_novalue:nTF { #1 }
3762   {
3763     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3764   }
3765   { \__enumext_keyans_pic_do:n { #1 } }
3766   \__enumext_stop_list:
3767   \__enumext_check_starred_cmd:n { anspic }
3768   \setcounter { enumXvi } { 0 }
3769   \vspace { \l__enumext_topsep_v_skip }
3770   %\bool_set_false:N \l__enumext_store_active_bool
3771 }

```

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

`__enumext_keyans_pic_safe_exec:`

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

```

3772 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3773 {
3774   \int_incr:N \l__enumext_keyans_pic_level_int
3775   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3776   {
3777     \msg_error:nn { enumext } { keyanspic-nested }

```

```

3778     }
3779     \__enumext_keyans_name_and_start:
3780 }

```

(End of definition for __enumext_keyans_pic_safe_exec:.)

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

```

3781 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3782 {
3783     \dim_compare:nNnT { #1 } < { 0pt }
3784     { \skip_set:Nn #1 { -#1 } }
3785 }

```

(End of definition for __enumext_keyans_pic_skip_abs:N.)

__enumext_keyans_pic_arg_two: The function __enumext_keyans_pic_arg_two: will be used in the second argument of the __enumext_start_list:nn function that defines the `keyanspic` environment, it will handle the setting of spaces.

```

3786 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
3787 {

```

The first thing to do is to set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```

3788     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3789     \__enumext_list_arg_two_v:

```

We will add the value of \itemsep to \parsep which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of \leftmargin, the label and counter are handled directly by the \anspic command. Then we make equal to zero \labelwidth, \labelsep, \partopsep and \itemsep so that the horizontal and vertical spacing is not affected.

```

3790     \skip_add:Nn \parsep { \itemsep }
3791     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3792     \dim_zero:N \labelwidth
3793     \dim_zero:N \listparindent
3794     \dim_zero:N \labelsep
3795     \skip_zero:N \partopsep
3796     \skip_zero:N \itemsep

```

We set the value of \l__enumext_keyans_pic_above_skip which we will use to apply our “adjust” space above `keyanspic`, finally we call __enumext_item_std:w followed by \scan_stop: to prevent the error message returned by \TeX when not using the \item command.

```

3797     \__enumext_keyans_pic_skip_abs:N \parsep
3798     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3799     {
3800         \box_dp:N \strutbox
3801         + \l__enumext_topsep_v_skip
3802         - \parsep
3803     }
3804     \__enumext_item_std:w \scan_stop:
3805     % paranoia
3806     \RenewDocumentCommand \item {}
3807     {
3808         \msg_error:nn { enumext } { keyanspic-item-cmd }
3809     }
3810 }

```

(End of definition for __enumext_keyans_pic_arg_two:.)

__enumext_keyans_pic_do:n The optional argument is split by comma and is handled directly by the function __enumext_keyans_pic_do:n and passed to the function __enumext_keyans_pic_row:n.

```

3811 \cs_new_protected:Npn \__enumext_keyans_pic_do:n
3812 {
3813     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3814 }
3815 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for __enumext_keyans_pic_do:n.)

`__enumext_keyans_pic_row:n`

The function `__enumext_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content *⟨stored⟩* by `\anspic*` in the `\l__enumext_keyans_pic_body_seq` sequence inside them.

```

3816 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3817 {
3818   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3819   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3820   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3821   \int_step_inline:nnn
3822     { \l__enumext_keyans_pic_above_int + 1 }
3823     { \l__enumext_keyans_pic_below_int }
3824     {
3825       \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3826       \centering
3827       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3828       \__enumext_endminipage:
3829     }
3830   \par
3831 }

```

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

12.41 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` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

12.42 Redefining `\footnote` command

`__enumext_footnotetext:nn`

`__enumext_renew_footnote:`

`__enumext_print_footnote:`

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

```

3832 \cs_new_protected:Nn \__enumext_footnotetext:nn
3833 {
3834   \footnotetext[#1]{#2}
3835 }
3836 \cs_new_protected:Nn \__enumext_renew_footnote:
3837 {
3838   \seq_gclear:N \g__enumext_footnote_arg_seq
3839   \seq_gclear:N \g__enumext_footnote_int_seq
3840   \RenewDocumentCommand \footnote { o +m }
3841   {
3842     \tl_if_novalue:nTF {##1}
3843     {
3844       \stepcounter{footnote}
3845       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3846     }
3847     {
3848       \int_gset:Nn \g__enumext_footnote_int { ##1 }
3849     }
3850     \footnotemark [ \g__enumext_footnote_int ]
3851     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3852     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3853   }
3854 }
3855 \cs_new_protected:Nn \__enumext_print_footnote:
3856 {
3857   \seq_if_empty:NF \g__enumext_footnote_int_seq
3858   {
3859     \seq_map_pairwise_function:NNN
3860       \g__enumext_footnote_int_seq
3861       \g__enumext_footnote_arg_seq
3862       \__enumext_footnotetext:nn
3863   }
3864 }

```

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

12.42.1 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content 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 optional first argument (`\langle number \rangle`).

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

```
\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
```

```
3865 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3866 {
3867   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3868   {
3869     \dim_set:Nn \__enumext_columns_sep_vii_dim
3870     {
3871       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
3872       / \__enumext_columns_vii_int
3873     }
3874   }
3875   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
3876   \dim_set:Nn \__enumext_item_width_vii_dim
3877   {
3878     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
3879     / \__enumext_columns_vii_int
3880     - \__enumext_labelwidth_vii_dim
3881     - \__enumext_labelsep_vii_dim
3882   }
```

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

```
3883   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3884   {
3885     \dim_sub:Nn \__enumext_item_width_vii_dim
3886     {
3887       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
3888       / \__enumext_columns_vii_int
3889     }
3890     \dim_add:Nn \__enumext_columns_sep_vii_dim
3891     {
3892       \__enumext_rightmargin_vii_dim
3893     }
3894   }
3895 }
```

Same implementation for the `keyans*` environment.

```
3896 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3897 {
3898   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3899   {
3900     \dim_set:Nn \__enumext_columns_sep_viii_dim
3901     {
3902       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
3903       / \__enumext_columns_viii_int
3904     }
3905   }
3906   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
3907   \dim_set:Nn \__enumext_item_width_viii_dim
3908   {
3909     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
3910     / \__enumext_columns_viii_int
3911     - \__enumext_labelwidth_viii_dim
3912     - \__enumext_labelsep_viii_dim
3913   }
3914   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3915   {
3916     \dim_sub:Nn \__enumext_item_width_viii_dim
3917     {
3918       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_viii_int )
3919       / \__enumext_columns_viii_int
3920     }
3921     \dim_add:Nn \__enumext_columns_sep_viii_dim
3922     {
3923       \__enumext_rightmargin_viii_dim
3924     }
3925   }
```

```

3924     }
3925   }
3926 }

```

(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.

```

3927 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3928 {
3929   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3930   \int_compare:nNtT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3931   {
3932     \msg_warning:nnee { enumext } { item-joined }
3933     { \int_use:N \l__enumext_joined_item_vii_int }
3934     { \int_use:N \l__enumext_columns_vii_int }
3935     \int_set:Nn \l__enumext_joined_item_vii_int
3936     {
3937       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3938     }
3939   }
3940   \int_compare:nNtT
3941   { \l__enumext_joined_item_vii_int }
3942   >
3943   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3944   {
3945     \msg_warning:nnee { enumext } { item-joined-columns }
3946     { \int_use:N \l__enumext_joined_item_vii_int }
3947     {
3948       \int_eval:n
3949       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3950     }
3951     \int_set:Nn \l__enumext_joined_item_vii_int
3952     {
3953       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3954     }
3955   }
3956   \int_compare:nNtTF { \l__enumext_joined_item_vii_int } > { 1 }
3957   {
3958     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3959     \int_decr:N \l__enumext_joined_item_aux_vii_int
3960     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3961     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3962     \dim_set:Nn \l__enumext_joined_width_vii_dim
3963     {
3964       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3965       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3966         + \l__enumext_columns_sep_vii_dim
3967       ) * \l__enumext_joined_item_aux_vii_int
3968     }
3969     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3970   }
3971   {
3972     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3973     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3974   }
3975 }

```

Same implementation for the `keyans*` environment.

```

3976 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3977 {
3978   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3979   \int_compare:nNtT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3980   {
3981     \msg_warning:nnee { enumext } { item-joined }
3982     { \int_use:N \l__enumext_joined_item_viii_int }
3983     { \int_use:N \l__enumext_columns_viii_int }
3984     \int_set:Nn \l__enumext_joined_item_viii_int
3985     {

```



```

3986         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3987     }
3988 }
3989 \int_compare:nNnT
3990 { \l__enumext_joined_item_viii_int }
3991 >
3992 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3993 {
3994     \msg_warning:nnee { enumext } { item-joined-columns }
3995     { \int_use:N \l__enumext_joined_item_viii_int }
3996     {
3997         \int_eval:n
3998         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3999     }
4000     \int_set:Nn \l__enumext_joined_item_viii_int
4001     {
4002         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4003     }
4004 }
4005 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4006 {
4007     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4008     \int_decr:N \l__enumext_joined_item_aux_viii_int
4009     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4010     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4011     \dim_set:Nn \l__enumext_joined_width_viii_dim
4012     {
4013         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4014         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4015           + \l__enumext_columns_sep_viii_dim
4016           ) * \l__enumext_joined_item_aux_viii_int
4017     }
4018     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4019 }
4020 {
4021     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4022     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4023 }
4024 }

```

(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_env*` 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`.

```

4025 \cs_new_protected:Nn \__enumext_start_mini_vii:
4026 {
4027     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4028     {
4029         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4030         {
4031             \linewidth
4032             - \l__enumext_minipage_right_vii_dim
4033             - \l__enumext_minipage_hsep_vii_dim
4034         }
4035         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4036         \dim_gset_eq:NN
4037             \g__enumext_minipage_right_vii_dim
4038             \l__enumext_minipage_right_vii_dim
4039         \__enumext_mini_addvspace_vii:
4040         \nointerlineskip\noindent
4041         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
4042     }
4043 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_env*` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `__enumext_after_env:n` to execute the `__enumext_mini_env*` on the “right side”.

```

4044 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4045 {
4046   \bool_if:NT \l__enumext_minipage_active_vii_bool
4047   {
4048     \end{__enumext_mini_env*}
4049     \hfill
4050     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4051   }
4052 }

```

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 `__enumext_mini_env*` 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.

```

4053 \__enumext_after_env:n {enumext*}
4054 {
4055   \bool_if:NT \g__enumext_minipage_active_vii_bool
4056   {
4057     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
4058     \par\addvspace { \g__enumext_minipage_right_skip }
4059     \bool_if:NF \g__enumext_minipage_center_vii_bool
4060     {
4061       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4062       {
4063         \centering
4064       }
4065     }
4066     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4067     {
4068       \tl_use:N \g__enumext_miniright_code_vii_tl
4069     }
4070     \box_use_drop:N \l__enumext_miniright_code_vii_box
4071     \end{__enumext_mini_env*}
4072     \par\addvspace{ \g__enumext_minipage_after_skip }
4073   }
4074   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4075   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4076   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4077   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4078   \bool_gset_false:N \g__enumext_starred_bool
4079 }

```

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

`__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.

```

4080 \cs_new_protected:Nn \__enumext_start_mini_viii:
4081 {
4082   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4083   {
4084     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4085     {
4086       \linewidth
4087       - \l__enumext_minipage_right_viii_dim
4088       - \l__enumext_minipage_hsep_viii_dim
4089     }
4090     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4091     \dim_gset_eq:NN
4092     \g__enumext_minipage_right_viii_dim
4093     \l__enumext_minipage_right_viii_dim
4094     \__enumext_mini_addvspace_viii:
4095     \nointerlineskip\noindent
4096     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
4097   }
4098 }
4099 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4100 {
4101   \bool_if:NT \l__enumext_minipage_active_viii_bool
4102   {
4103     \end{__enumext_mini_env*}
4104     \hfill

```

```

4105     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4106   }
4107 }
4108 \__enumext_after_env:nn {keyans*}
4109 {
4110   \bool_if:NT \g__enumext_minipage_active_viii_bool
4111   {
4112     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
4113     \par\addvspace { \g__enumext_minipage_right_skip }
4114     \bool_if:NF \g__enumext_minipage_center_viii_bool
4115     {
4116       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4117       {
4118         \centering
4119       }
4120     }
4121     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4122     {
4123       \tl_use:N \g__enumext_miniright_code_viii_tl
4124     }
4125     \box_use_drop:N \l__enumext_miniright_code_viii_box
4126     \end{__enumext_mini_env*}
4127     \par\addvspace{ \g__enumext_minipage_after_skip }
4128   }
4129   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4130   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4131   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4132   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4133 }

```

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

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 \noindent and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later.

```

4134 \NewDocumentEnvironment{enumext*}{ o }
4135 {
4136   \__enumext_safe_exec_vii:
4137   \__enumext_parse_keys_vii:n {#1}
4138   \__enumext_before_list_vii:
4139   \__enumext_start_store_level_vii:
4140   \__enumext_start_list:nn { }
4141   {
4142     \__enumext_list_arg_two_vii:
4143     \__enumext_before_keys_exec_vii:
4144   }
4145   \__enumext_starred_columns_set_vii:
4146   \item[] \scan_stop:
4147   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4148   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4149 }
4150 {
4151   \__enumext_stop_item_tmp_vii:
4152   \__enumext_remove_extra_parsep_vii:
4153   \__enumext_stop_list:
4154   \__enumext_stop_store_level_vii:
4155   \__enumext_after_list_vii:
4156 }

```

(End of definition for enumext*. This function is documented on page 4.)

__enumext_safe_exec_vii: We will first call the function __enumext_internal_mini_page: to create the environment **__enumext-mini_env***, 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.

```

4157 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4158 {

```

```

4159     \__enumext_internal_mini_page:
4160     \__enumext_is_not_nested:
4161     \int_incr:N \__enumext_level_h_int
4162     \int_compare:nNnT { \__enumext_level_h_int } > { 1 }
4163     {
4164         \msg_error:nn { enumext } { nested }
4165     }
4166     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
4167     {
4168         \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4169     }
4170     \bool_set_true:N \__enumext_starred_bool
4171     \bool_set_false:N \__enumext_standar_bool
4172     \__enumext_is_on_first_level:
4173 }

```

(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 and finally we call the function __enumext_nested_base_line_fix: used by the key `base-fix`.

```

4174 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4175 {
4176     \tl_if_novalue:nF {#1}
4177     {
4178         \str_clear:N \l__enumext_series_str
4179         \keys_set:nn { enumext / enumext* } {#1}
4180         \__enumext_parse_series:n {#1}
4181         \__enumext_store_active_keys_vii:n {#1}
4182         \__enumext_nested_base_line_fix:
4183     }
4184 }

```

(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*`.

```

4185 \cs_new_protected:Nn \__enumext_before_list_vii:
4186 {
4187     \__enumext_vspace_above_vii:
4188     \__enumext_check_ans_active:
4189     \__enumext_before_args_exec_vii:
4190     \__enumext_start_mini_vii:
4191 }

```

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

```

4192 \cs_new_protected:Nn \__enumext_after_list_vii:
4193 {
4194     \__enumext_stop_mini_vii:
4195     \__enumext_after_stop_list_vii:
4196     \__enumext_check_ans_key_hook:
4197     \__enumext_vspace_below_vii:
4198     \bool_set_false:N \l__enumext_starred_bool
4199     \__enumext_resume_save_counter:
4200 }

```

(End of definition for __enumext_after_list_vii:.)

__enumext_start_store_level_vii:
 __enumext_stop_store_level_vii:

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

```

4201 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4202 {
4203   \bool_if:NT \l__enumext_store_active_bool
4204   {
4205     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
4206     {
4207       \__enumext_store_level_open_vii:
4208     }
4209   }
4210 }
4211 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4212 {
4213   \bool_if:NT \l__enumext_store_active_bool
4214   {
4215     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
4216     {
4217       \__enumext_store_level_close_vii:
4218     }
4219   }
4220 }
```

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

12.43.1 The command \item in enumext*

__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 \l__enumext_item_column_pos_vii_int that will count the item's by rows and the value of \g__enumext_item_count_all_vii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_vii: that will handle the arguments passed to \item.

```

4221 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4222 {
4223   \__enumext_stop_item_tmp_vii:
4224   \int_incr:N \l__enumext_item_column_pos_vii_int
4225   \int_gincr:N \g__enumext_item_count_all_vii_int
4226   \__enumext_item_peek_args_vii:
4227 }
```

(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).

```

4228 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4229 {
4230   \peek_meaning:NTF (
4231     { \__enumext_joined_item_vii:w }
4232     { \__enumext_joined_item_vii:w (1) }
4233   }
```

(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.

```

4234 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4235 {
4236   \__enumext_starred_joined_item_vii:n {#1}
4237   \peek_meaning_remove:NTF *
4238   { \__enumext_starred_item_vii:w }
4239   { \__enumext_standar_item_vii:w }
4240 }
```

(End of definition for __enumext_joined_item_vii:w.)

__enumext_standar_item_vii:w

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

```

4241 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4242 {
4243   \bool_set_false:N \l__enumext_item_starred_vii_bool
4244   \peek_meaning:NTF [
4245     {
4246       \bool_set_eq:NN
4247         \l__enumext_wrap_label_vii_bool
4248         \l__enumext_wrap_label_opt_vii_bool
4249       \__enumext_start_item_vii:w
4250     }
4251     {
4252       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4253       \legacy_if_set_true:n { @noitemarg }
4254       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4255     }
4256   }

```

(End of definition for __enumext_standar_item_vii:w.)

