

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-10-06*

©2024 by Pablo González†

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

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

Abstract

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

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	13
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	16
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	16
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	17
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	17
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	17
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	18
5	The keyval system	7	7	Full examples	19
5.1	Keys for label and ref	7	8	Tagged PDF examples	21
5.2	Keys for spaces	8	9	The way of non-enumerated lists	22
5.2.1	Vertical spaces	8	10	References	24
5.2.2	Horizontal spaces	9	11	Change history	24
5.3	Keys for add code	9	12	Index of Documentation	25
5.4	Keys for start, series and resume	10	13	Implementation	27
5.5	Keys for <code>multicols</code>	10	14	Index of Implementation	141
5.6	Keys for <code>minipage</code>	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

Motivation and acknowledgments

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

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

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all \LaTeX team for their great work and to the different members of the `TeX-SX` community who have provided great answers and ideas. Here a note of the main ones:

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

*This file describes a documentation for v1.0, last revised 2024-10-06.

†E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), version 1.3 or later (<https://www.latex-project.org/lppl.txt>). The software has the status “maintained”.
The enumext package loads and requires multicol[3] and scontents[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. 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_EX 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_EX is cool?

*

Very True!

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

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn't exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete

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

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

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- ⌘ (b) Yes, dnf

⌘ (c) i. doesn't exist for now :(

⌘ ii. very good

⌘ iii. obsolete

⌘

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

* 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

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

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- Where what we are interested in the $\langle label \rangle$ and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L
- 2 / 156

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.

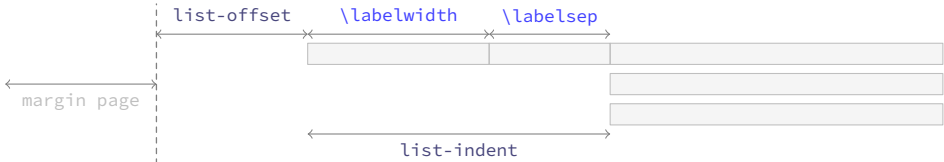


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

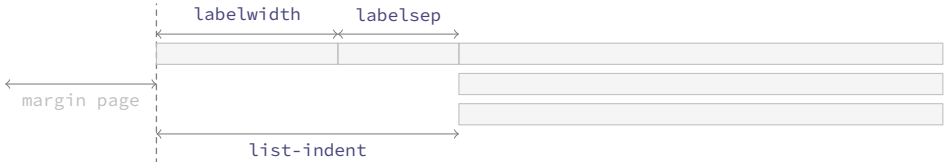


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

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 \TeX `\label` and `\ref` commands work as usual. It also provides an “internal reference” system for the “stored content” by means of the key `save-ref` (§6.1.1) when the key `save-ans` (§6.1) is active.

1.3.6 Support for \footnote

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

The best way to solve this is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

```
\usepackage{footnotehyper}
\makesavenoteenv{enumext}
\makesavenoteenv{enumext*}
```

At the moment the `footnotehyper` package is not compatible with *tagged* PDF.

2 The environments provided

The package `enumext` provides two main list environments, the *vertical* environment `enumext` and the *horizontal* environment `enumext*`.

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

2.1 The environment enumext

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

Example with `columns=2`

1. This text is in the first level.
- A. This text is in the fourth level.
- (a) This text is in the second level.
- X This text is in the first level.
- i. This text is in the third level.
- * 2. This text is in the first level.

2.2 The environment enumext*

The `enumext*` is a *horizontal list environment* similar to the `enumerate*` environment provided by the `enumitem` package or `task` environment provided by the `task` package, `\item` and `\item[\langle custom \rangle]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[\langle key = val \rangle]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “*item content*” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwidth`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

Example with `columns=2`

1. This text is in the first level.
2. This text is in the first level.
- X This text is in the first level.
- * 4. This text is in the first level.