__enumext_starred_item_vii:w

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

__enumext_starred_item_vii_aux_i:w

__enumext_starred_item_vii_aux_ii:w

__enumext_starred_item_vii_aux_iii:w

```

4257 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4258 {
4259   \bool_set_true:N \l__enumext_item_starred_vii_bool
4260   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4261   \peek_meaning:NTF [
4262     { \__enumext_starred_item_vii_aux_i:w }
4263     { \__enumext_starred_item_vii_aux_ii:w }
4264   }
4265   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4266   {
4267     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4268     \__enumext_starred_item_vii_aux_ii:w
4269   }
4270   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4271   {
4272     \peek_meaning:NTF [
4273       { \__enumext_starred_item_vii_aux_iii:w }
4274       {
4275         \dim_set_eq:NN
4276           \l__enumext_item_symbol_sep_vii_dim
4277           \l__enumext_labelsep_vii_dim
4278         \legacy_if_set_true:n { @noitemarg }
4279         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4280       }
4281     }
4282   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4283   {
4284     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4285     \legacy_if_set_true:n { @noitemarg }
4286     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4287   }

```

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

12.43.2 Real definition of \item in enumext*

__enumext_start_item_vii:w

The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true definition of `\item` inside the `enumext*` environment. 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 and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is

given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

4288 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4289 {
4290   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4291   \legacy_if:nT { @noitemarg }
4292   {
4293     \legacy_if_set_false:n { @noitemarg }
4294     \legacy_if:nT { @nmbrlist }
4295     {
4296       \bool_if:NT \l__enumext_hyperref_bool
4297       {
4298         \legacy_if_set_true:n { @hyper@item }
4299       }
4300       \refstepcounter{enumXvii}
4301       \bool_if:NT \l__enumext_check_answers_bool
4302       {
4303         \int_gincr:N \g__enumext_item_number_int
4304         \bool_set_true:N \l__enumext_item_number_bool
4305       }
4306     }
4307   }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `\lrbox` environment. If the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the *symbol* defined for `\item*` if it is present and open a new group inside which we execute `font key` next to `\item` and the keys `wrap-label`, `wrap-label*`, `align`, close the group and execute the key `labelsep` and then the key `first`. Finally we open the `\minipage` environment and execute the `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

4308   \group_begin:
4309   \lrbox{ \l__enumext_item_text_vii_box }
4310   \bool_if:NF \l__enumext_footnotes_key_bool
4311   {
4312     \__enumext_renew_footnote:
4313   }
4314   \bool_if:NT \l__enumext_item_starred_vii_bool
4315   {
4316     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4317     {
4318       \tl_gset_eq:NN
4319       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4320     }
4321     \mode_leave_vertical:
4322     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4323     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4324     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4325     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4326   }
4327   \group_begin:
4328   \tl_use:N \l__enumext_label_font_style_vii_tl
4329   \bool_if:NTF \l__enumext_wrap_label_vii_bool
4330   {
4331     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4332     { \__enumext_wrapper_label_vii:n {#1} }
4333   }
4334   {
4335     \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4336   }
4337   \group_end:
4338   \skip_horizontal:N \l__enumext_labelsep_vii_dim
4339   \tl_use:N \l__enumext_after_list_args_vii_tl
4340   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4341   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4342   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4343   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4344 }

```

(End of definition for `__enumext_start_item_vii:w`)

`__enumext_stop_item_vii:` The function `__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its `\contents`. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4345 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4346 {
4347     \__enumext_endminipage:
4348     \endlrbox
4349     \group_end:
4350     \box_set_wd:Nn \l__enumext_item_text_vii_box
4351     {
4352         \l__enumext_joined_width_vii_dim
4353         + \l__enumext_labelwidth_vii_dim
4354         + \l__enumext_labelsep_vii_dim
4355     }
4356     \int_set:Nn \hbadness { 10000 }
4357     \box_use_drop:N \l__enumext_item_text_vii_box
4358     \bool_if:NF \l__enumext_footnotes_key_bool
4359     {
4360         \__enumext_print_footnote:
4361     }
4362     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4363     {
4364         \par\noindent
4365         \int_zero:N \l__enumext_item_column_pos_vii_int
4366     }
4367     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4368 }

```

(End of definition for `__enumext_stop_item_vii:`.)

`__enumext_remove_extra_parsep_vii:` 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.

```

4369 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4370 {
4371     \int_compare:nNnT
4372     {
4373         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4374     }
4375     =
4376     { 0 }
4377     {
4378         \par
4379         \vspace{ -\l__enumext_itemsep_vii_skip }
4380         \int_gzero:N \g__enumext_item_count_all_vii_int
4381     }
4382 }

```

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`.

```

4383 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`.)

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 `\noindent` and next to `\item` equal to `__enumext_start_item_tmp_viii:` which we will redefine later.

```

4384 \NewDocumentEnvironment{keyans*}{ o }
4385 {
4386     \__enumext_safe_exec_viii:
4387     \__enumext_parse_keys_viii:n {#1}
4388     \__enumext_before_list_viii:
4389     \__enumext_start_list:nn { }
4390     {
4391         \__enumext_list_arg_two_viii:
4392         \__enumext_before_keys_exec_viii:
4393     }
4394     \__enumext_starred_columns_set_viii:
4395     \item[] \scan_stop:
4396     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent

```

```

4397     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4398   }
4399   {
4400     \__enumext_stop_item_tmp_viii:
4401     \__enumext_remove_extra_parsep_viii:
4402     \__enumext_check_starred_cmd:n { item }
4403     \__enumext_stop_list:
4404     \__enumext_after_list_viii:
4405   }

```

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

`__enumext_safe_exec_viii:` First check the maximum nesting level for the `keyans*` environment.

```

4406 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4407 {
4408   \int_incr:N \__enumext_keyans_level_h_int
4409   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4410   {
4411     \msg_error:nn { enumext } { nested }
4412   }
4413   \__enumext_keyans_name_and_start:
4414   \bool_if:NT \__enumext_starred_bool
4415   {
4416     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4417   }
4418   \bool_set_true:N \__enumext_starred_bool
4419   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4420   \bool_set_false:N \__enumext_store_active_bool
4421   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4422   {
4423     \msg_error:nn { enumext } { keyans-wrong-level }
4424   }
4425 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

4426 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4427 {
4428   \tl_if_novalue:nF {#1}
4429   {
4430     \keys_set:nn { enumext / keyans* } {#1}
4431   }
4432 }

```

(End of definition for `__enumext_parse_keys_viii:n`)

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

```

4433 \cs_new_protected:Nn \__enumext_before_list_viii:
4434 {
4435   \__enumext_vspace_above_viii:
4436   \__enumext_before_args_exec_viii:
4437   \__enumext_start_mini_viii:
4438 }

```

(End of definition for `__enumext_before_list_viii:`)

`__enumext_after_list_viii:` The function `__enumext_after_list:` 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.

```

4439 \cs_new_protected:Nn \__enumext_after_list_viii:
4440 {
4441   \__enumext_stop_mini_viii:
4442   \__enumext_after_stop_list_viii:
4443   \__enumext_vspace_below_viii:
4444 }

```

(End of definition for `__enumext_after_list_viii:`)

12.44.1 The command `\item` in keyans*

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`\langle number \rangle`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\langle label \rangle` next to the `[\langle content \rangle]` if it is present in the `\langle sequence \rangle` and `\langle prop list \rangle` defined by `save-ans` key for `\item*`, `\item* [\langle content \rangle]`, `\item \langle number \rangle *` and `\item \langle number \rangle * [\langle content \rangle]` commands.

`_enumext_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`.

```
4445 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
4446 {
4447   \_enumext_stop_item_tmp_viii:
4448   \int_incr:N \l\_enumext_item_column_pos_viii_int
4449   \int_gincr:N \g\_enumext_item_count_all_viii_int
4450   \_enumext_item_peek_args_viii:
4451 }
```

(End of definition for `_enumext_start_item_tmp_viii:`.)

`_enumext_item_peek_args_viii:`

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

```
4452 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
4453 {
4454   \peek_meaning:NTF (
4455     { \_enumext_joined_item_viii:w }
4456     { \_enumext_joined_item_viii:w (1) }
4457 }
```

(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`.

```
4458 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
4459 {
4460   \_enumext_starred_joined_item_viii:n {#1}
4461   \peek_meaning_remove:NTF *
4462     { \_enumext_starred_item_viii:w }
4463     { \_enumext_standar_item_viii:w }
4464 }
```

(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 [\langle custom \rangle]` 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]`.

```
4465 \cs_new_protected:Npn \_enumext_standar_item_viii:w
4466 {
4467   \bool_set_false:N \l\_enumext_item_starred_viii_bool
4468   \peek_meaning:NTF [
4469     {
4470       \bool_set_eq:NN
4471         \l\_enumext_wrap_label_viii_bool
4472         \l\_enumext_wrap_label_opt_viii_bool
4473       \_enumext_start_item_viii:w
4474     }
4475     {
4476       \bool_set_true:N \l\_enumext_wrap_label_viii_bool
4477       \legacy_if_set_true:n { @noitemarg }
4478       \_enumext_start_item_viii:w [ \l\_enumext_label_viii_tl ]
4479     }
4480 }
```

```

4479     }
4480 }

```

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

```

\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii: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*[\langle content \rangle]`.

```

4481 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4482 {
4483   \bool_set_true:N \l__enumext_item_starred_viii_bool
4484   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4485   \peek_meaning:NTF [
4486     { \__enumext_starred_item_viii_aux_i:w }
4487     { \__enumext_starred_item_viii_aux_ii:w }
4488   }

```

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`.

```

4489 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4490 {
4491   \tl_clear:N \l__enumext_store_current_label_tl
4492   \tl_if_no_value:nF { #1 }
4493   {
4494     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4495     {
4496       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4497       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4498     }
4499     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4500   }
4501   \__enumext_starred_item_viii_aux_ii:w
4502 }
4503 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4504 {
4505   \legacy_if_set_true:n { @noitemarg }
4506   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4507 }

```

(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 `[\langle content \rangle]` for `\item*[\langle content \rangle]` if present in the `\sequence` and `\prop list` set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

4508 \cs_new_protected:Nn \__enumext_starred_item_exec:
4509 {
4510   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4511   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4512   \__enumext_keyans_store_ref:
4513   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4514   \__enumext_keyans_addto_seq_link:
4515   \int_gincr:N \g__enumext_check_starred_cmd_int
4516   \bool_if:NT \l__enumext_show_answer_bool
4517   {
4518     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4519   }
4520   \bool_if:NT \l__enumext_show_position_bool
4521   {
4522     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4523     {
4524       \group_begin:
4525       \exp_not:N \normalfont
4526       \exp_not:N \footnotesize [ \int_eval:n
4527         {
4528           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4529         }
4530       ]
4531       \group_end:
4532     }

```

```

4533         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_viii_dim \l__enumext_labelsep_viii_dim
4534     }
4535 }

```

(End of definition for `__enumext_starred_item_exec:`.)

12.44.2 Real definition of `\item` in keyans*

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

```

4536 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4537 {
4538     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4539     \legacy_if:nT { @noitemarg }
4540     {
4541         \legacy_if_set_false:n { @noitemarg }
4542         \legacy_if:nT { @nmbrrlist }
4543         {
4544             \bool_if:NT \l__enumext_hyperref_bool
4545             {
4546                 \legacy_if_set_true:n { @hyper@item }
4547             }
4548             \refstepcounter{enumXviii}
4549         }
4550     }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment.

```

4551     \group_begin:
4552     \lrbox{ \l__enumext_item_text_viii_box }
4553     \bool_if:NF \l__enumext_footnotes_key_bool
4554     {
4555         \__enumext_renew_footnote:
4556     }
4557     \bool_if:NT \l__enumext_item_starred_viii_bool
4558     {
4559         \__enumext_starred_item_exec:
4560     }
4561     \group_begin:
4562     \tl_use:N \l__enumext_label_font_style_viii_tl
4563     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4564     {
4565         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4566         { \__enumext_wrapper_label_viii:n {#1} }
4567     }
4568     {
4569         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4570     }
4571     \group_end:
4572     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4573     \tl_use:N \l__enumext_after_list_args_viii_tl
4574     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4575     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4576     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4577     \bool_if:NT \l__enumext_item_starred_viii_bool
4578     {
4579         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4580         \__enumext_keyans_show_item_opt:
4581         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim
4582     }
4583     {
4584         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4585     }
4586 }

```

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

`__enumext_stop_item_viii:` The function `__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4587 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4588 {
4589     \__enumext_endminipage:
4590     \endlrbox

```

```

4591 \group_end:
4592 \box_set_wd:Nn \l__enumext_item_text_viii_box
4593 {
4594   \l__enumext_joined_width_viii_dim
4595   + \l__enumext_labelwidth_viii_dim
4596   + \l__enumext_labelsep_viii_dim
4597 }
4598 \int_set:Nn \hbadness { 10000 }
4599 \box_use_drop:N \l__enumext_item_text_viii_box
4600 \bool_if:NF \l__enumext_footnotes_key_bool
4601 {
4602   \l__enumext_print_footnote:
4603 }
4604 \int_compare:nNnTF
4605 { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4606 {
4607   \par\noindent
4608   \int_zero:N \l__enumext_item_column_pos_viii_int
4609 }
4610 { \hspace{ \l__enumext_columns_sep_viii_dim } }
4611 }

```

(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.

```

4612 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_viii:
4613 {
4614   \int_compare:nNnT
4615   {
4616     \int_mod:nn
4617     { \g__enumext_item_count_all_viii_int }
4618     { \l__enumext_columns_viii_int }
4619   }
4620   =
4621   { 0 }
4622   {
4623     \par
4624     \vspace{ -\l__enumext_itemsep_viii_skip }
4625     \int_gzero:N \g__enumext_item_count_all_viii_int
4626   }
4627 }

```

(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 $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from $\langle \textit{prop list} \rangle$ defined by `save-ans` key.

```

4628 \NewDocumentCommand \getkeyans { m }
4629 {
4630   \exp_args:Ne \l__enumext_getkeyans_aux:n
4631   { \tl_to_str:e { \text_expand:n {#1} } }
4632 }

```

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

`\l__enumext_getkeyans_aux:n`

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

```

4633 \cs_new_protected:Npn \l__enumext_getkeyans_aux:n #1
4634 {
4635   \str_if_in:nnTF {#1} { : }
4636   {
4637     \use:e
4638     {
4639       \cs_set:Npn \exp_not:N \l__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4640       { {##1} {##2} }
4641     }
4642     \exp_after:wN \l__enumext_getkeyans:nn \l__enumext_tmp:w #1 \scan_stop:
4643   }
4644   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4645 }

```

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

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the *prop list*, if it does not exist it will return an error message, then it will fetch the content specified by the second *argument* from *prop list*.

```

4646 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4647 {
4648   \prop_if_exist:cTF { g__enumext_#1_prop }
4649   {
4650     \prop_item:cn { g__enumext_#1_prop }{#2}
4651   }
4652   {
4653     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4654   }
4655 }

```

(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>]`.

```

4656 \keys_define:nn { enumext / print }
4657 {
4658   print* .code:n      = \keys_precompile:neN { enumext / enumext* }
4659                   { \__enumext_filter_save_key:n {#1} }
4660                   \l__enumext_print_keyans_starred_tl, % starred cmd
4661   print* .initial:n   = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4662   print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4663                   { \__enumext_filter_save_key:n {#1} }
4664                   \l__enumext_print_keyans_i_tl,
4665   print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4666   print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4667                   { \__enumext_filter_save_key:n {#1} }
4668                   \l__enumext_print_keyans_ii_tl,
4669   print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4670   print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4671                   { \__enumext_filter_save_key:n {#1} }
4672                   \l__enumext_print_keyans_iii_tl,
4673   print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4674   print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4675                   { \__enumext_filter_save_key:n {#1} }
4676                   \l__enumext_print_keyans_iv_tl,
4677   print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4678   print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4679                   { \__enumext_filter_save_key:n {#1} }
4680                   \l__enumext_print_keyans_vii_tl, % starred nested
4681   print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4682 }

```

- 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`.

```

4683 \NewDocumentCommand \printkeyans { s O{} m }
4684 {
4685   \group_begin:
4686     \tl_use:N \l__enumext_print_keyans_i_tl
4687     \tl_use:N \l__enumext_print_keyans_ii_tl
4688     \tl_use:N \l__enumext_print_keyans_iii_tl

```



```

4689     \tl_use:N \l__enumext_print_keyans_iv_tl
4690     \tl_use:N \l__enumext_print_keyans_vii_tl
4691     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4692   \group_end:
4693 }

```

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

`__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.

```

4694 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4695 {
4696   \seq_if_exist:cTF { g__enumext_#3_seq }
4697   {
4698     \seq_if_empty:cF { g__enumext_#3_seq }
4699     {
4700       %%\seq_show:c { g__enumext_#3_seq }

```

If the starred argument is present we will check that the environment `enumext*` is not saved in the `<sequence>`, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default `<keys>` for the environment `enumext*`, 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>`.

```

4701       \bool_if:nTF {#1}
4702       {
4703         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4704         {
4705           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4706         }
4707         {
4708           \tl_use:N \l__enumext_print_keyans_starred_tl
4709           \begin{enumext*}[#2]
4710             \keys_set:nn { enumext / level-1 }{ base-fix }
4711             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4712             \end{enumext*}
4713           }
4714         }

```

Otherwise it will open the environment `enumext` passing the optional argument to the “first level”, set the key `base-fix` and then map the `<sequence>`.

```

4715         {
4716           \begin{enumext}[#2]
4717             \keys_set:nn { enumext / enumext* }{ base-fix }
4718             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4719             \end{enumext}
4720         }
4721       }
4722     }
4723   {
4724     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4725   }
4726 }

```

(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.

`__enumext_filter_first_level:n`
`__enumext_filter_first_level_key:n`
`__enumext_filter_first_level_pair:nn`

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`.

```

4727 \cs_new:Npn \__enumext_filter_first_level:n #1
4728 {
4729   \use:e
4730   {
4731     \keyval_parse:NNn
4732     \__enumext_filter_first_level_key:n
4733     \__enumext_filter_first_level_pair:nn {#1}
4734   }
4735 }

```

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*`.

```

4736 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4737 {
4738   \str_case:nnF {#1}
4739   {
4740     { resume } {}
4741     { resume* } {}
4742   }
4743   { , { \exp_not:n {#1} } }
4744 }

```

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.

```

4745 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4746 {
4747   \str_case:nnF {#1}
4748   {
4749     { series } {}
4750     { resume } {}
4751     { save-ans } {}
4752   }
4753   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
4754 }

```

(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`.

```

4755 \keys_define:nn { enumext / meta-families }
4756 {
4757   enumext-1 .code:n =
4758   {
4759     \keys_set:ne { enumext / level-1 }
4760     {
4761       \__enumext_filter_first_level:n {#1}
4762     }
4763   } ,
4764   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4765   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4766   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4767   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4768   enumext* .code:n =
4769   {
4770     \keys_set:ne { enumext / enumext* }
4771     {
4772       \__enumext_filter_first_level:n {#1}
4773     }
4774   } ,
4775   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4776   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4777   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4778   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4779   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4780   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4781   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4782   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4783 }

```

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

```

4784 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4785 {
4786   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4787   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4788 }

```

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

```

4789 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4790 {
4791   \seq_clear:N \l__enumext_setkey_tmpa_seq
4792   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4793   \int_set:Nn \l__enumext_setkey_tmpa_int

```

```

4794     {
4795         \seq_count:N \l__enumext_setkey_tmpb_seq
4796     }
4797     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4798     {
4799         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4800         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4801         \seq_set_map_e:Nn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4802         {
4803             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4804         }
4805     }
4806     {
4807         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4808     }
4809     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4810     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4811     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4812     {
4813         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4814     }
4815 }

```

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

```

\__enumext_set_parse:n Internal functions used by the \setenumext command.
\__enumext_set_error:nn
4816 \cs_new_protected:Npn \__enumext_set_parse:n #1
4817 {
4818     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4819     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4820     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4821     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4822     {
4823         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4824         { \tl_trim_spaces:n {#1} }
4825     }
4826     { \__enumext_set_error:nn {#1} { } }
4827 }
4828 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4829 { \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
4830 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4831 {
4832     {enumext,1} = level-1,
4833     {enumext,2} = level-2,
4834     {enumext,3} = level-3,
4835     {enumext,4} = level-4,
4836     {enumext*} = enumext*
4837 }

```

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

```

4838 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
4839 {
4840     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4841     { \msg_error:nn { enumext } { prohibited-unknown } }
4842     {
4843         \bool_if:nTF {#1}
4844         {
4845             \int_step_inline:nn { 4 }
4846             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4847             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4848         }
4849         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }

```

```

4850     }
4851 }

```

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

```

4852 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
4853 {
4854   \tl_set:Nn \l__enumext_meta_path_tl {#1}
4855   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4856   \prop_get:NVNTF
4857     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4858     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4859   {
4860     \msg_error:nnn { enumext } { unknown-set } {#1}
4861     \use_none:nn
4862   }
4863 }
4864 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
4865 {
4866   \bool_lazy_or:nnTF
4867     { \keys_if_exist_p:nn { enumext / #1 } {#2} }
4868     { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
4869     { \msg_error:nnn { enumext } { already-defined } {#2} }
4870   {
4871     \keys_define:nn { enumext / #1 }
4872     {
4873       #2 .meta:n = {#3},
4874       #2 .value_forbidden:n = true
4875     }
4876   }
4877 }
4878 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

12.49 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the `<prop list>` and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function](#).

`\foreachkeyans`

```

\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
\__enumext_foreach_keyans:nn
\__enumext_foreach_add_body:n

```

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

```

4879 \keys_define:nn { enumext / foreach }
4880 {
4881   before .tl_set:N = \l__enumext_foreach_before_tl,
4882   before .value_required:n = true,
4883   after .tl_set:N = \l__enumext_foreach_after_tl,
4884   after .value_required:n = true,
4885   start .int_set:N = \l__enumext_foreach_start_int,
4886   start .value_required:n = true,
4887   stop .int_set:N = \l__enumext_foreach_stop_int,
4888   stop .value_required:n = true,
4889   step .int_set:N = \l__enumext_foreach_step_int,
4890   step .value_required:n = true,
4891   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
4892   wrapper .value_required:n = true,
4893   sep .tl_set:N = \l__enumext_foreach_sep_tl,
4894   sep .value_required:n = true,
4895   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
4896 }
4897 \keys_precompile:nnN { enumext / foreach }
4898 {
4899   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4900 }
4901 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown `<keys>`.

```

4902 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
4903 {
4904   \tl_if_blank:nTF {#2}
4905   {
4906     \msg_error:nnn { enumext } { for-key-unknown } {#1}

```

```

4907     }
4908     {
4909         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
4910     }
4911 }
4912 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
4913 {
4914     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4915 }

```

We create the command.

```

4916 \NewDocumentCommand \foreachkeyans { +0{ } m }
4917 {
4918     \__enumext_foreach_keyans:nn {#1} {#2}
4919 }

```

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

```

4920 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
4921 {
4922     \tl_use:N \g__enumext_foreach_default_keys_tl
4923     \keys_set:nn { enumext / foreach } {#1}
4924     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4925     \prop_if_exist:cF { g__enumext_#2_prop }
4926     {
4927         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
4928     }
4929     \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
4930     {
4931         \int_set:Nn \l__enumext_foreach_stop_int
4932         { \prop_count:c { g__enumext_#2_prop } }
4933     }
4934     \seq_clear:N \l__enumext_foreach_print_seq
4935     \int_step_function:nnnN
4936     { \l__enumext_foreach_start_int }
4937     { \l__enumext_foreach_step_int }
4938     { \l__enumext_foreach_stop_int }
4939     \__enumext_foreach_add_body:n
4940     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4941 }
4942 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
4943 {
4944     \seq_put_right:Ne \l__enumext_foreach_print_seq
4945     {
4946         \exp_not:V \l__enumext_foreach_before_tl
4947         \__enumext_foreach_wrapper:n
4948         {
4949             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4950         }
4951         \exp_not:V \l__enumext_foreach_after_tl
4952     }
4953 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 16.)

12.50 Messages

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

```

4954 \msg_new:nnn { enumext } { package-load }
4955 {
4956     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4957 }
4958 \msg_new:nnn { enumext } { package-not-load }
4959 {
4960     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4961 }
4962 \msg_new:nnn { enumext } { package-load-foot }
4963 {
4964     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4965 }

```

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

```

4966 \msg_new:nnn { enumext } { counters }

```

```

4967 {
4968     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4969     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4970 }

```

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

```

4971 \msg_new:nnn { enumext } { unknown-choice }
4972 {
4973     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4974 }

```

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

```

4975 \msg_new:nnnn { enumext } { anskey-env-error }
4976 {
4977     The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4978     'enumext' ~ package, ~ It~ is~ already~ defined.
4979 }
4980 {
4981     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4982     for ~ the ~ 'save-ans' ~ key.\\
4983 }

```

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

```

4984 \msg_new:nnn { enumext } { store-prop }
4985 {
4986     * ~ Package ~ enumext: ~ Creating ~
4987     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4988 }
4989 \msg_new:nnn { enumext } { store-seq }
4990 {
4991     * ~ Package ~ enumext: ~ Creating ~
4992     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4993 }
4994 \msg_new:nnn { enumext } { store-int }
4995 {
4996     * ~ Package ~ enumext: ~ Creating ~
4997     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4998 }
4999 \msg_new:nnn { enumext } { prop-seq-int-hook }
5000 {
5001     * ~ Package ~ enumext: ~ Elements ~ in ~
5002     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5003     * ~ Package ~ enumext: ~ Elements ~ in ~
5004     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5005     * ~ Package ~ enumext: ~ Value ~ off ~
5006     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5007 }
5008 \msg_new:nnn { enumext } { item-answer-hook }
5009 {
5010     * ~ Package ~ enumext: ~ Value ~ off ~
5011     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5012     * ~ Package ~ enumext: ~ Value ~ off ~
5013     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5014     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5015 }

```

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

```

5016 \msg_new:nnn { enumext } { invalid-key }
5017 {
5018     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5019 }
5020 \msg_new:nnn { enumext } { unknown-key-family }
5021 {
5022     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5023 }

```

Messages used in length calculation.

```

5024 \msg_new:nnn { enumext } { width-negative }
5025 {
5026     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5027     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ #2pt.
5028 }
5029 \msg_new:nnn { enumext } { width-zero }

```

```

5030 {
5031     Invalid ~ '#1=#2' ~ \msg_line_context:.\
5032     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5033 }

```

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

```

5034 \msg_new:nnn { enumext } { list-lengths }
5035 {
5036     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5037     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5038     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5039     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5040     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5041     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5042     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5043     \__enumext_show_length:nnn { skip } { topsep } {#1}
5044     \__enumext_show_length:nnn { skip } { parsep } {#1}
5045     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5046     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5047     *****
5048 }

```

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

```

5049 \msg_new:nnn { enumext } { list-lengths-not-nested }
5050 {
5051     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5052     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5053     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5054     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5055     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5056     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5057     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5058     \__enumext_show_length:nnn { skip } { topsep } {#1}
5059     \__enumext_show_length:nnn { skip } { parsep } {#1}
5060     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5061     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5062     *****
5063 }

```

Messages used by `ref` key.

```

5064 \msg_new:nnn { enumext } { key-ref-empty }
5065 {
5066     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5067 }

```

Messages used by `save-ans` key.

```

5068 \msg_new:nnn { enumext } { save-ans-empty }
5069 {
5070     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5071 }
5072 \msg_new:nnn { enumext } { save-ans-log }
5073 {
5074     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5075 }
5076 \msg_new:nnn { enumext } { save-ans-log-hook }
5077 {
5078     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5079 }
5080 \msg_new:nnn { enumext } { save-ans-hook }
5081 {
5082     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5083 }

```

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

```

5084 \msg_new:nnn { enumext } { need-save-ans }
5085 {
5086     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5087 }
5088 \msg_new:nnn { enumext } { items-same-answer }
5089 {
5090     *****\\
5091     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5092     for ~ \c_left_brace_str #2 \c_right_brace_str\\

```



```

5093     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5094     'OK', ~ all ~ items ~ with ~ answer.\\
5095     *****
5096   }
5097   \msg_new:nnn { enumext } { item-greater-answer }
5098   {
5099     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5100     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5101     Items ~ > ~ Answers.
5102   }
5103   \msg_new:nnn { enumext } { item-less-answer }
5104   {
5105     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5106     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5107     Items ~ < ~ Answers.
5108   }

```

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

```

5109   \msg_new:nnn { enumext } { missing-starred }
5110   {
5111     Missing ~ '\c_backslash_str #1*' ~ #2.
5112   }
5113   \msg_new:nnn { enumext } { many-starred }
5114   {
5115     Many ~ '\c_backslash_str #1*' ~ #2.
5116   }

```

Messages used by `\printkeyans*` command.

```

5117   \msg_new:nnn { enumext } { print-starred }
5118   {
5119     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5120     #2 ~ environment ~ \msg_line_context:.
5121   }

```

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

```

5122   \msg_new:nnn { enumext } { list-too-deep }
5123   {
5124     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context::~ \\
5125     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5126   }

```

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

```

5127   \msg_new:nnn { enumext } { anskey-unnumber-item }
5128   {
5129     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5130   }
5131   \msg_new:nnn { enumext } { anskey-already-stored }
5132   {
5133     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5134   }
5135   \msg_new:nnn { enumext } { anskey-empty-arg }
5136   {
5137     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5138   }
5139   \msg_new:nnn { enumext } { anskey-wrong-place }
5140   {
5141     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context::~ \\
5142     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5143   }
5144   \msg_new:nnn { enumext } { anskey-nested }
5145   {
5146     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5147   }
5148   \msg_new:nnn { enumext } { anskey-math-mode }
5149   {
5150     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5151   }
5152   \msg_new:nnn { enumext } { anskey-env-wrong }
5153   {
5154     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5155   }
5156   \msg_new:nnn { enumext } { anspic-wrong-place }
5157   {

```

```

5158     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5159     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5160 }
5161 \msg_new:nnn { enumext } { command-wrong-place }
5162 {
5163     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5164     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5165 }
5166 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5167 {
5168     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5169     'anskey*' ~ and ~ is ~ being ~ ignored.
5170 }
5171 {
5172     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5173     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5174 }
5175 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5176 {
5177     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5178     'anskey*' ~ and ~ is ~ being ~ ignored.
5179 }
5180 {
5181     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5182     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5183 }
5184 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5185 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5186 {
5187     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5188     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5189 }
5190 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5191 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5192 {
5193     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5194     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5195 }

```

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

```

5196 \msg_new:nnn { enumext } { keyans-nested }
5197 {
5198     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5199 }
5200 \msg_new:nnn { enumext } { keyans-wrong-level }
5201 {
5202     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5203     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5204 }
5205 \msg_new:nnn { enumext } { wrong-place }
5206 {
5207     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
5208     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5209 }
5210 \msg_new:nnn { enumext } { keyanspic-nested }
5211 {
5212     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5213 }
5214 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5215 {
5216     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5217     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5218 }
5219 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5220 {
5221     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5222 }
5223 \msg_new:nnnn { enumext } { keyans-unknown-key }
5224 {
5225     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5226     '\l_enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5227 }

```

```

5228 {
5229     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5230     ~ have ~ a ~ key ~ called ~ '#1'.\\
5231     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5232 }
5233 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5234 {
5235     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5236     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5237 }
5238 {
5239     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5240     ~ have ~ a ~ key ~ called ~ '#1'.\\
5241     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5242 }

```

Message used by unknown $\langle keys \rangle$ in enumext*. environment.

```

5243 \msg_new:nnnn { enumext } { starred-unknown-key }
5244 {
5245     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5246     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5247 }
5248 {
5249     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5250     ~ have ~ a ~ key ~ called ~ '#1'.\\
5251     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5252 }
5253 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5254 {
5255     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5256     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5257 }
5258 {
5259     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5260     ~ have ~ a ~ key ~ called ~ '#1'.\\
5261     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5262 }

```

Message used by unknown $\langle keys \rangle$ in enumext environment.

```

5263 \msg_new:nnnn { enumext } { standar-unknown-key }
5264 {
5265     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5266     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5267 }
5268 {
5269     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5270     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5271     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5272 }
5273 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5274 {
5275     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5276     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5277 }
5278 {
5279     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5280     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5281     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5282 }

```

Message used by unknown $\langle keys \rangle$ in \foreachkeyans.

```

5283 \msg_new:nnnn { enumext } { for-key-unknown }
5284 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5285 {
5286     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5287     Check~that~you~have~spelled~the~key~name~correctly.
5288 }
5289 \msg_new:nnnn { enumext } { for-key-value-unknown }
5290 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5291 {
5292     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5293     Check~that~you~have~spelled~the~key~name~correctly.
5294 }

```

Messages used by `\getkeyans` command.

```
5295 \msg_new:nnn { enumext } { undefined-storage-anskey }
5296 {
5297   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5298 }
```

Messages used by `\miniright` command.

```
5299 \msg_new:nnn { enumext } { missing-miniright }
5300 {
5301   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
5302   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5303 }
5304 \msg_new:nnn { enumext } { wrong-miniright-place }
5305 {
5306   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5307   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5308 }
5309 \msg_new:nnn { enumext } { wrong-miniright-use }
5310 {
5311   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.\
5312   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5313 }
5314 \msg_new:nnn { enumext } { wrong-miniright-starred }
5315 {
5316   Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5317 }
5318 \msg_new:nnn { enumext } { many-miniright-used }
5319 {
5320   Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5321 }
```

Messages used by `\setenumextmeta` command.

```
5322 \msg_new:nnn { enumext } { unknown-set }
5323 {
5324   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5325 }
5326 \msg_new:nnn { enumext } { already-defined }
5327 {
5328   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5329 }
5330 \msg_new:nnn { enumext } { prohibited-unknown }
5331 {
5332   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5333 }
```

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

```
5334 \msg_new:nnn { enumext } { nested }
5335 {
5336   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5337 }
5338 \msg_new:nnn { enumext } { nested-horizontal }
5339 {
5340   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
5341 }
5342 \msg_new:nnn { enumext } { item-joined }
5343 {
5344   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5345 }
5346 \msg_new:nnn { enumext } { item-joined-columns }
5347 {
5348   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5349 }
```