2.3 The command \item*

```
\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 *second optional argument*. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\textasteriskcentered *` and the value set by `labelsep` key.

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

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

2.3.1 Keys for \item*

`item-sym*` = {<symbol>} default: \textasteriskcentered
Sets the *symbol* to be displayed in the “left” of the box containing the current <label> set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in *text* or *math* mode, for example `item-sym*={\star}`.
`item-pos*` = {<rigid length>} default: by levels
Sets the *offset* between the box containing the current <label> defined by `labelwidth` key and the <symbol> set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

2.4 The command \item in enumext*

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

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

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

3 The command \setenumext

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

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

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

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

4 The command \setenumextmeta

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

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

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

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

`mode-box` $\langle value forbidden \rangle$ default: *not used*

This is a “*switch-key*” that does not receive an argument and is “*only*” available for the “*first level*” of the `enumext` environment and the `enumext*` environment. When this is set the `label`, `font`, `wrap-label` and `wrap-label*` keys are executed within `\makebox` for the `enumext` and `keyans` environments.

- This key is intended for compatibility with *tagged* PDF and is forcibly “*enabled*” when `\DocumentMetadata` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.
- In the `enumext*` and `keyans*` environments `\makeLabel` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Keys for spaces

`show-length = {\true|false}` default: *false*

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

5.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

`noitemsep` *<value forbidden>* default: *not used*

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

`nosep` *<value forbidden>* default: *not used*

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

`base-fix` *<value forbidden>* default: *not used*

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

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

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

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

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

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

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

Set the *extra vertical space* 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* 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

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

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

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

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=0pt` is set in the environments `enumext` and `keyans` the $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”.

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

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

Sets the extra *horizontal indentation*, beyond `labelsep`, of the “*first line*” off each `\item` that is not followed by a “*blank line*” or the `\par` command. This value must be greater than or equal to `0pt` and is applied internally using `\hspace` without modifying the value of `\itemindent`.

- This key is intended for the `enumext*` and `keyans*` environments where, by their implementation, it is not possible to adjust `labelwidth` and `list-indent` without modifying the output. If you use `enumext` or `keyans` and want to get around the *blank line* limitation or the `\par` command followed by `\item` you can modify `labelwidth` and `label-indent` and get the same effect.

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

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

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

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

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

5.3 Keys for add code

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

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

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

`before* = {⟨code⟩}` default: *not used*
 Execute {⟨code⟩} “before” the environment starts. The {⟨code⟩} must be passed between braces, is executed “before” performing all calculations related to the *list parameters* and [`key = val`] sets in the environment that is, before the arguments defining the environment are executed: {⟨code⟩}\begin{list}{⟨arg one⟩}{⟨arg two⟩}.

`first = {⟨code⟩}` default: *not used*
 Executes {⟨code⟩} when “starting” the environment. The {⟨code⟩} must be passed between braces, is executed right “after” all *list parameters* are done, after the second argument of list, just before the first occurrence of \item: \begin{list}{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}\item.

- Keep in mind that the code set in this key will affect the entire “body” of the environment and therefore the inner levels of the list and the *keyans* environment. It is recommended to set this key per level.
- In the *enumext** and *keyans** environments this key is executed after the *listparindent*, *parsep* and *itemindent* keys within the *minipage* environment in which the “item content” is placed.

`after = {⟨code⟩}` default: *not used*
 Execute {⟨code⟩} “after” finishing the environment. The {⟨code⟩} must be passed between braces.

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

5.5 Keys for multicol

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

`columns-sep = {⟨rigid length⟩}` default: *by level*
 Set the *space between columns* used by the *multicol* environment within the environment. Internally sets the value of \columnsep, by default its value is equal to the sum of the values set in the keys *labelwidth* and *labelsep* of the current level.