12.51 Finish package

Finish package implementation.

```
5350 \file_input_stop:
5351 </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>	219
<code>\+</code>	211
<code>\-</code>	211
<code>\\</code>	227, 2730, 3731, 4968, 4977, 4982, 5002, 5004, 5011, 5013, 5026, 5031, 5036, 5051, 5090, 5092, 5094, 5099, 5100, 5105, 5106, 5124, 5141, 5158, 5163, 5172, 5181, 5187, 5193, 5202, 5207, 5216, 5230, 5240, 5250, 5260, 5270, 5280, 5286, 5292, 5301, 5306, 5311
A	
<code>above</code>	<u>1545</u>
<code>above*</code>	<u>1545</u>
<code>\addvspace</code>	1140, 1169, 1212, 1215, 1389, 1452, 1458, 1495, 1503, 1531, 3544, 3548, 3684, 3700, 4058, 4072, 4113, 4127
<code>after</code>	<u>978</u>
<code>align</code>	<u>527</u>
<code>\Alpha</code>	36, <u>41</u>
<code>\Alph</code>	479, 594, 639, 707, 4677
<code>\alph</code>	36, <u>41</u>
<code>\alph</code>	480, 592, 4669
<code>\anskey</code>	12, 74, 76, <u>2548</u>
<code>anskey*</code>	13, <u>2658</u>
<code>\anspic</code>	15, <u>100</u> , <u>3709</u>
<code>\anspic*</code>	68
<code>\arabic</code>	30, <u>36</u>
<code>\arabic</code>	478, 591, 638, 4661, 4665, 4681
B	
<code>base-fix</code>	<u>837</u>
<code>\baselineskip</code>	<u>50</u>
<code>\baselineskip</code>	854, 865
<code>before</code>	<u>978</u>
<code>before*</code>	<u>978</u>
<code>below</code>	<u>1545</u>
<code>below*</code>	<u>1545</u>
bool commands:	
<code>\bool_gset_false:N</code>	350, 351, 352, 2834, 2836, 4074, 4078, 4129
<code>\bool_gset_true:N</code>	258, 268, 1081, 2038, 2044, 4050, 4075, 4105, 4130
<code>\bool_if:NTF</code>	418, 430, 447, 1474, 1567, 1581, 1594, 1605, 1616, 1627, 1638, 1649, 1698, 1715, 1720, 1728, 1755, 1793, 1798, 1805, 1809, 1831, 1836, 1844, 1851, 1882, 1890, 1983, 2181, 2191, 2270, 2294, 2301, 2325, 2423, 2445, 2485, 2498, 2502, 2552, 2571, 2595, 2649, 2660, 2749, 2786, 2850, 2883, 2898, 2973, 2984, 2988, 3007, 3020, 3062, 3096, 3131, 3265, 3327, 3337, 3369, 3374, 3477, 3525, 3540, 3554, 3613, 3668, 3682, 3690, 3711, 4046, 4055, 4059, 4101, 4110, 4114, 4203, 4213, 4296, 4301, 4310, 4314, 4329, 4358, 4414, 4516, 4520, 4544, 4553, 4557, 4563, 4577, 4600
<code>\bool_if:nTF</code>	1504, 1532, 3118, 3247, 3285, 3732, 4701, 4843
<code>\bool_if_p:N</code>	277, 292, 850, 851, 861, 862, 1862, 1863, 1871, 1872, 1996, 2022, 2035, 2036, 2041, 2042, 2358, 2368, 2380, 2395, 2396, 2430, 2471, 2472, 2772, 2960, 2961, 2998, 2999, 3450, 3452, 3463, 3739, 3740
<code>\bool_lazy_all:nTF</code>	275, 290, 1994, 2020, 2356, 2365, 2378, 2393, 3448, 3461
<code>\bool_lazy_and:nnTF</code>	254, 264, 849, 860, 1467, 1861, 1870, 2034, 2040, 2429, 2436, 2470, 2613, 2625, 2771, 2777, 2959
<code>\bool_lazy_or:nnTF</code>	1923, 1930, 2997, 3738, 4866
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 95, 100, 101, 106, 107, 110, 135, 136, 144, 145, 150, 152, 153, 167, 179, 181
<code>\bool_not_p:n</code>	255, 265, 2367, 2431, 2437, 2773, 2778, 3451, 3464
<code>\bool_set_eq:NN</code>	3071, 3227, 4246, 4470
<code>\bool_set_false:N</code>	427, 871, 1968, 1969, 2001, 2006, 2010, 2014, 2027, 2713, 3425, 3571, 3621, 3705, 3770, 3788, 4171, 4198, 4243, 4420, 4467
<code>\bool_set_true:N</code>	282, 283, 297, 298, 409, 413, 520, 886, 1551, 1556, 1818, 1940, 1941, 2213, 2221, 2714, 3065, 3067, 3099, 3101, 3223, 3235, 3349, 3424, 3457, 3470, 3496, 3618, 3645, 4035, 4090, 4170, 4252, 4259, 4260, 4304, 4418, 4476, 4483, 4484
box commands:	
<code>\box_dp:N</code>	1398, 1399, 1402, 1409, 1422, 1430, 1436, 1444, 3800
<code>\box_ht:N</code>	1212, 1215, 1226, 1227, 1238, 1240, 1255, 1258, 1266, 1267, 1278, 1280, 1295, 1298, 1305, 1306, 1317, 1319, 1334, 1337, 1365, 1389
<code>\box_new:N</code>	70, 174, 180
<code>\box_set_wd:Nn</code>	4350, 4592
<code>\box_use_drop:N</code>	4070, 4125, 4357, 4599
<code>\box_wd:N</code>	486
C	
<code>\c</code>	219, 220, 744, 746, 758, 760
<code>\catcode</code>	2730
<code>\cB</code>	220
<code>\cE</code>	220
<code>\centering</code>	1506, 1534, 3826, 4063, 4118
<code>check-ans</code>	<u>1960</u>
Document class:	
<code>article</code>	43
clist commands:	
<code>\clist_const:Nn</code>	186
<code>\clist_map_function:nN</code>	3813
<code>\clist_map_inline:Nn</code>	526, 792, 977, 992, 1073, 1561
<code>\clist_map_inline:nn</code>	49, 60, 78, 85, 97, 109, 138, 161, 185, 554, 574, 846, 891, 912, 1087, 1667, 1907, 1974, 2160, 2178, 2210, 2353, 2892, 3152, 3164, 3204, 3314, 3317, 3344, 3356, 3359, 3379, 4819
<code>\columnbreak</code>	74
<code>\columnbreak</code>	2433
<code>columns</code>	<u>1057</u>
<code>columns-sep</code>	<u>1057</u>
<code>\columnsep</code>	96
<code>\columnsep</code>	3520, 3666
<code>\columnseprule</code>	96
<code>\columnseprule</code>	3523, 3667
Commands provide by enumext:	
<code>\anskey</code>	28, 64, 65, 70, 71, 73–77, 83, 85, 95, 110, 118, 119, 127
<code>\anspic*</code>	28, 29, 68, 71, 83, 84, 100, 101, 103, 118, 119

\anspic	71, 100–102, 127
\foreachkeyans	123, 129
\getkeyans	71, 118, 130
\item*	28, 29, 68, 71, 83, 84, 86, 87, 90, 111, 116, 118, 119
\item	86, 90, 105, 110, 111, 115
\miniright	27, 47, 55, 56, 96, 97, 130
\printkeyans*	119
\printkeyans	28, 71, 119
\setenumextmeta	122, 130
\setenumext	28, 119, 121, 122, 125
Counters defined by enumext :	
enumXiii	26, 36
enumXii	26, 36
enumXiv	26, 36
enumXi	26, 36
enumXviii	26, 36
enumXvii	26, 36, 111
enumXvi	26, 36
enumXv	26, 36
cs commands:	
\cs_generate_variant:Nn	191, 192, 488, 504, 750, 766, 2262, 2267, 2343, 2666, 3304, 3815, 4878
\cs_if_exist:NTF	458
\cs_if_free:NTF	2617, 2629
\cs_new:Nn	205
\cs_new:Npn	223, 1668, 1677, 1685, 2225, 2234, 2242, 4727, 4736, 4745
\cs_new_eq:NN	377, 378, 379, 383, 384, 432, 433, 436, 437
\cs_new_protected:Nn	215, 247, 273, 306, 336, 342, 348, 354, 360, 368, 386, 404, 615, 678, 730, 847, 993, 997, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 1041, 1045, 1049, 1053, 1088, 1100, 1124, 1142, 1153, 1171, 1197, 1218, 1343, 1378, 1391, 1413, 1448, 1454, 1562, 1576, 1590, 1601, 1612, 1623, 1634, 1645, 1726, 1829, 1842, 1859, 1880, 1908, 1913, 1938, 1979, 1989, 2032, 2047, 2054, 2063, 2068, 2073, 2078, 2087, 2092, 2097, 2268, 2292, 2299, 2323, 2330, 2344, 2569, 2588, 2604, 2667, 2703, 2734, 2769, 2811, 2832, 2840, 2881, 2896, 2924, 2957, 2993, 3005, 3018, 3104, 3114, 3125, 3243, 3259, 3400, 3417, 3446, 3475, 3482, 3504, 3534, 3552, 3594, 3611, 3635, 3652, 3677, 3688, 3728, 3772, 3786, 3811, 3816, 3832, 3836, 3855, 3865, 3896, 4025, 4044, 4080, 4099, 4157, 4185, 4192, 4201, 4211, 4228, 4369, 4406, 4433, 4439, 4452, 4508, 4612
\cs_new_protected:Npn	193, 197, 201, 229, 440, 456, 473, 483, 489, 595, 640, 712, 737, 751, 1484, 1517, 1694, 1713, 1783, 1816, 1918, 2102, 2179, 2189, 2211, 2219, 2254, 2263, 2419, 2482, 2496, 2534, 2538, 2658, 2689, 2693, 2724, 2860, 2934, 2978, 3058, 3077, 3165, 3169, 3183, 3187, 3205, 3209, 3219, 3231, 3273, 3307, 3347, 3428, 3631, 3781, 3927, 3976, 4174, 4234, 4241, 4257, 4265, 4270, 4282, 4426, 4458, 4465, 4481, 4489, 4503, 4633, 4646, 4694, 4816, 4828, 4852, 4864, 4902, 4912, 4920, 4942
\cs_new_protected_nopar:Nn	4221, 4345, 4445, 4587
\cs_new_protected_nopar:Npn	4288, 4536
\cs_set:Npn	2354, 2391, 4639
\cs_set_eq:NN	4147, 4148, 4290, 4396, 4397, 4538
\cs_set_protected:Nn	917, 933, 945, 957
\cs_set_protected:Npn	45, 54, 71, 79, 92, 98, 131, 157, 165, 505, 527, 559, 575, 622, 767, 793, 837, 873, 896, 969, 978, 1057, 1074, 1545, 1656, 1899, 1960, 2119, 2161, 2197, 2346, 2885, 3141, 3157, 3197, 3305, 3345
\cs_to_str:N	475, 498
\cs_undefine:N	2606, 2607, 2608, 2609
D	
\d	211
\DeclareDocumentEnvironment	390
dim commands:	
\dim_abs:n	3278, 3283
\dim_add:Nn	3791, 3890, 3921
\dim_compare:nNnTF	919, 935, 947, 959, 1230, 1242, 1270, 1282, 1309, 1321, 1486, 1519, 3275, 3280, 3286, 3292, 3294, 3296, 3487, 3509, 3639, 3656, 3783, 3867, 3883, 3898, 3914, 4027, 4082
\dim_compare:nTF	2455, 2799, 3406, 3600
\dim_gset_eq:NN	4036, 4091
\dim_gzero:N	2838, 4077, 4132
\dim_new:N	67, 74, 75, 76, 94, 140, 173, 175, 176, 182
\dim_set:Nn	486, 887, 3094, 3278, 3283, 3285, 3288, 3289, 3293, 3295, 3298, 3299, 3301, 3402, 3490, 3512, 3596, 3641, 3658, 3818, 3869, 3876, 3900, 3907, 3962, 4011, 4029, 4084, 4284
\dim_set_eq:NN	582, 629, 700, 704, 3009, 3010, 3022, 3023, 3089, 3316, 3358, 3520, 3666, 3969, 3972, 3973, 4018, 4021, 4022, 4275
\dim_sub:Nn	3411, 3605, 3885, 3916
\dim_use:N	920, 928, 1487, 1500, 2333, 2336, 2341, 3109, 3111, 3408, 3413, 3488, 3493, 3494, 3500, 3510, 3514, 3515, 3517
\dim_zero:N	3350, 3523, 3667, 3792, 3793, 3794
\dim_zero_new:N	455
\c_zero_dim	922, 936, 948, 960, 1487, 1519, 2457, 2801, 3275, 3280, 3286, 3293, 3408, 3488, 3510, 3602, 3639, 3656, 3867, 3883, 3898, 3914, 4027, 4082
\dimeval	2126
E	
\end	1497, 1526, 2296, 2327, 3539, 3563, 3681, 3699, 4048, 4071, 4103, 4126, 4703, 4712, 4719
\endgroup	2730
\endlist	378
\endlrbox	4348, 4590
\endminipage	384
enumext	5, <u>3380</u>
enumext internal commands:	
\l__enumext_ref_the_count_tl	38
\l__enumext_resume_name_tl	60
__enumext_add_meta_key:nnn	123, <u>4830</u> , 4846, 4847, 4849, 4852
__enumext_add_pre_parsep:	48, 1098, <u>1100</u> , 1100
__enumext_after_args_exec:	46, <u>993</u> , 1005, 3393
__enumext_after_args_exec_v:	<u>1009</u> , 1021, 3587
__enumext_after_args_exec_vii:	<u>1025</u> , 1049
__enumext_after_args_exec_viii:	1053
__enumext_after_env:nn	80, 81, 83, 97, 106, 113, <u>197</u> , 197, 2744, 3574, 4053, 4108, 4383
__enumext_after_hyperref:	34, 402, <u>404</u> , 404
__enumext_after_list:	97, 114, 3398, <u>3552</u> , 3552
\l__enumext_after_list_args_v_tl	1023
\l__enumext_after_list_args_vii_tl	1051, 4339
\l__enumext_after_list_args_viii_tl	1055, 4573
__enumext_after_list_v:	3592, <u>3635</u> , 3688
__enumext_after_list_vii:	109, 4155, <u>4192</u> , 4192
__enumext_after_list_viii:	4404, <u>4439</u> , 4439


```

\__enumext_after_stop_list: .. 46, 97, 993, 1001,
    3568
\__enumext_after_stop_list_v: 1009, 1017, 3706
\l__enumext_after_stop_list_v_tl ..... 1019
\__enumext_after_stop_list_vii: .. 109, 1025,
    1041, 4195
\l__enumext_after_stop_list_vii_tl ... 1043
\__enumext_after_stop_list_viii: . 1045, 4442
\l__enumext_after_stop_list_viii_tl ... 1047
\l__enumext_align_label_vii_str .. 4331, 4335
\l__enumext_align_label_viii_str . 4565, 4569
\l__enumext_align_label_X_str ..... 165
\c__enumext_all_envs_clist .. 186, 526, 792, 977,
    992, 1073, 1561
\c__enumext_all_families_seq .. 121, 4784, 4810
\l__enumext_anskey_env_bool 31, 79, 34, 283, 298,
    2660
\__enumext_anskey_env_clean_vars: . 82, 2765,
    2769, 2832
\__enumext_anskey_env_define_keys: 79, 2658,
    2667, 2738
\__enumext_anskey_env_exec: 80, 2663, 2734, 2734
\__enumext_anskey_env_make:n 64, 79, 1943, 2658,
    2658, 2666
\__enumext_anskey_env_reset_keys: 80, 81, 2703,
    2766
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2658
\__enumext_anskey_env_save_keys: .. 81, 2746,
    2769, 2769
\__enumext_anskey_env_store: .. 82, 2762, 2769,
    2811
\__enumext_anskey_env_unknown:n 80, 2686, 2689
\__enumext_anskey_env_unknown:nn . 2691, 2693
\l__enumext_anskey_level_int .. 28, 2590, 2591
\__enumext_anskey_safe_inner: . 77, 2563, 2569,
    2588
\__enumext_anskey_safe_inner:n ..... 77
\__enumext_anskey_safe_outer: . 77, 2550, 2569,
    2569
\__enumext_anskey_show_wrap_arg:n . 75, 2482,
    2482, 2500, 2515
\__enumext_anskey_show_wrap_left:n 76, 2427,
    2496, 2496
\__enumext_anskey_unknown:n 76, 2518, 2532, 2534
\__enumext_anskey_unknown:nn . 2518, 2536, 2538
\__enumext_anskey_wrapper:n ..... 2123, 2494
\__enumext_at_begin_document:n 33, 34, 193, 193,
    375, 381
\l__enumext_base_line_fix_bool . 841, 851, 862,
    871
\__enumext_before_args_exec: .. 46, 95, 109, 993,
    993, 3485
\__enumext_before_args_exec_v: 1009, 1009, 3638
\__enumext_before_args_exec_vii: . 1025, 1025,
    4189
\__enumext_before_args_exec_viii: 1029, 4436
\__enumext_before_env:nn 79, 197, 201, 2611, 2623,
    2635, 2736
\__enumext_before_keys_exec: 46, 993, 997, 3390
\__enumext_before_keys_exec_v: 1009, 1013, 3584
\__enumext_before_keys_exec_vii ..... 1025
\__enumext_before_keys_exec_vii: . 1033, 4143
\__enumext_before_keys_exec_viii: 1037, 4392
\__enumext_before_list: ... 95, 3384, 3482, 3482
\__enumext_before_list_v: ... 3579, 3635, 3635
\__enumext_before_list_vii: ... 109, 4138, 4185,
    4185
\__enumext_before_list_viii: . 114, 4388, 4433,
    4433
\l__enumext_before_no_starred_key_v_tl 1015
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1035
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1039
\l__enumext_before_starred_key_v_tl ... 1011
\l__enumext_before_starred_key_vii_tl . 1027
\l__enumext_before_starred_key_viii_tl 1031
\__enumext_calc_hspace:NNNNNNN 91, 3273, 3273,
    3304, 3309, 3351
\__enumext_check_ans_active: . 66, 95, 109, 1979,
    1979, 3486, 4188
\g__enumext_check_ans_item_tl ..... 85
\g__enumext_check_ans_key_bool 67, 68, 144, 350,
    2038, 2044, 2850
\l__enumext_check_ans_key_bool 67, 1964, 1969,
    2035, 2041
\__enumext_check_ans_key_hook: 67, 97, 109, 2032,
    2032, 3569, 4196
\__enumext_check_ans_level: 66, 1979, 1985, 1989
\__enumext_check_ans_log: 67, 68, 82, 2078, 2078,
    2854
\__enumext_check_ans_log_msg_greater: 2078,
    2084, 2097
\__enumext_check_ans_log_msg_less: 2078, 2082,
    2087
\__enumext_check_ans_log_msg_same_ok: 2078,
    2083, 2092
\__enumext_check_ans_msg_greater: 2054, 2060,
    2073
\__enumext_check_ans_msg_less: 2054, 2058, 2063
\__enumext_check_ans_msg_same_ok: 2054, 2059,
    2068
\__enumext_check_ans_show: .. 67, 82, 2054, 2054,
    2852
\l__enumext_check_answers_bool 64, 66, 77, 86, 87,
    144, 1941, 1968, 1983, 2270, 2294, 2301, 2325, 2552,
    2749, 2973, 3062, 3096, 4301
\__enumext_check_starred_cmd:n 32, 68, 85, 2102,
    2102, 3590, 3767, 4402
\g__enumext_check_starred_cmd_int 144, 2105,
    2111, 2116, 3241, 3737, 4515
\l__enumext_check_start_line_env_tl . 32, 144,
    313, 321, 329, 2108, 2114, 2117
\l__enumext_columns_sep_v_dim 3656, 3658, 3666
\l__enumext_columns_sep_vii_dim .. 3867, 3869,
    3878, 3890, 3966, 4367
\l__enumext_columns_sep_viii_dim . 3898, 3900,
    3909, 3921, 4015, 4610
\l__enumext_columns_v_int 1372, 1522, 3654, 3662,
    3674, 3679
\l__enumext_columns_vii_int .. 3872, 3875, 3879,
    3888, 3930, 3934, 3937, 3943, 3949, 3953, 4362, 4373
\l__enumext_columns_viii_int . 3903, 3906, 3910,
    3919, 3979, 3983, 3986, 3992, 3998, 4002, 4605, 4618
\l__enumext_counter_i_tl ..... 45, 465
\l__enumext_counter_ii_tl ..... 45, 466
\l__enumext_counter_iii_tl ..... 45, 467
\l__enumext_counter_iv_tl ..... 45, 468

```



```

\c__enumext_counter_style_tl . . . . . 30, 50, 217
\g__enumext_counter_styles_tl . 26, 36, 67, 476,
    494
\l__enumext_counter_v_tl . . . . . 45, 469, 720
\l__enumext_counter_vi_tl . . . . . 45, 470
\l__enumext_counter_vii_tl . . . . . 45, 471, 650
\l__enumext_counter_viii_tl . . . . . 45, 472, 667
\l__enumext_current_widest_dim 26, 67, 500, 583,
    630, 701, 705
\__enumext_def_meta_key:nnn . . . 123, 4830, 4858,
    4864, 4878
\__enumext_default_item:n . . . 3058, 3058, 3122
\__enumext_define_counters:Nn 26, 456, 456, 465,
    466, 467, 468, 469, 470, 471, 472
\__enumext_endminipage: . 34, 381, 384, 398, 3828,
    4347, 4589
\g__enumext_envir_name_tl 32, 34, 284, 299, 358,
    1911, 1916, 1926, 2066, 2071, 2076, 2090, 2095, 2100
\l__enumext_envir_name_tl . 31, 32, 34, 253, 263,
    312, 320, 328, 5226, 5229, 5236, 5239, 5246, 5249,
    5256, 5259, 5265, 5269, 5275, 5279, 5336, 5340
\__enumext_execute_after_env: 33, 64, 67, 68, 78,
    82, 2840, 2840, 3574, 4383
\__enumext_fake_item: . . . . . 917, 917, 3336
\l__enumext_fake_item_indent_v_dim 936, 941
\l__enumext_fake_item_indent_v_tl 938, 3224,
    3228, 3236
\l__enumext_fake_item_indent_vii_dim 948, 953
\l__enumext_fake_item_indent_vii_tl 950, 4343
\l__enumext_fake_item_indent_viii_dim . 960,
    965, 4581
\l__enumext_fake_item_indent_viii_tl . . 962,
    4579, 4584
\l__enumext_fake_item_indent_X_tl . . . . . 98
\__enumext_fake_item_vii: . . . . 917, 945, 3368
\__enumext_fake_item_viii: . . . . 917, 957, 3373
\__enumext_filter_first_level:n . . 120, 4727,
    4727, 4761, 4772
\__enumext_filter_first_level_key:n 121, 4727,
    4732, 4736
\__enumext_filter_first_level_pair:nn . 121,
    4727, 4733, 4745
\__enumext_filter_save_key:n . . 70, 2186, 2194,
    2217, 2223, 2225, 2225, 4659, 4663, 4667, 4671, 4675,
    4679
\__enumext_filter_save_key_key:n . . 71, 2225,
    2230, 2234
\__enumext_filter_save_key_pair:nn 71, 2225,
    2231, 2242
\__enumext_filter_series:n 59, 1668, 1668, 1706,
    1718, 1723
\__enumext_filter_series_key:n 59, 1668, 1673,
    1677
\__enumext_filter_series_pair:nn . . 59, 1668,
    1674, 1685
\g__enumext_footnote_arg_seq . 162, 3838, 3851,
    3861
\g__enumext_footnote_int . 162, 3845, 3848, 3850,
    3852
\g__enumext_footnote_int_seq . 162, 3839, 3852,
    3857, 3860
\__enumext_footnotes_key_bool . . . . . 34
\l__enumext_footnotes_key_bool 29, 35, 112, 152,
    413, 418, 427, 4310, 4358, 4553, 4600
\__enumext_footnotetext:nn . . . 3832, 3832, 3862
\__enumext_foreach_add_body:n 124, 4879, 4939,
    4942
\l__enumext_foreach_after_tl . . . . . 4883, 4951
\l__enumext_foreach_before_tl . . . . 4881, 4946
\g__enumext_foreach_default_keys_tl 123, 124,
    4901, 4922
\__enumext_foreach_keyans:nn . 124, 4879, 4918,
    4920
\l__enumext_foreach_name_prop_tl . 124, 4924,
    4949
\l__enumext_foreach_print_seq 124, 4934, 4940,
    4944
\l__enumext_foreach_sep_tl . . . . . 4893, 4940
\l__enumext_foreach_start_int . . . . 4885, 4936
\l__enumext_foreach_step_int . . . . . 4889, 4937
\l__enumext_foreach_stop_int . 4887, 4929, 4931,
    4938
\__enumext_foreach_wrapper:n . . . . . 4891, 4947
\__enumext_getkeyans:nn . . 119, 4642, 4646, 4646
\__enumext_getkeyans_aux:n 118, 4630, 4633, 4633
\l__enumext_hyperref_bool . 29, 34, 35, 152, 409,
    430, 447, 2472, 2961, 4296, 4544
\__enumext_hypertarget:nn 35, 404, 432, 436, 452
\__enumext_if_is_int:n . . . . . 209
\__enumext_if_is_int:nTF . . . . . 209, 739, 753
\__enumext_internal_mini_page: 34, 94, 108, 386,
    386, 3419, 4159
\__enumext_is_not_nested: 26, 31, 94, 108, 247, 247,
    3420, 4160
\__enumext_is_on_first_level: . 26, 31, 94, 108,
    247, 273, 3426, 4172
\g__enumext_item_anskey_int 77, 85, 144, 345, 372,
    373, 2051, 2421, 2975
\__enumext_item_answer_diff: . . 67, 68, 82, 2047,
    2047, 2847
\g__enumext_item_answer_diff_int . 67, 68, 144,
    346, 2049, 2056, 2080
\l__enumext_item_column_pos_vii_int 110, 3937,
    3943, 3949, 3953, 3960, 4224, 4362, 4365
\l__enumext_item_column_pos_viii_int . . 115,
    3986, 3992, 3998, 4002, 4009, 4448, 4605, 4608
\l__enumext_item_column_pos_X_int . . . . . 165
\g__enumext_item_count_all_vii_int 110, 3961,
    4225, 4373, 4380
\g__enumext_item_count_all_viii_int 115, 4010,
    4449, 4617, 4625
\g__enumext_item_count_all_X_int . . . . . 165
\g__enumext_item_number_bool . . . . . 144
\l__enumext_item_number_bool 66, 150, 2001, 2006,
    2010, 2014, 2027, 2595, 2649, 3065, 3099, 4304
\g__enumext_item_number_int . . 66, 144, 344, 371,
    373, 2000, 2005, 2009, 2013, 2026, 2051, 3064, 3098,
    4303
\__enumext_item_peek_args_vii: 110, 4226, 4228,
    4228
\__enumext_item_peek_args_viii: . . 115, 4450,
    4452, 4452
\__enumext_item_star_exec: . 87, 3077, 3104, 3133
\l__enumext_item_starred_vii_bool 4243, 4259,
    4314
\l__enumext_item_starred_viii_bool 4467, 4483,
    4557, 4577
\l__enumext_item_starred_X_bool . . . . . 165

```

__enumext_item_std:w 33, 86, 87, 90, 102, 375, 379,
 3068, 3074, 3102, 3224, 3228, 3236, 3804
 \g__enumext_item_symbol_aux_tl . 87, 128, 3082,
 3085, 3110, 3138
 \g__enumext_item_symbol_aux_vii_tl 4267, 4316,
 4319, 4323, 4325
 \g__enumext_item_symbol_aux_X_tl 165
 \l__enumext_item_symbol_sep_vii_dim . . 4276,
 4284, 4322, 4324
 \l__enumext_item_symbol_vii_tl 4319
 \l__enumext_item_text_vii_box 4309, 4350, 4357
 \l__enumext_item_text_viii_box 4552, 4592, 4599
 \l__enumext_item_text_X_box 165
 \l__enumext_item_width_vii_dim . . . 3876, 3885,
 3964, 3972, 3973
 \l__enumext_item_width_viii_dim . . 3907, 3916,
 4013, 4021, 4022
 \l__enumext_item_width_X_dim 165
 \l__enumext_itemindent_X_dim 71
 \l__enumext_itemsep_i_skip . . 1224, 1231, 1234,
 1236, 1243, 1247, 1250, 1252, 1363, 1368
 \l__enumext_itemsep_ii_skip . . 1264, 1271, 1274,
 1276, 1283, 1287, 1290, 1292
 \l__enumext_itemsep_iii_skip . 1303, 1310, 1313,
 1315, 1322, 1326, 1329, 1331
 \l__enumext_itemsep_vii_skip 4379
 \l__enumext_itemsep_viii_skip 4624
 \l__enumext_joined_item_aux_vii_int . . 3958,
 3959, 3960, 3961, 3967
 \l__enumext_joined_item_aux_viii_int . 4007,
 4008, 4009, 4010, 4016
 \l__enumext_joined_item_aux_X_int 165
 __enumext_joined_item_vii:w . . 110, 4231, 4232,
 4234, 4234
 \l__enumext_joined_item_vii_int . . 3929, 3930,
 3933, 3935, 3941, 3946, 3951, 3956, 3958, 3964
 __enumext_joined_item_viii:w . 115, 4455, 4456,
 4458, 4458
 \l__enumext_joined_item_viii_int . 3978, 3979,
 3982, 3984, 3990, 3995, 4000, 4005, 4007, 4013
 \l__enumext_joined_item_X_int 165
 \l__enumext_joined_width_vii_dim . 3962, 3969,
 3972, 4340, 4352
 \l__enumext_joined_width_viii_dim 4011, 4018,
 4021, 4574, 4594
 \l__enumext_joined_width_X_dim 165
 __enumext_keyans_addto_prop:n 83, 2860, 2860,
 3238, 3734
 __enumext_keyans_addto_seq:n . 84, 2934, 2934,
 3240, 3736
 __enumext_keyans_addto_seq_link: 2934, 2955,
 2957, 4514
 __enumext_keyans_anspic_code:nnn . 100, 101,
 3725, 3728, 3728
 __enumext_keyans_default_item:n . . 90, 3219,
 3219, 3255
 \l__enumext_keyans_env_bool 34, 3451, 3464, 3618,
 3705
 __enumext_keyans_fake_item: . . 917, 933, 3326
 \l__enumext_keyans_level_h_int . . 28, 660, 687,
 2579, 2641, 2912, 4166, 4408, 4409
 \l__enumext_keyans_level_int . . 28, 1478, 2575,
 2637, 2907, 3617, 3622, 3719
 __enumext_keyans_make_label: 37, 91, 3243, 3259,
 3324
 __enumext_keyans_mini_right_cmd:n 56, 1480,
 1517, 1517
 __enumext_keyans_mini_set_vskip: 53
 __enumext_keyans_minipage_add_space: . . 53,
 1378, 1378, 3647
 __enumext_keyans_minipage_set_skip: . 1343,
 1343, 1380
 __enumext_keyans_multi_addvspace: 1142, 1153,
 3671
 __enumext_keyans_multi_set_vskip: 49, 1142,
 1142, 1155
 __enumext_keyans_multicols_start: 3635, 3650,
 3652
 __enumext_keyans_multicols_stop: 1521, 3635,
 3677, 3703
 __enumext_keyans_name_and_start: 26, 32, 306,
 306, 3619, 3779, 4413
 __enumext_keyans_parse_keys:n 3578, 3631, 3631
 \l__enumext_keyans_pic_above_int . 139, 3819,
 3820, 3822
 \l__enumext_keyans_pic_above_skip . 102, 139,
 3758, 3798
 __enumext_keyans_pic_arg_two: 102, 3756, 3786,
 3786
 \l__enumext_keyans_pic_below_int . 139, 3819,
 3820, 3823
 \l__enumext_keyans_pic_body_seq 100, 101, 103,
 139, 3723, 3763, 3827
 __enumext_keyans_pic_do:n 102, 3763, 3765, 3811,
 3811, 3815
 \l__enumext_keyans_pic_level_int . . 28, 1462,
 2583, 2645, 2863, 2902, 2937, 3025, 3774, 3775
 __enumext_keyans_pic_row:n 102, 103, 3813, 3816,
 3816
 __enumext_keyans_pic_safe_exec: . 101, 3752,
 3772, 3772
 __enumext_keyans_pic_skip_abs:N . 102, 3781,
 3781, 3797
 \l__enumext_keyans_pic_width_dim . 139, 3818,
 3825
 __enumext_keyans_redefine_item: . . 90, 3243,
 3243, 3323
 __enumext_keyans_ref: 41, 712, 730, 3325
 __enumext_keyans_ref:n 40, 709, 712, 712
 __enumext_keyans_safe_exec: . 3577, 3611, 3611
 __enumext_keyans_set_item_width: . 98, 3586,
 3594, 3594
 __enumext_keyans_show_ans: . . 2978, 2986, 3005
 __enumext_keyans_show_item_opt: . 2978, 2993,
 3236, 3748, 4580
 __enumext_keyans_show_left:n . 90, 2978, 2978,
 3234, 3743
 __enumext_keyans_show_pos: . . 2978, 2990, 3018
 __enumext_keyans_starred_item:n . . 90, 3231,
 3231, 3251
 __enumext_keyans_store_ref: . . 83, 2881, 2881,
 3239, 3735, 4512
 __enumext_keyans_store_ref_aux_i: 84, 2881,
 2893, 2896
 __enumext_keyans_store_ref_aux_ii: 84, 2881,
 2922, 2924
 __enumext_keyans_unknown_keys:n . 3157, 3161,
 3165

`__enumext_keyans_unknown_keys:nn` [3157](#), [3167](#), [3169](#)
`__enumext_keyans_wrapper_opt:n` [2129](#), [3001](#)
`\l__enumext_label_copy_i_tl` [2387](#), [2900](#), [2905](#), [2910](#), [2915](#)
`\l__enumext_label_copy_v_tl` [2910](#)
`\l__enumext_label_copy_vi_tl` [2905](#)
`\l__enumext_label_copy_vii_tl` [2363](#), [2374](#), [2403](#), [2900](#)
`\l__enumext_label_copy_viii_tl` [2915](#)
`\l__enumext_label_copy_X_tl` [154](#)
`\l__enumext_label_fill_left_v_tl` [3263](#)
`\l__enumext_label_fill_left_X_tl` [98](#)
`\l__enumext_label_fill_right_v_tl` [3270](#)
`\l__enumext_label_fill_right_X_tl` [98](#)
`\l__enumext_label_font_style_v_tl` [3264](#), [3747](#)
`\l__enumext_label_font_style_vii_tl` [4328](#)
`\l__enumext_label_font_style_viii_tl` [4562](#)
`\l__enumext_label_i_tl` [575](#)
`\l__enumext_label_ii_tl` [575](#)
`\l__enumext_label_iii_tl` [575](#)
`\l__enumext_label_iv_tl` [575](#)
`__enumext_label_style:Nnn` [26](#), [36](#), [489](#), [489](#), [504](#), [580](#), [627](#), [698](#), [702](#)
`\l__enumext_label_v_tl` [83](#), [84](#), [695](#), [2868](#), [2942](#), [3012](#), [3052](#), [3233](#), [3237](#), [3581](#), [3742](#), [3744](#)
`\l__enumext_label_vi_tl` [83](#), [84](#), [695](#), [2865](#), [2939](#), [3742](#), [3744](#), [3748](#)
`\l__enumext_label_vii_tl` [622](#), [4254](#), [4279](#), [4286](#)
`\l__enumext_label_viii_tl` [622](#), [4478](#), [4506](#), [4510](#)
`\l__enumext_label_width_by_box` [67](#), [485](#), [486](#)
`__enumext_label_width_by_box:Nn` [36](#), [483](#), [483](#), [488](#), [500](#), [763](#)
`\l__enumext_labelsep_i_dim` [3010](#), [3015](#), [3023](#), [3055](#), [4518](#), [4533](#)
`\l__enumext_labelsep_v_dim` [3661](#)
`\l__enumext_labelsep_vii_dim` [2487](#), [3010](#), [3023](#), [3871](#), [3881](#), [3965](#), [4277](#), [4338](#), [4354](#)
`\l__enumext_labelsep_viii_dim` [3902](#), [3912](#), [4014](#), [4572](#), [4581](#), [4596](#)
`\l__enumext_labelwidth_i_dim` [3009](#), [3015](#), [3022](#), [3055](#), [4518](#), [4533](#)
`\l__enumext_labelwidth_v_dim` [3661](#)
`\l__enumext_labelwidth_vii_dim` [2487](#), [3009](#), [3022](#), [3871](#), [3880](#), [3965](#), [4331](#), [4335](#), [4353](#)
`\l__enumext_labelwidth_viii_dim` [3902](#), [3911](#), [4014](#), [4565](#), [4569](#), [4595](#)
`\l__enumext_leftmargin_tmp_v_bool` [102](#), [3788](#)
`\l__enumext_leftmargin_tmp_X_bool` [71](#)
`\l__enumext_leftmargin_tmp_X_dim` [71](#)
`\l__enumext_leftmargin_X_dim` [71](#)
`__enumext_level:` [205](#), [205](#), [604](#), [607](#), [608](#), [617](#), [619](#), [920](#), [924](#), [928](#), [995](#), [999](#), [1003](#), [1007](#), [1090](#), [1092](#), [1094](#), [1096](#), [1129](#), [1131](#), [1133](#), [1135](#), [1140](#), [1175](#), [1181](#), [1186](#), [1188](#), [1191](#), [1194](#), [1207](#), [1210](#), [1487](#), [1491](#), [1500](#), [1565](#), [1567](#), [1569](#), [1572](#), [1579](#), [1581](#), [1583](#), [1586](#), [2181](#), [2183](#), [2185](#), [2213](#), [2214](#), [2216](#), [2272](#), [2280](#), [2284](#), [2288](#), [2491](#), [2492](#), [3067](#), [3068](#), [3072](#), [3073](#), [3074](#), [3082](#), [3090](#), [3091](#), [3094](#), [3101](#), [3102](#), [3106](#), [3109](#), [3111](#), [3129](#), [3130](#), [3131](#), [3134](#), [3137](#), [3387](#), [3389](#), [3408](#), [3413](#), [3457](#), [3470](#), [3477](#), [3488](#), [3490](#), [3493](#), [3494](#), [3496](#), [3500](#), [3507](#), [3510](#), [3512](#), [3514](#), [3515](#), [3516](#), [3517](#), [3520](#), [3525](#), [3531](#), [3537](#), [3540](#), [3544](#), [3548](#), [3554](#)