- The \footnote{⟨text⟩} command in the nested levels of *multicol* will not work as expected, prefer the use of \footnotemark[⟨number⟩] inside the environment and \footnotetext[⟨number⟩]{⟨text⟩} outside the environment or via the *after* key.
- By default the *hyperref* package does not provide support for \footnotemark[⟨number⟩] and \footnotetext[⟨number⟩]{⟨text⟩}, but when *tagged* PDF is enabled and the package is loaded the \footnotemark[⟨number⟩] and \footnotetext[⟨number⟩]{⟨text⟩} commands create links correctly.

5.6 Keys for minipage

`mini-env = {⟨rigid length⟩}`

default: *not used*

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

`mini-sep = {⟨rigid length⟩}`

default: `0.3333em`

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

5.6.1 The command `\miniright`

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

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

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

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

• By default the `hyperref` package does not provide support for `\footnotemark[⟨number⟩]` and `\footnotetext[⟨number⟩]{⟨text⟩}`, but when *tagged PDF* is enabled and the package is loaded the `\footnotemark[⟨number⟩]` and `\footnotetext[⟨number⟩]{⟨text⟩}` commands create links correctly.

5.6.2 The key `mini-right`

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

`mini-right = {⟨content⟩}`

default: *not used*

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

`mini-right* = {⟨content⟩}`

default: *not used*

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

6 The storage system

The entire mechanism for “*storing content*” it is activated according to `save-ans` key on the “*first level*” of `enumext` or `enumext*` environments and it is ignored if they are established when they are nested inside each other. Only when this `⟨key⟩` is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
---	---

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

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

6.1 Keys for storage system

The only `⟨keys⟩` available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the `⟨keys⟩` described in this section must be passed directly in the *optional argument* of the “*first level*” of the environment in which the key `save-ans` is executed. The key `save-ans` should NOT be passed with the command `\setenumext`.

`save-ans = {⟨store name⟩}` default: *not set*
 Sets the *name* of the *sequence* and *prop list* in which the {⟨contents⟩} will be “stored” by `\anskey` and `\anskey*` in `enumext` and `enumext*` environments and the current {⟨labels⟩} for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic`. If the *sequence* or *prop list* {⟨store name⟩} does not exist, it will be created globally and will not be *overwritten* if the key is used again.

`save-key = {⟨key list⟩}` default: *not set*
 This key *overrides* the default “stored keys” of the *optional argument* of the inner levels or nested environment that will be passed to the *sequence*. The {⟨key list⟩} passed to this key ignores any {⟨keys⟩} in the “stored structure” and must be passed between braces. For example, if we execute at a second level:

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

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

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

6.1.1 Keys for `label` and `ref`

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

`mark-ref = {⟨symbol⟩}` default: `\%`
 Sets the *symbol* that will be displayed by the `\printkeyans` command only if the `hyperref` package is detected and the `save-ref` key are active. This “symbol” is used as a “link” between the environment in which the `save-ans` key was used and the place where the command is executed.

6.1.2 Keys for `wrap` and `display`

`wrap-ans = {⟨code {#1} more code⟩}` default: `\fbox+\parbox{#1}`
 Wraps the *argument* passed to the `\anskey` and the *body* in `\anskey*` environment referenced by {#1} when using the `show-ans` or `show-pos` keys. The {⟨code⟩} must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.

`wrap-opt = {⟨code {#1} more code⟩}` default: `[{#1}]`
 Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by {#1} in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The {⟨code⟩} must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* {⟨store name⟩} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.

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

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

`mark-pos = {⟨left | right⟩}` default: *left*
 Sets the *aligned* of the symbol defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

6.1.3 Keys for debug and checking

`show-pos` = { $\langle true \mid false \rangle$ } 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` = { $\langle true \mid false \rangle$ } 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` $\langle \text{anskey}[\langle keys \rangle] \langle content \rangle \rangle$

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 \LaTeX 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: *not set*
Sets the *symbol* for `\item*` when using the key `item-star` and stores { $\langle content \rangle$ } in the *sequence* { $\langle store\ name \rangle$ } of the form `\item*`[$\langle symbol \rangle$] $\langle content \rangle$. The *symbol* can be in text or math mode, for example `item-sym*={\ast}` stores `\item*`[\ast] $\langle content \rangle$.

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

Example

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

1. Text containing our instructions or questions.

*

first answer
2. Text containing our instructions or questions.

(a)

Question.

*

second answer
3. Text containing our instructions or questions.

*

third answer
4. Text containing our instructions or questions.

*

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* `\{store name\}` set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperLink` and `\hypertarget` will be used, otherwise the usual “*label and ref*” system provided by \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]
      \langle first answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \langle second answer \rangle
        \end{anskey*}
      \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle third answer \rangle
    \end{anskey*}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \langle fourth answer \rangle
    \end{anskey*}
\end{enumext}
```