`\l__enumext_level_h_int` [108](#), [28](#), [256](#), [279](#), [293](#), [643](#), [680](#), [1469](#), [1997](#), [2017](#), [2382](#), [2615](#), [2627](#), [3465](#), [4161](#), [4162](#)
`\l__enumext_level_int` [94](#), [28](#), [207](#), [266](#), [278](#), [294](#), [388](#), [1102](#), [1220](#), [1468](#), [1991](#), [2023](#), [2359](#), [2369](#), [2375](#), [2381](#), [2388](#), [2397](#), [2402](#), [2614](#), [2626](#), [2842](#), [3339](#), [3421](#), [3422](#), [3433](#), [3441](#), [3455](#), [3468](#), [3521](#), [3626](#), [3715](#), [4205](#), [4215](#), [4421](#), [5266](#), [5270](#), [5276](#), [5280](#)
`__enumext_list_arg_two_i:` [3305](#)
`__enumext_list_arg_two_ii:` [3305](#)
`__enumext_list_arg_two_iii:` [3305](#)
`__enumext_list_arg_two_iv:` [3305](#)
`__enumext_list_arg_two_v:` [90](#), [3305](#), [3583](#), [3789](#)
`__enumext_list_arg_two_vii:` [3345](#), [4142](#)
`__enumext_list_arg_two_viii:` [3345](#), [4391](#)
`\l__enumext_listoffset_v_dim` [3602](#), [3607](#), [3663](#)
`\l__enumext_listparindent_vii_dim` [4341](#)
`\l__enumext_listparindent_viii_dim` [4575](#)
`__enumext_log_answer_vars:` [33](#), [360](#), [368](#), [2849](#)
`__enumext_log_global_vars:` [33](#), [360](#), [360](#), [2848](#)
`__enumext_make_label` [3114](#)
`__enumext_make_label:` [37](#), [88](#), [3125](#), [3334](#)
`\l__enumext_mark_answer_sym_tl` [73](#), [2135](#), [2338](#), [2504](#), [3027](#), [3040](#), [4522](#)
`\l__enumext_mark_position_str` [128](#), [2139](#), [2140](#), [2166](#), [2167](#), [2336](#)
`\l__enumext_mark_ref_sym_tl` [2152](#), [2477](#), [2969](#)
`\l__enumext_meta_path_tl` [124](#), [4854](#), [4855](#), [4857](#), [4858](#)
`\c__enumext_meta_paths_prop` [122](#), [4830](#)
`__enumext_mini_addvspace_vii:` [55](#), [1448](#), [1448](#), [4039](#)
`__enumext_mini_addvspace_viii:` [55](#), [1448](#), [1454](#), [4094](#)
`__enumext_mini_env*` [386](#)
`__enumext_mini_right_cmd:n` [55](#), [56](#), [1482](#), [1484](#), [1484](#)
`__enumext_mini_set_vskip_vii:` [54](#), [1391](#), [1391](#), [1450](#)
`__enumext_mini_set_vskip_viii:` [54](#), [1391](#), [1413](#), [1456](#)
`__enumext_minipage:w` [34](#), [381](#), [383](#), [392](#), [3825](#), [4340](#), [4574](#)
`\l__enumext_minipage_active_v_bool` [3645](#), [3668](#), [3682](#), [3690](#)
`\g__enumext_minipage_active_vii_bool` [106](#), [4050](#), [4055](#), [4074](#)
`\l__enumext_minipage_active_vii_bool` [4035](#), [4046](#)
`\g__enumext_minipage_active_viii_bool` [4105](#), [4110](#), [4129](#)
`\l__enumext_minipage_active_viii_bool` [4090](#), [4101](#)
`\g__enumext_minipage_active_X_bool` [165](#)
`\l__enumext_minipage_active_X_bool` [86](#)
`__enumext_minipage_add_space:` [50](#), [96](#), [1171](#), [1197](#), [3498](#)
`\g__enumext_minipage_after_skip` [86](#), [1395](#), [1407](#), [4072](#), [4127](#)
`\l__enumext_minipage_after_skip` [50](#), [97](#), [86](#), [1184](#), [1224](#), [1226](#), [1231](#), [1234](#), [1238](#), [1243](#), [1247](#), [1250](#), [1254](#), [1266](#), [1271](#), [1274](#), [1278](#), [1283](#), [1287](#), [1290](#), [1294](#), [1305](#), [1310](#), [1313](#), [1317](#), [1322](#), [1326](#), [1329](#), [1333](#), [1345](#), [1359](#), [1361](#), [1365](#), [1368](#), [1415](#), [1428](#), [1442](#), [1493](#), [1495](#), [1524](#), [3700](#)

`\g__enumext_minipage_center_vii_bool` . 4059, 4075
`\g__enumext_minipage_center_viii_bool` 4114, 4130
`\g__enumext_minipage_center_X_bool` ... 165
`\l__enumext_minipage_hsep_v_dim` 3643
`\l__enumext_minipage_hsep_vii_dim` 4033
`\l__enumext_minipage_hsep_viii_dim` . . . 4088
`\l__enumext_minipage_left_skip` 86, 1346, 1393, 1398, 1402, 1416, 1420, 1434, 1452, 1458
`\l__enumext_minipage_left_v_dim` . . 3641, 3648
`\l__enumext_minipage_left_vii_dim` 4029, 4041
`\l__enumext_minipage_left_viii_dim` 4084, 4096
`\l__enumext_minipage_left_X_dim` 86
`\g__enumext_minipage_right_skip` 86, 1394, 1399, 1403, 4058, 4113
`\l__enumext_minipage_right_skip` . 50, 86, 1173, 1179, 1184, 1186, 1188, 1347, 1348, 1354, 1359, 1375, 1417, 1424, 1438, 1503, 1531
`\l__enumext_minipage_right_v_dim` . 1519, 1528, 3639, 3643
`\g__enumext_minipage_right_vii_dim` 106, 4037, 4057, 4077
`\l__enumext_minipage_right_vii_dim` 106, 4027, 4032, 4038
`\g__enumext_minipage_right_viii_dim` . . 4092, 4112, 4132
`\l__enumext_minipage_right_viii_dim` . . 4082, 4087, 4093
`\g__enumext_minipage_right_X_dim` 165
`\g__enumext_minipage_right_X_skip` 165
`__enumext_minipage_set_skip`: . 50, 1171, 1171, 1199
`\g__enumext_minipage_stat_int` 96, 86, 1508, 1536, 3497, 3556, 3561, 3646, 3692, 3697
`\l__enumext_miniright_code_vii_box` 4066, 4070
`\g__enumext_miniright_code_vii_tl` 107, 4061, 4068, 4076
`\l__enumext_miniright_code_viii_box` . . 4121, 4125
`\g__enumext_miniright_code_viii_tl` 4116, 4123, 4131
`\l__enumext_miniright_code_X_box` 165
`__enumext_multi_addvspace`: . 49, 96, 1124, 1124, 3528
`__enumext_multi_set_vskip`: 48, 1088, 1088, 1126
`\l__enumext_multicols_above_ii_skip` . . 1107
`\l__enumext_multicols_above_iii_skip` . . 1113
`\l__enumext_multicols_above_iv_skip` . . 1119
`\l__enumext_multicols_above_v_skip` 1144, 1158, 1169
`\l__enumext_multicols_above_X_skip` 79
`\l__enumext_multicols_below_ii_skip` . . 1227, 1236, 1240, 1252, 1257
`\l__enumext_multicols_below_iii_skip` . 1267, 1276, 1280, 1292, 1297
`\l__enumext_multicols_below_iv_skip` . . 1306, 1315, 1319, 1331, 1336
`\l__enumext_multicols_below_v_skip` 1148, 1162, 3684
`\l__enumext_multicols_below_X_skip` 79
`\g__enumext_multicols_right_X_skip` 79
`__enumext_multicols_start`: 96, 3502, 3504, 3504
`__enumext_multicols_stop`: 97, 1489, 3534, 3534, 3566
`__enumext_nested_base_line_fix`: . 43, 94, 109, 837, 847, 3437, 4182
`__enumext_newlabel:nn` 29, 35, 74, 440, 440, 2413, 2928
`\l__enumext_newlabel_arg_one_tl` 29, 35, 74, 84, 154, 2406, 2414, 2476, 2917, 2929, 2967
`\l__enumext_newlabel_arg_two_tl` 29, 35, 73, 154, 2362, 2372, 2385, 2400, 2415, 2904, 2909, 2914, 2930
`__enumext_parse_foreach_keys:n` . . 4879, 4895, 4912
`__enumext_parse_foreach_keys:nn` . 4879, 4902, 4914
`__enumext_parse_keys:n` 43, 60, 3383, 3428, 3428
`__enumext_parse_keys_vii:n` . 43, 60, 4137, 4174, 4174
`__enumext_parse_keys_viii:n` . 4387, 4426, 4426
`__enumext_parse_save_key:n` 70, 2206, 2211, 2211
`__enumext_parse_save_key_vii:n` 70, 2201, 2211, 2219
`__enumext_parse_series:n` 60, 94, 109, 1694, 1694, 3436, 4180
`__enumext_parse_store_keys:n` 94
`\l__enumext_parsep_i_skip` 1105, 1107
`\l__enumext_parsep_ii_skip` 1111, 1113
`\l__enumext_parsep_iii_skip` 1117, 1119
`\l__enumext_parsep_vii_skip` 4342
`\l__enumext_parsep_viii_skip` 4576
`\l__enumext_partopsep_v_skip` . 1160, 1164, 1356, 1387
`\l__enumext_partopsep_viii_skip` 1426
`__enumext_phantomsection`: 35, 404, 433, 437, 453
`__enumext_pre_itemsep_skip`: 50, 51, 1189, 1218, 1218
`__enumext_print_footnote`: . . . 3832, 3855, 4360, 4602
`__enumext_print_keyans_box:NN` 73, 2330, 2330, 2343, 2487, 2490, 3014, 3054, 4518, 4533
`\l__enumext_print_keyans_i_tl` . . . 4664, 4686
`\l__enumext_print_keyans_ii_tl` . . 4668, 4687
`\l__enumext_print_keyans_iii_tl` . . 4672, 4688
`\l__enumext_print_keyans_iv_tl` . . 4676, 4689
`\l__enumext_print_keyans_starred_tl` 119, 120, 128, 4660, 4708
`\l__enumext_print_keyans_vii_tl` 119, 4680, 4690
`\l__enumext_print_keyans_X_tl` 128
`__enumext_printkeyans:nnn` 119, 120, 4691, 4694, 4694
`__enumext_redefine_item`: . 88, 3114, 3114, 3333
`\l__enumext_ref_key_arg_tl` 38, 50, 220, 597, 598, 611, 642, 645, 656, 662, 673, 714, 715, 726
`\l__enumext_ref_the_count_tl` . 38, 50, 604, 607, 610, 650, 652, 655, 667, 669, 672, 720, 722, 725
`__enumext_regex_counter_style`: . . 30, 38, 215, 215, 605, 651, 668, 721
`__enumext_register_counter_style:Nn` . . 473, 473, 478, 479, 480, 481, 482
`__enumext_remove_extra_parsep_vii`: . . 4152, 4369, 4369
`__enumext_remove_extra_parsep_viii`: . 4401, 4612, 4612
`__enumext_renew_footnote`: . . . 3832, 3836, 4312, 4555
`\l__enumext_renew_the_count_v_tl` 723, 732, 734
`\l__enumext_renew_the_count_vii_tl` 653, 682, 684

`\l__enumext_renew_the_count_viii_tl` 670, 689, 691
`\l__enumext_renew_the_count_X_tl` 50
`__enumext_rescan_anskey_env:n` . . 80, 82, 2724, 2819, 2827
`__enumext_reset_global_bool:` . . 336, 339, 348
`__enumext_reset_global_int:` . . . 336, 338, 342
`__enumext_reset_global_tl:` 336, 340, 354
`__enumext_reset_global_vars:` . 33, 82, 336, 336, 2857
`\l__enumext_resume_active_bool` 60, 62, 61, 1698, 1818
`__enumext_resume_counter:` . . 61, 62, 1816, 1822, 1829
`__enumext_resume_counter:n` . 60, 62, 1787, 1792, 1816, 1816, 1886, 1894
`__enumext_resume_counter_save_ans:` . . 62, 63, 1816, 1827, 1859
`__enumext_resume_counter_series:` . 62, 1816, 1825, 1842
`\g__enumext_resume_int` . . . 61, 1739, 1833, 1834
`__enumext_resume_last:n` . . 60, 1694, 1700, 1713
`\l__enumext_resume_name_tl` 61, 1735, 1743, 1746, 1762, 1770, 1773, 1819, 1820, 1848, 1855
`__enumext_resume_save_counter:` . . 60, 97, 109, 1726, 1726, 3572, 4199
`__enumext_resume_series:n` . 61, 1662, 1783, 1783
`__enumext_resume_starred:` . 63, 1663, 1880, 1880
`\g__enumext_resume_vii_int` 61, 1766, 1838, 1839
`\l__enumext_rightmargin_vii_dim` . . 3883, 3887, 3892
`\l__enumext_rightmargin_viii_dim` . 3914, 3918, 3923
`__enumext_safe_exec:` . . 34, 94, 3382, 3417, 3417
`__enumext_safe_exec_vii:` . 34, 4136, 4157, 4157
`__enumext_safe_exec_viii:` . . . 4386, 4406, 4406
`\l__enumext_series_name_tl` 62
`\l__enumext_series_str` . . 60, 94, 109, 1660, 1696, 1704, 1705, 1707, 1709, 1730, 1733, 1737, 1757, 1760, 1764, 3432, 4178
`__enumext_set_error:nn` 4816, 4826, 4828
`__enumext_set_item_width:` . 94, 3392, 3400, 3400
`__enumext_set_parse:n` 4800, 4816, 4816
`\l__enumext_setkey_tmpa_int` . . . 119, 4793, 4797
`\l__enumext_setkey_tmpa_seq` . . 119, 4791, 4801, 4807, 4809, 4811, 4823
`\l__enumext_setkey_tmpa_tl` 119, 4799, 4803
`\l__enumext_setkey_tmpb_seq` . . 119, 4792, 4795, 4799, 4800
`\l__enumext_setkey_tmpb_tl` 119, 4818, 4820, 4821
`\l__enumext_show_answer_bool` . 2146, 2170, 2498, 2984, 2998, 3739, 4516
`__enumext_show_length:nnn` . . 45, 223, 223, 5037, 5038, 5039, 5040, 5041, 5042, 5043, 5044, 5045, 5046, 5052, 5053, 5054, 5055, 5056, 5057, 5058, 5059, 5060, 5061
`\l__enumext_show_position_bool` . . . 2149, 2173, 2502, 2988, 2999, 3740, 4520
`\g__enumext_standar_bool` 31, 94, 34, 255, 258, 277, 351, 1728, 1793, 1805, 1831, 1844, 1882, 2022, 2036, 2367, 2380, 2395, 3452
`\l__enumext_standar_bool` . 94, 97, 34, 2368, 3424, 3571, 4171
`\l__enumext_standar_first_bool` 31, 94, 34, 282, 850, 1715, 1862, 1924, 1931
`__enumext_standar_item_vii:w` . 110, 111, 4239, 4241, 4241
`__enumext_standar_item_viii:w` 115, 4463, 4465, 4465
`__enumext_standar_ref:` 39, 595, 615, 3335
`__enumext_standar_ref:n` 38, 587, 595, 595
`\g__enumext_standar_series_tl` . 61, 1717, 1718, 1884, 1887
`__enumext_standar_unknown_keys:n` 3197, 3201, 3205
`__enumext_standar_unknown_keys:nn` 3197, 3207, 3209
`\g__enumext_starred_bool` 31, 108, 34, 265, 268, 292, 352, 1755, 1798, 1809, 1836, 1851, 1890, 1996, 2042, 2358, 2898, 4078
`\l__enumext_starred_bool` 108, 109, 34, 1474, 2396, 2431, 2437, 2485, 2773, 2778, 3007, 3020, 3425, 4170, 4198, 4414, 4418
`__enumext_starred_columns_set_vii:` . . 3865, 3865, 4145
`__enumext_starred_columns_set_viii:` . 3865, 3896, 4394
`\l__enumext_starred_first_bool` 31, 108, 34, 297, 861, 1720, 1871, 1924, 1931
`__enumext_starred_item:nn` . . . 3077, 3077, 3120
`__enumext_starred_item_exec:` . 116, 4508, 4508, 4559
`__enumext_starred_item_vii:w` . 110, 111, 4238, 4257, 4257
`__enumext_starred_item_vii_aux_i:w` . . 4257, 4262, 4265
`__enumext_starred_item_vii_aux_ii:w` . 4257, 4263, 4268, 4270
`__enumext_starred_item_vii_aux_iii:w` 4257, 4273, 4282
`__enumext_starred_item_viii:w` 115, 116, 4462, 4481, 4481
`__enumext_starred_item_viii_aux_i:w` . . 116, 4481, 4486, 4489
`__enumext_starred_item_viii_aux_ii:w` . 116, 4481, 4487, 4501, 4503
`__enumext_starred_joined_item_vii:n` 105, 110, 3927, 3927, 4236
`__enumext_starred_joined_item_viii:n` . 105, 115, 3927, 3976, 4460
`__enumext_starred_ref:` 40, 640, 678, 3365
`__enumext_starred_ref:n` 39, 634, 640, 640
`\g__enumext_starred_series_tl` . 61, 1722, 1723, 1892, 1895
`__enumext_starred_unknown_keys:n` 3179, 3181, 3183
`__enumext_starred_unknown_keys:nn` 3179, 3185, 3187
`__enumext_start_from:NNn` 41, 737, 737, 750, 772, 778
`\l__enumext_start_i_int` 1834, 1846, 1865
`__enumext_start_item_tmp_vii:` 108, 4148, 4221, 4221
`__enumext_start_item_tmp_viii:` . . 113, 4397, 4445, 4445
`__enumext_start_item_vii:w` . . . 111, 4249, 4254, 4279, 4286, 4288, 4288
`__enumext_start_item_viii:w` . . 115, 4473, 4478, 4506, 4536, 4536