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

6.4 The environments keyans and keyans*

keyans

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

keyans*

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

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

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

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

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

6.4.1 The \item* in keyans and keyans*

\item*

```
\item*
\item*[\langle content \rangle]
```

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

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

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

Example

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

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

  \item Text containing a question and image.

  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item*[\langle note \rangle] 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 key = val \rangle] \anspic*[\langle content \rangle]{\langle drawing or tabular \rangle} \end{keyanspic}
```

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

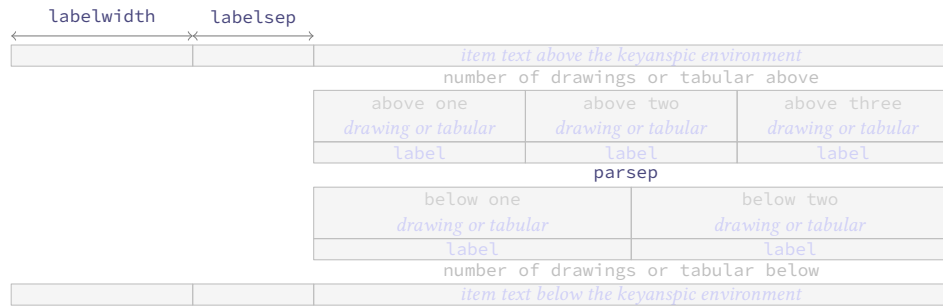


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

When the `keyanspic` environment is used *without keys* the `\label`s are centered *below* the *drawings or tabular* in a *single line* layout style.

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

6.5.1 Keys for keyanspic

`label-pos = {\langle above | below \rangle}` default: *below*

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

`label-sep = {\langle rubber length | rigid length \rangle}` default: *internal adjustment*

Set the *vertical spacing* between the `\label` centered “*above*” or “*below*” and *drawings or tabular* when running the `\anspic` command.

`layout-sty = {\langle n° upper , n° lower \rangle}` default: *not set*

Set the *number of drawings or tabular* that will be distributed “*upper*” and “*lower*” within the environment when executing the `\anspic` command. The value must be passed in braces and if not set or the `\langle n° lower \rangle` is omitted the *drawings or tabular* will be put on a *single line*.

`layout-sep = {\langle rubber length | rigid length \rangle}` default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings or tabular* placed at the “*upper*” and “*lower*” within the environment when executing the `\anspic` command. Internally adjusts the `parsep` value taken from the `keyans` environment.

`layout-top = {\langle rubber length | rigid length \rangle}` default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of `topsep` taken from `keyans` environment.

6.5.2 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 *optional argument* `\langle content \rangle` 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 and labels below.

  \begin{keyanspic}[layout-sty={3,2}]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
  \end{keyanspic}
```

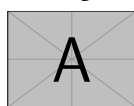
`\item` Question with images and labels above.

```
\begin{keyanspic}[label-pos=above, layout-sty={3,2}]
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
```

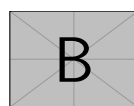
`\item` Question with images and labels *below* on a single line.

```
\begin{keyanspic}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}
\end{enumext}
```

1. Question with images and labels below.



A)



B)



C)

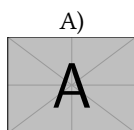


D)

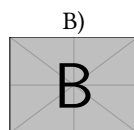


* E)[note]

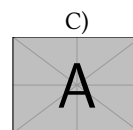
2. Question with images and labels above.