```

\g__enumext_start_line_tl    31, 34, 285, 300, 357,
    2066, 2071, 2076, 2090, 2095, 2100
\__enumext_start_list:nn    .. 33, 91, 102, 375, 377,
    3386, 3580, 3753, 4140, 4389
\__enumext_start_mini_vii:  109, 4025, 4025, 4190
\__enumext_start_mini_viii: .. 114, 4080, 4080,
    4437
\__enumext_start_save_ans_msg: 64, 1908, 1908,
    1933
\__enumext_start_store_level: . 95, 3385, 3446,
    3446
\__enumext_start_store_level_vii: 110, 4139,
    4201, 4201
\l__enumext_start_vii_int    ... 1839, 1853, 1874
\l__enumext_start_X_int      ..... 98
\__enumext_stop_item_tmp_vii: .. 108, 110, 111,
    4147, 4151, 4223, 4290
\__enumext_stop_item_tmp_viii: 113, 115, 4396,
    4400, 4447, 4538
\__enumext_stop_item_vii:    111, 113, 4290, 4345,
    4345
\__enumext_stop_item_viii:   117, 4538, 4587, 4587
\__enumext_stop_list: .. 33, 375, 378, 3396, 3591,
    3766, 4153, 4403
\__enumext_stop_mini_vii:    106, 109, 4025, 4044,
    4194
\__enumext_stop_mini_viii:   114, 4080, 4099, 4441
\__enumext_stop_save_ans_msg: . 64, 1908, 1913,
    2846
\__enumext_stop_store_level: .. 95, 3397, 3446,
    3475
\__enumext_stop_store_level_vii: . 110, 4154,
    4201, 4211
\l__enumext_store_active_bool 28, 64, 110, 1863,
    1872, 1940, 2571, 3450, 3463, 3613, 3621, 3711, 3770,
    4203, 4213, 4420
\__enumext_store_active_keys:n 69, 70, 94, 2179,
    2179, 3443
\__enumext_store_active_keys_vii:n 69, 70, 109,
    2179, 2189, 4181
\__enumext_store_addto_prop:n 71, 83, 2254, 2254,
    2262, 2422, 2879, 4511
\__enumext_store_addto_seq:n 71, 85, 2263, 2263,
    2267, 2274, 2288, 2296, 2305, 2319, 2327, 2480, 2972
\l__enumext_store_anskey_arg_tl 28, 74, 75, 110,
    2428, 2433, 2435, 2440, 2447, 2450, 2460, 2465, 2468,
    2474, 2480
\__enumext_store_anskey_code:n 74, 77, 82, 2419,
    2419, 2564, 2817, 2825
\l__enumext_store_anskey_env_tl .. 28, 80, 110,
    2747, 2751, 2757, 2819, 2827
\l__enumext_store_anskey_opt_tl .. 28, 81, 110,
    2748, 2775, 2781, 2788, 2794, 2804, 2814, 2823
\__enumext_store_anskey_safe_outer: .... 77
\g__enumext_store_columns_break_bool . 2671,
    2772, 2834
\l__enumext_store_columns_break_bool . 2430,
    2520
\l__enumext_store_current_label_tl 28, 83-85,
    116, 110, 2862, 2865, 2868, 2875, 2877, 2879, 2936,
    2939, 2942, 2948, 2953, 2963, 2972, 4491, 4496, 4497,
    4510, 4511, 4513
\l__enumext_store_current_label_tmp_tl . 28,
    110, 3233, 3237
\l__enumext_store_current_opt_arg_tl 28, 116,
    110, 2982, 2995, 3001, 4499
\__enumext_store_internal_ref: .. 73, 74, 2344,
    2344, 2425
\g__enumext_store_item_join_int .. 2674, 2779,
    2783, 2835
\l__enumext_store_item_join_int .. 2438, 2442,
    2523
\g__enumext_store_item_star_bool . 2676, 2786,
    2836
\l__enumext_store_item_star_bool . 2445, 2525
\g__enumext_store_item_symbol_sep_dim 2681,
    2801, 2806, 2838
\l__enumext_store_item_symbol_sep_dim 2457,
    2462, 2530
\g__enumext_store_item_symbol_tl . 2679, 2792,
    2796, 2837
\l__enumext_store_item_symbol_tl . 2448, 2452,
    2528
\l__enumext_store_keyans_item_opt_sep_-
    tl .... 2132, 2873, 2875, 2946, 2950, 4494, 4496
\__enumext_store_level_close: . 72, 2268, 2292,
    3479
\__enumext_store_level_close_vii: . 72, 2299,
    2323, 4217
\__enumext_store_level_open: 72, 95, 2268, 2268,
    3458, 3471
\__enumext_store_level_open_vii: .. 72, 2299,
    2299, 4207
\g__enumext_store_name_tl 28, 64, 110, 356, 363,
    364, 365, 366, 1916, 1942, 2065, 2070, 2075, 2089,
    2094, 2099, 2844
\l__enumext_store_name_tl 28, 64, 66, 110, 1749,
    1752, 1776, 1779, 1867, 1876, 1911, 1920, 1921, 1942,
    1943, 1944, 1946, 1947, 1949, 1951, 1952, 1954, 1956,
    1957, 1981, 2256, 2258, 2265, 2408, 2409, 2510, 2753,
    2919, 2920, 3033, 3046, 4528
\l__enumext_store_ref_key_bool 74, 2155, 2423,
    2471, 2883, 2960
\l__enumext_store_save_key_vii_bool .. 2191,
    2221
\l__enumext_store_save_key_vii_tl 2193, 2194,
    2222, 2223, 2303, 2311, 2315, 2319
\l__enumext_store_save_key_X_bool .. 69, 128
\l__enumext_store_save_key_X_tl .. 69, 70, 128
\l__enumext_store_upper_level_X_bool .. 128
\__enumext_storing_exec: 64, 79, 1918, 1934, 1938
\__enumext_storing_set:n .. 64, 1903, 1918, 1918
\l__enumext_the_counter_v_tl ..... 722
\l__enumext_the_counter_vii_tl ..... 652
\l__enumext_the_counter_viii_tl ..... 669
\l__enumext_the_counter_X_tl ..... 50
\__enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 85, 92, 97,
    98, 109, 131, 138, 157, 161, 165, 185, 837, 846, 1656,
    1667, 1899, 1907, 1960, 1978, 2119, 2160, 2161, 2178,
    2197, 2210, 2346, 2353, 2354, 2375, 2388, 2391, 2402,
    2885, 2892, 3157, 3164, 3197, 3204, 3305, 3344, 3345,
    3379
\__enumext_tmp:nn 505, 526, 527, 558, 559, 574, 767,
    792, 873, 895, 896, 916, 969, 977, 978, 992, 1057, 1073,
    1074, 1087, 1545, 1561, 3141, 3156
\__enumext_tmp:nnn 575, 591, 592, 593, 594, 622, 638,
    639
\__enumext_tmp:nnnnn 793, 818, 821, 824, 826, 828,
    831, 834
\__enumext_tmp:w ..... 4639, 4642

```

<code>\l__enumext_tmpa_vii_int</code>	3875, 3878, 3887, 3918
<code>\l__enumext_tmpa_viii_int</code>	3906, 3909
<code>\l__enumext_tmpa_X_dim</code>	165
<code>\l__enumext_tmpa_X_int</code>	165
<code>\l__enumext_topsep_v_skip</code>	1146, 1150, 1350, 3769, 3801
<code>\l__enumext_topsep_vii_skip</code>	1396, 1405, 1409
<code>\l__enumext_topsep_viii_skip</code>	1418, 1440, 1444
<code>__enumext_undefine_anskey_env:</code>	78, 82, 2604, 2604, 2855
<code>__enumext_unskip_unkern:</code>	31, 229, 229, 1138, 1167, 1200, 1494, 3542, 3543, 3547, 3562
<code>\l__enumext_vspace_a_star_v_bool</code>	1594
<code>\l__enumext_vspace_a_star_vii_bool</code>	1616
<code>\l__enumext_vspace_a_star_viii_bool</code>	1627
<code>\l__enumext_vspace_a_star_X_bool</code>	98
<code>__enumext_vspace_above:</code>	57, 95, 1562, 1562, 3484
<code>__enumext_vspace_above_v:</code>	57, 1590, 1590, 3637
<code>\l__enumext_vspace_above_v_skip</code>	1592, 1596, 1598
<code>__enumext_vspace_above_vii:</code>	58, 109, 1612, 1612, 4187
<code>\l__enumext_vspace_above_vii_skip</code>	1614, 1618, 1620
<code>__enumext_vspace_above_viii:</code>	58, 1612, 1623, 4435
<code>\l__enumext_vspace_above_viii_skip</code>	1625, 1629, 1631
<code>\l__enumext_vspace_b_star_v_bool</code>	1605
<code>\l__enumext_vspace_b_star_vii_bool</code>	1638
<code>\l__enumext_vspace_b_star_viii_bool</code>	1649
<code>\l__enumext_vspace_b_star_X_bool</code>	98
<code>__enumext_vspace_below:</code>	57, 97, 1576, 1576, 3570
<code>__enumext_vspace_below_v:</code>	58, 1601, 1601, 3707
<code>\l__enumext_vspace_below_v_skip</code>	1603, 1607, 1609
<code>__enumext_vspace_below_vii:</code>	58, 109, 1634, 1634, 4197
<code>\l__enumext_vspace_below_vii_skip</code>	1636, 1640, 1642
<code>__enumext_vspace_below_viii:</code>	58, 1634, 1645, 4443
<code>\l__enumext_vspace_below_viii_skip</code>	1647, 1651, 1653
<code>__enumext_widest_from:nNNn</code>	41, 751, 751, 766, 785
<code>\g__enumext_widest_label_tl</code>	26, 36, 67, 493, 497, 501
<code>\l__enumext_wrap_label_opt_v_bool</code>	3227
<code>\l__enumext_wrap_label_opt_vii_bool</code>	111, 4248
<code>\l__enumext_wrap_label_opt_viii_bool</code>	115, 4472
<code>\l__enumext_wrap_label_opt_X_bool</code>	98
<code>\l__enumext_wrap_label_v_bool</code>	3223, 3227, 3235, 3265
<code>\l__enumext_wrap_label_vii_bool</code>	111, 4247, 4252, 4260, 4329
<code>\l__enumext_wrap_label_viii_bool</code>	115, 4471, 4476, 4484, 4563
<code>\l__enumext_wrap_label_X_bool</code>	98
<code>__enumext_wrapper_label_v:n</code>	3267, 3748
<code>__enumext_wrapper_label_vii:n</code>	4332
<code>__enumext_wrapper_label_viii:n</code>	4566
<code>\l__enumext_write_aux_file_tl</code>	29, 74, 84, 154, 2411, 2417, 2926, 2932
<code>enumext*</code>	5, 4134
<code>enumXi</code>	465
<code>enumXii</code>	465
<code>enumXiii</code>	465
<code>enumXiv</code>	465
<code>enumXv</code>	465
<code>enumXvi</code>	465
<code>enumXvii</code>	465
<code>enumXviii</code>	465
Environments provide by <code>enumext</code> :	
<code>anskey*</code>	28, 64, 73–76, 78–80, 82, 95, 110, 119, 125, 127
<code>enumext*</code>	25, 26, 29–31, 34, 36, 39, 40, 42–45, 47, 54, 55, 58–61, 63–66, 68–78, 81, 83, 84, 88, 89, 93–95, 103–105, 107, 110, 111, 113, 115, 117, 119, 120, 122, 126, 129, 130
<code>enumext</code>	25, 26, 30, 31, 34, 36–40, 42–50, 53, 55–57, 59–61, 63–66, 68–74, 76–78, 81, 83, 84, 86–89, 91–93, 95, 97–99, 101, 104, 106, 108, 110, 119, 120, 122, 126, 127, 129
<code>keyans*</code>	25, 26, 28–32, 36, 39–42, 44, 45, 47, 54, 55, 58, 64, 65, 68, 69, 71, 79, 83, 89, 93, 103–105, 114, 126, 128, 130
<code>keyanspic</code>	25, 26, 28, 29, 32, 36, 37, 40, 64, 65, 68, 71, 79, 83–85, 100–102, 128
<code>keyans</code>	25, 26, 28, 29, 31, 32, 36, 37, 40, 42, 44, 45, 47, 49, 53, 55–58, 64, 65, 68, 69, 71, 79, 83–85, 89–92, 97–102, 106, 115, 126, 128
Environments:	
<code>list</code>	30, 33, 91, 93
<code>lrbox</code>	104, 112, 113, 117
<code>minipage</code>	30, 34, 47, 50, 51, 100–104, 112, 113, 117
<code>multicols</code>	48–51, 55, 96, 97
<code>scontents</code>	79, 81
exp commands:	
<code>\exp_after:wN</code>	4642
<code>\exp_args:Ne</code>	2816, 2824, 3440, 4630
<code>\exp_args:Nv</code>	2536, 2691, 3167, 3185, 3207, 4914
<code>\exp_not:N</code>	58, 496, 610, 655, 672, 725, 926, 940, 941, 952, 953, 964, 965, 2476, 2507, 2508, 2965, 3030, 3031, 3043, 3044, 4525, 4526, 4639
<code>\exp_not:n</code>	287, 302, 315, 323, 331, 549, 569, 610, 611, 655, 656, 672, 673, 725, 726, 927, 1683, 1692, 2143, 2240, 2252, 2414, 2442, 2452, 2462, 2476, 2477, 2783, 2796, 2806, 2929, 2967, 2969, 4743, 4753, 4946, 4951
F	
<code>\fbox</code>	2126
<code>\fboxrule</code>	2126
<code>\fboxsep</code>	2126
file commands:	
<code>\file_input_stop:</code>	5350
<code>first</code>	978
<code>font</code>	505
<code>\footnote</code>	103
<code>\footnote</code>	103, 3840
<code>\footnotemark</code>	3850
<code>\footnotesize</code>	2508, 3031, 3044, 4526
<code>\footnotetext</code>	3834
<code>\foreachkeyans</code>	16, 123, 4879
G	
<code>\getkeyans</code>	16, 118, 4628
group commands:	
<code>\group_begin:</code>	2506, 2551, 2726, 2813, 3029, 3042, 4308, 4327, 4524, 4551, 4561, 4685

\group_end: 2513, 2567, 2830, 3036, 3049, 4337, 4349, 4531, 4571, 4591, 4692

H

\hbadness 4356, 4598
hbox commands:
 \hbox_set:Nn 485
\hfill 535, 539, 544, 545, 1498, 1527, 2476, 2965, 4049, 4104
hook commands:
 \hook_gput_code:nnn 9, 195, 199, 203, 402
 \hook_gremove_code:nn 81, 2742
 \hook_gset_rule:nnnn 403
 \hook_if_empty:nTF 2740
\hspace 4367, 4610
\hyperlink 75, 85
\hyperlink 2476, 2965
\hypertarget 35
\hypertarget 432

I

\IfHyperBoolean 410
\IfPackageLoadedTF 11, 19, 406, 420
\ignorespaces 929
\inputlineno 287, 302, 315, 323, 331
int commands:
 \int_add:Nn 3960, 4009
 \int_case:nn ... 1102, 1220, 1991, 2017, 2056, 2080
 \int_case:nnTF 231
 \int_compare:nNnTF .. 388, 643, 660, 680, 687, 1190, 1209, 1372, 1462, 1478, 1490, 1522, 2104, 2110, 2575, 2579, 2583, 2591, 2637, 2641, 2645, 2842, 2863, 2902, 2907, 2912, 2937, 3025, 3422, 3433, 3455, 3468, 3506, 3521, 3536, 3556, 3622, 3626, 3654, 3679, 3692, 3715, 3719, 3775, 3930, 3940, 3956, 3979, 3989, 4005, 4162, 4166, 4205, 4215, 4362, 4371, 4409, 4421, 4604, 4614, 4797, 4929
 \int_compare_p:nNn ... 256, 266, 278, 279, 293, 294, 1468, 1469, 1997, 2023, 2359, 2369, 2381, 2382, 2397, 2438, 2614, 2615, 2626, 2627, 2779, 3465
 \int_decr:N 3959, 4008
 \int_eval:n .. 373, 780, 2258, 2409, 2508, 2920, 3031, 3044, 3320, 3364, 3948, 3997, 4526
 \int_from_alph:n 745, 759
 \int_from_roman:n 747, 761
 \int_gadd:Nn 3961, 4010
 \int_gdecr:N 2000, 2005, 2009, 2013, 2026
 \int_gincr:N 1833, 1838, 2421, 2975, 3064, 3098, 3241, 3497, 3646, 3737, 4225, 4303, 4449, 4515
 \int_gset:Nn 2049, 3848
 \int_gset_eq:NN 1732, 1739, 1745, 1751, 1759, 1766, 1772, 1778, 3845
 \int_gzero:N . 344, 345, 346, 1508, 1536, 2116, 2835, 3561, 3697, 4380, 4625
 \int_if_exist:NnTF 1707, 1743, 1749, 1770, 1776, 1954
 \int_incr:N 2590, 3421, 3617, 3774, 4161, 4224, 4408, 4448
 \int_mod:nn 4373, 4616
 \int_new:N . 28, 29, 30, 31, 32, 33, 61, 62, 86, 102, 121, 141, 142, 147, 148, 149, 151, 162, 168, 169, 170, 171, 172, 1709, 1957
 \int_set:Nn 741, 745, 747, 1846, 1853, 1865, 1874, 2727, 3819, 3820, 3875, 3906, 3929, 3935, 3951, 3978, 3984, 4000, 4356, 4598, 4793, 4931
 \int_set_eq:NN 1834, 1839, 3958, 4007
 \int_sign:n 2051
 \int_step_function:nnN 2375, 2388, 2402

\int_step_function:nnnN 4935
\int_step_inline:nn 4845
\int_step_inline:nnn 3821
\int_to_roman:n 207, 2355, 2392
\int_use:N 366, 371, 372, 1191, 1210, 1491, 1848, 1855, 1867, 1876, 3320, 3339, 3364, 3441, 3507, 3516, 3531, 3537, 3933, 3934, 3946, 3982, 3983, 3995, 5266, 5270, 5276, 5280
 \int_zero:N 4365, 4608
\item . 86, 90, 110, 111, 115, 117, 379, 2276, 2282, 2307, 2313, 2435, 2939, 2942, 3116, 3245, 3806, 4146, 4148, 4395, 4397, 4513
\item* 5, 14, 68, 3243
item-pos* 3141
item-sym* 3141
\itemindent 92
\itemindent 91
itemindent 873
\itemsep 101, 102
\itemsep 3790, 3796
\itemwidth . 455, 2126, 3402, 3411, 3596, 3605, 3969, 3973, 4018, 4022

K

keyans 14, 3575
keyans* 14, 4384
keyanspic 15, 3750
Keys for \anskey provide by enumext:
 break-col 74, 76, 79-81
 item-join 75, 76, 79-81
 item-pos* 75, 76, 79-81
 item-star 75, 76, 79-81
 item-sym* 75, 76, 79-81
Keys for anskey* provide by enumext:
 break-col 74, 76, 79-81
 item-join 75, 76, 79-81
 item-pos* 75, 76, 79-81
 item-star 75, 76, 79-81
 item-sym* 75, 76, 79-81
Keys for environments provide by enumext:
 above* 27, 57, 58, 95, 109
 above 27, 57, 58, 95, 109, 114
 after 45, 46, 97, 109, 114
 align 27, 37, 88, 91, 112, 125
 base-fix 43, 59, 71, 94, 109, 120
 before* 45, 46, 95, 109, 114
 before 45, 46
 below* 27, 57, 58, 97, 109
 below 27, 57, 58, 97, 109, 114
 check-ans . 29, 30, 32, 63-65, 67, 68, 71, 82, 85, 97, 109, 113, 126
 columns-sep 47, 96
 columns 27, 47, 56, 96
 first 45, 46, 112
 font 37, 88, 91, 112
 item-pos* 87, 88
 item-sym* 28, 87, 88
 itemindent 27, 44, 86, 87, 90, 112
 itemsep 42, 93
 labelsep 37, 92, 112
 labelwidth 36-41, 92
 label 26, 36, 38, 41, 104
 lisparindent 93
 list-indent 27, 44, 102
 list-offset 44, 94, 98

listparindent	44, 112
mark-ans	68, 71, 76
mark-pos	68, 69, 125
mark-ref	68, 71, 73, 75
mini-env	27, 34, 47, 55, 56, 71, 96, 106, 107, 109, 114
mini-right*	27, 30, 47, 71, 107, 109
mini-right	27, 30, 47, 55, 71, 107, 109
mini-sep	27, 47, 71, 96
no-store	29, 63–66, 71, 77, 86, 87
noitemsep	42
nosep	42
parindent	93
parsep	42, 93, 112
partopsep	42
ref	26, 30, 38–40, 126
resume*	26, 59, 60, 63, 64, 71, 97, 109, 121
resume	26, 33, 59–64, 71, 97, 109, 121
rightmargin	44, 104
save-ans	28, 33, 59–64, 66, 67, 69–71, 77–79, 82–84, 90, 98, 100, 115, 116, 118, 119, 121, 126
save-key	28, 59, 70, 94, 109
save-pos	71
save-ref	29, 35, 68, 71, 73–75, 83, 85, 90, 116
save-sep	68, 71, 116
series	26, 59–63, 71, 94, 97, 109, 121
show-ans	68, 69, 71, 73, 74, 76, 90, 116
show-length	31, 45, 126
show-pos	28, 68, 69, 73, 74, 76, 85, 90, 116
start*	27, 41, 42, 59
start	27, 30, 41, 42, 59
store-key	69
topsep	42
widest	26, 30, 41, 42
wrap-ans	35, 68, 71, 73, 75
wrap-label*	27, 37, 87, 88, 91, 111, 112, 115
wrap-label	27, 37, 86–88, 91, 111, 112, 115
wrap-opt	68, 71
keys commands:	
\keys_define:nn	507, 529, 561, 577, 624, 695, 769, 795, 839, 875, 898, 971, 980, 1059, 1076, 1547, 1658, 1901, 1962, 2121, 2163, 2199, 2204, 2518, 2669, 2705, 3143, 3159, 3179, 3199, 4656, 4755, 4871, 4879
\keys_if_exist_p:nn	4867, 4868
\l_keys_key_str	76, 80, 2536, 2691, 3167, 3185, 3207, 4914, 5022
\keys_precompile:nnN	119, 191, 191, 4658, 4662, 4666, 4670, 4674, 4678, 4897
\keys_set:nn	521, 855, 866, 1082, 1552, 1557, 1795, 1800, 1887, 1895, 2556, 3435, 3440, 3633, 4179, 4430, 4710, 4717, 4759, 4764, 4765, 4766, 4767, 4770, 4775, 4776, 4777, 4778, 4779, 4780, 4781, 4813, 4923
\keys_set_known:nn	2823
keyval commands:	
\keyval_parse:NNn	1672, 2229, 4731
L	
label	575, 622, 695
Labels provide by enumext:	
\Alph*	36
\Roman*	36
\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	102
\labelsep	3791, 3794
labelsep	505

\labelwidth	36, 102
\labelwidth	3791, 3792
labelwidth	505
\lastkern	242
\lastnodetype	231
\lastskip	236
\leftmargin	92
\leftmargin	91, 3791
legacy commands:	
\legacy_if:nTF	4291, 4294, 4539, 4542
\legacy_if_gset_false:n	393
\legacy_if_set_false:n	4293, 4541
\legacy_if_set_true:n	4253, 4278, 4285, 4298, 4477, 4505, 4546
\linewidth	96
\linewidth	3404, 3492, 3598, 3643, 3818, 3878, 3909, 4031, 4086
\list	377
list-indent	873
list-offset	873
\listparindent	3793
listparindent	873
\lrbox	4309, 4552
M	
\makebox	104
\makebox	2334, 2336, 3110, 4323, 4331, 4335, 4565, 4569
\makelabel	86, 88, 91, 103
\makelabel	86, 90, 3127, 3261
\makesavenoteenv	426
mark-ans	2119
mark-pos	2119, 2161
mark-ref	2119
mini-env	1057
mini-sep	1057
\minipage	383
\miniright	10, 55, 1460, 1512, 1540, 3559, 3695
mode commands:	
\mode_if_math:TF	2599, 2653
\mode_if_vertical:TF	1127, 1156, 1177, 1201, 1352, 1381
\mode_leave_vertical:	853, 864, 926, 940, 952, 964, 2332, 3108, 4321
msg commands:	
\msg_error:nn	1514, 1542, 2560, 2593, 2597, 2651, 2759, 3624, 3628, 3717, 3777, 3808, 4164, 4411, 4423, 4782, 4841
\msg_error:nnn	600, 647, 664, 717, 1464, 1471, 1476, 1510, 1538, 1807, 1811, 1926, 2542, 2601, 2619, 2631, 2639, 2643, 2647, 2655, 2697, 3173, 3191, 3213, 4168, 4416, 4644, 4653, 4724, 4829, 4860, 4869, 4906, 4927
\msg_error:nnnn	2545, 2573, 2577, 2581, 2585, 2700, 3176, 3194, 3216, 3615, 3713, 3721, 4705, 4909
\msg_error:nnnnn	548, 568, 2142
\msg_fatal:nn	3423
\msg_fatal:nnn	459
\msg_info:nnn	13, 16, 21, 24, 408, 422
\msg_line_context:	4987, 4992, 4997, 5026, 5031, 5036, 5051, 5066, 5070, 5074, 5078, 5082, 5086, 5093, 5100, 5106, 5120, 5124, 5129, 5133, 5137, 5141, 5146, 5150, 5154, 5158, 5163, 5198, 5202, 5207, 5212, 5216, 5221, 5297, 5301, 5306, 5311, 5316, 5320, 5324, 5328, 5332, 5336, 5340, 5344, 5348
\msg_log:nnn	1946, 1951, 1956
\msg_log:nnnnn	370, 2089, 2094, 2099

<code>\msg_log:nnnnnn</code>	362	<code>\phantomsection</code>	35
<code>\msg_new:nnn</code> 4954, 4958, 4962, 4966, 4971, 4984, 4989, 4994, 4999, 5008, 5016, 5020, 5024, 5029, 5034, 5049, 5064, 5068, 5072, 5076, 5080, 5084, 5088, 5097, 5103, 5109, 5113, 5117, 5122, 5127, 5131, 5135, 5139, 5144, 5148, 5152, 5156, 5161, 5196, 5200, 5205, 5210, 5214, 5219, 5295, 5299, 5304, 5309, 5314, 5318, 5322, 5326, 5330, 5334, 5338, 5342, 5346		<code>\phantomsection</code>	433
<code>\msg_new:nnnn</code> .. 4975, 5166, 5175, 5184, 5190, 5223, 5233, 5243, 5253, 5263, 5273, 5283, 5289		prg commands:	
<code>\msg_term:nnnn</code> .. 1910, 1915, 3329, 3339, 3370, 3375		<code>\prg_do_nothing:</code>	437
<code>\msg_term:nnnnn</code>	2070	<code>\prg_new_protected_conditional:Npnn</code> ...	209
<code>\msg_warning:nn</code>	3558, 3694	<code>\prg_replicate:nn</code>	226
<code>\msg_warning:nnnn</code> 2107, 2113, 3277, 3282, 3932, 3945, 3981, 3994		<code>\prg_return_false:</code>	213
<code>\msg_warning:nnnnn</code>	2065, 2075	<code>\prg_return_true:</code>	212
<code>\multicolsep</code>	96	<code>\printkeyans</code>	16, 119, 4683
<code>\multicolsep</code>	1194, 1375, 3527, 3670	prop commands:	
N		<code>\prop_const_from_keyval:Nn</code>	4830
<code>\NeedsTeXFormat</code>	3	<code>\prop_count:N</code> 364, 2258, 2409, 2510, 2920, 3033, 3046, 4528, 4932	
<code>\newcounter</code>	462	<code>\prop_get:NnNTF</code>	4856
<code>\NewDocumentCommand</code> 1460, 2548, 3709, 4628, 4683, 4789, 4838, 4916		<code>\prop_gput_if_not_in:Nnn</code>	2256
<code>\NewDocumentEnvironment</code> .. 3380, 3575, 3750, 4134, 4384		<code>\prop_if_exist:NTF</code>	1944, 4648, 4925
<code>\newenvsc</code>	2662	<code>\prop_item:Nn</code>	4650, 4949
<code>\newlabel</code>	35	<code>\prop_new:N</code>	1947
<code>\newlabel</code>	444	<code>\ProvidesExplPackage</code>	4
no-store	1960	R	
<code>\noindent</code>	4040, 4095, 4147, 4364, 4396, 4607	<code>\raggedcolumns</code>	3530, 3673
<code>\nointerlineskip</code> 1203, 1206, 1383, 1386, 1502, 1530, 4040, 4095		<code>\ref</code>	73, 83
noitemsep	793	ref	575, 622, 695
<code>\nopagebreak</code> 1139, 1168, 1203, 1206, 1383, 1386, 1451, 1457		<code>\refstepcounter</code>	4300, 4548
<code>\normalfont</code>	2507, 3030, 3043, 4525	regex commands:	
nosep	793	<code>\regex_match:nnTF</code> .. 211, 744, 746, 758, 760, 2755	
P		<code>\regex_replace_once:nnN</code>	219
Packages:		<code>\renewcommand</code>	610, 655, 672, 725
<code>caption</code>	107	<code>\RenewDocumentCommand</code> 1512, 1540, 3116, 3127, 3245, 3261, 3806, 3840	
<code>enumext</code>	25, 35, 38, 63, 91, 100, 124, 125	<code>\RequirePackage</code>	17, 25
<code>enumitem</code>	36	resume	1656
<code>expl3</code>	103	resume*	1656
<code>footnotehyper</code>	35	rightmargin	873
<code>hyperref</code>	29, 30, 34, 35, 75, 85, 111, 124	<code>\Roman</code>	36, 41
<code>lua-visual-debug</code>	50	<code>\Roman</code>	481
<code>multicol</code>	25, 124	<code>\roman</code>	36, 41
<code>scontents</code>	25, 78, 79	<code>\roman</code>	482, 593, 4673
<code>shortlst</code>	103	S	
<code>\par</code> .. 1139, 1168, 1206, 1386, 1451, 1457, 1495, 1502, 1530, 2484, 3544, 3548, 3684, 3700, 3830, 4058, 4072, 4113, 4127, 4364, 4378, 4607, 4623		<code>\s</code>	2756
<code>\parbox</code>	2126	save-ans	1899
<code>\parindent</code>	4341, 4575	save-key	2197
<code>\parsep</code>	48, 101, 102	save-ref	2119
<code>\parsep</code>	3361, 3790, 3797, 3802	save-sep	2119
<code>parsep</code>	793	scan commands:	
<code>\parskip</code>	4342, 4576	<code>\scan_stop:</code>	102, 3804, 4146, 4395, 4639, 4642
<code>\partopsep</code>	102	scontents internal commands:	
<code>\partopsep</code>	3362, 3795	<code>\l_scontents_fname_out_tl</code>	2715
<code>partopsep</code>	793	<code>__scontents_parse_environment_keys:n</code> ..	2721
peek commands:		<code>__scontents_rescan_tokens:n</code>	2728
<code>\peek_meaning:NTF</code> 4230, 4244, 4261, 4272, 4454, 4468, 4485		<code>\l_scontents_storing_bool</code>	2713
<code>\peek_meaning_remove:NTF</code>	4237, 4461	<code>\l_scontents_writing_bool</code>	2714
<code>\peek_remove_spaces:n</code>	3249	seq commands:	
		<code>\seq_clear:N</code>	4791, 4934
		<code>\seq_const_from_clist:Nn</code>	4784
		<code>\seq_count:N</code>	365, 3763, 4795
		<code>\seq_gclear:N</code>	3838, 3839
		<code>\seq_gput_right:Nn</code>	2265, 3851, 3852
		<code>\seq_if_empty:NTF</code>	3857, 4698, 4809
		<code>\seq_if_exist:NTF</code>	1949, 4696
		<code>\seq_if_in:NnTF</code>	4703
		<code>\seq_item:Nn</code>	2753, 3827
		<code>\seq_map_function:NN</code>	4800

<code>\seq_map_inline:Nn</code>	4711, 4718, 4810, 4811
<code>\seq_map_pairwise_function:NNN</code>	3859
<code>\seq_new:N</code>	122, 123, 125, 139, 163, 164, 1952
<code>\seq_pop_left:NN</code>	4799
<code>\seq_put_right:Nn</code>	3723, 4807, 4823, 4944
<code>\seq_set_from_clist:Nn</code>	4792
<code>\seq_set_map_e:NNn</code>	4801
<code>\seq_show:N</code>	4700
<code>\seq_use:Nn</code>	191, 192, 4940
<code>series</code>	1656
<code>\setcounter</code>	755, 759, 761, 3320, 3364, 3768
<code>\setenumext</code>	6, 120, 4789
<code>\setenumextmeta</code>	6, 122, 4830
<code>show-ans</code>	2119, 2161
<code>show-length</code>	969
<code>show-pos</code>	2161
skip commands:	
<code>\skip_add:Nn</code> 1107, 1113, 1119, 1129, 1133, 1158, 1162, 1179, 1237, 1239, 1253, 1256, 1277, 1279, 1293, 1296, 1316, 1318, 1332, 1335, 1354, 1365, 3790	
<code>\skip_gset:Nn</code>	1399, 1403, 1407
<code>\skip_gzero_new:N</code>	1394, 1395
<code>\skip_horizontal:N</code> 941, 953, 965, 4324, 4338, 4572	
<code>\skip_horizontal:n</code> . . . 927, 2333, 2341, 3109, 3111, 4322, 4581	
<code>\skip_if_eq:nnTF</code> 1105, 1111, 1117, 1223, 1263, 1303, 1361, 1363, 1396, 1418, 1564, 1578, 1592, 1603, 1614, 1625, 1636, 1647	
<code>\skip_new:N</code> . . . 81, 82, 83, 87, 88, 89, 90, 91, 143, 183	
<code>\skip_set:Nn</code> 1090, 1094, 1144, 1148, 1173, 1226, 1227, 1245, 1266, 1267, 1285, 1305, 1306, 1324, 1348, 1398, 1402, 1420, 1424, 1428, 1434, 1438, 1442, 3784, 3798	
<code>\skip_set_eq:NN</code> 1184, 1185, 1187, 1194, 1359, 1375, 3318, 3360, 3361, 4341, 4342, 4575, 4576	
<code>\skip_sub:Nn</code> 1233, 1235, 1249, 1251, 1273, 1275, 1289, 1291, 1312, 1314, 1328, 1330, 1368	
<code>\skip_use:N</code> 1092, 1096, 1131, 1135, 1140, 1160, 1164, 1175, 1181, 1565, 1569, 1572, 1579, 1583, 1586, 3544, 3548	
<code>\skip_vertical:N</code>	394, 397, 1493, 1524
<code>\skip_zero:N</code> 1193, 1207, 1345, 1346, 1347, 1374, 1387, 3362, 3527, 3670, 3795, 3796	
<code>\skip_zero_new:N</code>	1393, 1415, 1416, 1417
<code>\l_tmpa_skip</code> 1245, 1255, 1258, 1285, 1295, 1298, 1324, 1334, 1337	
<code>\c_zero_skip</code> . 394, 397, 1105, 1111, 1117, 1264, 1303, 1361, 1363, 1396, 1418, 1565, 1579, 1592, 1603, 1614, 1625, 1636, 1647	
<code>\small</code>	4661, 4665, 4669, 4673, 4677, 4681
<code>\star</code>	3147
<code>start</code>	767
<code>start*</code>	767
<code>\stepcounter</code>	3730, 3844
str commands:	
<code>\c_backslash_str</code> 2601, 4987, 4992, 4997, 5002, 5004, 5006, 5011, 5013, 5111, 5115, 5119, 5129, 5133, 5141, 5142, 5146, 5158, 5159, 5163, 5164, 5185, 5187, 5191, 5193, 5221, 5284, 5286, 5290, 5292, 5301, 5302, 5306, 5311, 5312, 5316, 5320, 5324	
<code>\c_colon_str</code>	2408, 2919, 4639
<code>\c_left_brace_str</code>	5092, 5099, 5105
<code>\c_right_brace_str</code>	5092, 5099, 5105
<code>\str_case:nn</code>	249, 308
<code>\str_case:nnTF</code> . 1679, 1687, 2236, 2244, 4738, 4747	
<code>\str_clear:N</code>	3432, 4178
<code>\str_count:n</code>	226
<code>\str_if_empty:NTF</code>	1696, 1737, 1764
<code>\str_if_eq:nnTF</code>	3321, 3366, 4840
<code>\str_if_in:nnTF</code>	4635
<code>\str_new:N</code>	129, 178
<code>\str_set:Nn</code> . . . 564, 565, 566, 2139, 2140, 2166, 2167	
<code>\string</code>	426
<code>\strutbox</code> . 1212, 1215, 1226, 1227, 1238, 1240, 1255, 1258, 1266, 1267, 1278, 1280, 1295, 1298, 1305, 1306, 1317, 1319, 1334, 1337, 1365, 1389, 1398, 1399, 1402, 1409, 1422, 1430, 1436, 1444, 3800	
T	
TeX and \TeX 2 _ε commands:	
<code>\@auxout</code>	442
<code>\@currentvir</code>	249, 308
<code>\protected@write</code>	442
tex commands:	
<code>\tex_newlinechar:D</code>	2727
text commands:	
<code>\text_expand:n</code>	4631
<code>\textasteriskcentered</code>	2136, 2153
<code>\the</code>	236, 242
<code>\thepage</code>	448
tl commands:	
<code>\c_space_tl</code> 3001, 5036, 5051, 5074, 5078, 5265, 5266, 5275, 5276, 5336, 5340	
<code>\tl_clear:N</code> . . 534, 540, 2117, 2183, 2193, 2214, 2222, 2428, 2747, 2748, 2862, 2936, 4491	
<code>\tl_clear_new:N</code>	491
<code>\tl_const:Nn</code>	50, 475
<code>\tl_gclear:N</code> . 356, 357, 358, 1717, 1722, 2837, 3138, 4076, 4131, 4325	
<code>\tl_gclear_new:N</code>	1704
<code>\tl_gput_right:Nn</code>	476
<code>\tl_greplace_all:Nnn</code>	497
<code>\tl_gset:Nn</code> 284, 285, 299, 300, 1705, 1718, 1723, 1942, 2751, 3085, 4267	
<code>\tl_gset_eq:NN</code>	493, 3081, 4318
<code>\tl_if_blank:nTF</code> 2540, 2558, 2695, 3171, 3189, 3211, 4316, 4904	
<code>\tl_if_empty:NTF</code> . 598, 617, 645, 662, 682, 689, 715, 732, 1730, 1735, 1757, 1762, 1820, 1884, 1892, 1921, 1981, 2272, 2303, 2448, 2792, 2814, 2844, 2873, 2946, 2995, 3106, 4494, 4821	
<code>\tl_if_empty:nTF</code>	1785
<code>\tl_if_exist:NTF</code>	1790
<code>\tl_if_novalue:nTF</code> . . 2554, 2870, 2944, 2980, 3060, 3079, 3087, 3221, 3430, 3761, 3842, 4176, 4428, 4492	
<code>\tl_map_inline:Nn</code>	217, 494
<code>\tl_new:N</code> 42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 103, 104, 105, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 124, 126, 127, 128, 130, 133, 134, 146, 154, 155, 156, 159, 177	
<code>\tl_put_left::Ne</code>	2781
<code>\tl_put_left:Nn</code> 2280, 2311, 2433, 2775, 2788, 2794, 2804, 3012, 3052, 4061, 4116, 4510, 4513	
<code>\tl_put_right:Nn</code> 492, 608, 653, 670, 723, 2284, 2315, 2362, 2372, 2385, 2400, 2406, 2411, 2435, 2440, 2447, 2450, 2460, 2465, 2468, 2474, 2865, 2868, 2875, 2877, 2904, 2909, 2914, 2917, 2926, 2939, 2942, 2948, 2953, 2963, 4496, 4497	
<code>\tl_remove_all:Nn</code>	4820
<code>\tl_remove_once:Nn</code>	2350, 2889
<code>\tl_replace_all:Nnn</code>	496, 4855
<code>\tl_reverse:N</code>	2349, 2351, 2888, 2890

<code>\tl_set:Nn</code> .	58, 253, 263, 312, 313, 320, 321, 328, 329, 461, 535, 539, 544, 545, 597, 642, 714, 924, 938, 950, 962, 1819, 1920, 2184, 2194, 2215, 2223, 2504, 2715, 2982, 3027, 3040, 4499, 4522, 4818, 4854, 4924
<code>\tl_set_eq:NN</code>	502, 603, 606, 650, 652, 667, 669, 720, 722, 2348, 2887, 2900, 3233, 3237, 3742, 3744
<code>\tl_to_str:n</code>	1790, 1796, 1801, 4631
<code>\tl_trim_spaces:n</code> . . .	492, 4807, 4818, 4824, 4840
<code>\tl_use:N</code> .	498, 501, 619, 684, 691, 734, 995, 999, 1003, 1007, 1011, 1015, 1019, 1023, 1027, 1031, 1035, 1039, 1043, 1047, 1051, 1055, 2338, 2355, 2363, 2374, 2387, 2392, 2403, 3068, 3074, 3102, 3129, 3130, 3137, 3224, 3228, 3236, 3263, 3264, 3270, 3387, 3581, 3747, 4068, 4123, 4328, 4339, 4343, 4562, 4573, 4579, 4584, 4686, 4687, 4688, 4689, 4690, 4708, 4803, 4922
token commands:	
<code>\token_to_str:N</code>	444
<code>topsep</code>	<u>793</u>
<code>\topskip</code>	1193, 1374
<code>\typeout</code>	235, 236, 241, 242, 412, 415, 425, 426
U	
<code>\u</code>	220, 2756
<code>\unkern</code>	243
<code>unknown</code>	<u>3157</u> , <u>3179</u> , <u>3197</u>
<code>\unskip</code>	237, 3698
use commands:	
<code>\use:N</code>	227, 3134, 3389
<code>\use:n</code>	1670, 2227, 4637, 4729
<code>\use_none:nn</code>	436, 4861
<code>\usecounter</code>	3319, 3363
V	
<code>\value</code>	1733, 1739, 1746, 1752, 1760, 1766, 1773, 1779
vbox commands:	
<code>\vbox_set_top:Nn</code>	4066, 4121
<code>\vspace</code> .	854, 865, 1569, 1572, 1583, 1586, 1596, 1598, 1607, 1609, 1618, 1620, 1629, 1631, 1640, 1642, 1651, 1653, 3758, 3769, 4379, 4624
W	
<code>widest</code>	<u>767</u>
<code>wrap-ans</code>	<u>2119</u>
<code>wrap-label</code>	<u>505</u>
<code>wrap-label*</code>	<u>505</u>
<code>wrap-opt</code>	<u>2119</u>
Z	
<code>\z</code>	2756