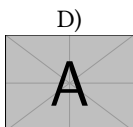
A)



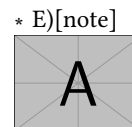
B)



C)



D)



* E)[note]

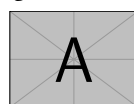
3. Question with images and labels below on a single line.



A)



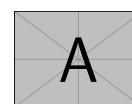
B)



C)



D)



* E)[note]

💡 Preferably use `label-pos=above` when creating a *tagged* PDF, this will preserve the reading order and navigation of the document.

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

6.6.3 The command `\printkeys`

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

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

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

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

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

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

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

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

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

Example

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

  \begin{enumext}[nosep]
    \item \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)

[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: 10^{-15} m). Qual è la relazione tra queste due unità di misura?
- A

36 km/h.

B

360 km/h.
- C
- 27,8 km/h.

D

A

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

B

C

D

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

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

©2024 by Pablo González L

19 / 156

- 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: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15} \text{ m}$). Qual è la relazione tra queste due unità di misura?
- A

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

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

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

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

1. B

2. A

3. B

4. A

Example 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 $1,00 \times 10^2 \text{ m/s}$ 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: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15} \text{ m}$). Qual è la relazione tra queste due unità di misura?
- A

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

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

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

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

1. B

2. A

3. La velocità di $1,00 \times 10^2 \text{ m/s}$ 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: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15} \text{ m}$). Qual è la relazione tra queste due unità di misura?

A

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

B

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

C

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

D

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

3. A

4. A

Example 3

A “simple multiple choice” test

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A

I only

B

II only

C

I and II only

D

I and III only

E

I, II, and III

3. Third type of questions

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

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

A

value

B

value

C

value

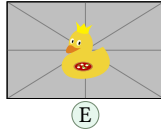
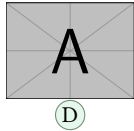
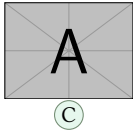
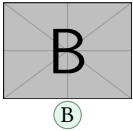
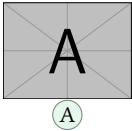
D

value

E

value

4. Question with image and label below:



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

21 / 156

- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged PDF* .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged PDF* .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file  and *tagged PDF* .

9 The way of non-enumerated lists

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

Here I leave as an example other uses of the `enumext` environment that can be helpful for specific purposes. The “trick” to generate these *fake environments* is set `label={}` or `label={\some}` and play with the `list-indent`, `list-offset`, `font` and `wrap-label` keys.

Fake itemize environment

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

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

Fake description environment

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

Description indented by label

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

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

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

Something A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

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

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

Description with multi-line labels

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

```
\NewDocumentCommand \labelbx { s +m }
{
  \%
  \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 <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
SoMeThInG	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

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

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

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

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.
Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

Why not random questions and other utilities

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

Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it's something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that `enumext` serves this purpose and inspires more users and authors to follow this path.

10 References

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

11 Change history

v1.0 2024-10-06 – First public release.

12 Index of Documentation

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

C		F	
Document class:		\footnote	5
article	2	I	
book	2	\itemsep	8
exam	2	K	
letter	2	Keys for \anskey provide by enumext:	
report	2	break-col	13
\columnbreak	4, 13	item-join	13
\columnsep	10	item-pos*	13
Commands provide by enumext:		item-star	13
\anskey	11-14	item-sym*	13
\anspic	11-13, 16	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	17	after	18
\getkeyans	13, 17	before	18
\item*	5-7, 11-13, 15	sep	18
\item	5-7, 10, 11, 13, 15, 16	start	18
\miniright	11	step	18
\printkeyans	6, 12, 18	stop	18
\setenumextmeta	6	wrapper	18
\setenumext	5-7, 11, 12, 15, 18	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	13
enumXiii	4	force-eol	14
enumXii	4	item-join	13
enumXiv	4	item-pos*	13
enumXi	4	item-star	13
enumXviii	4	item-sym*	13
enumXvii	4	overwrite	14
enumXvi	4	write-env	14
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8, 9
anskey*	11-14, 22	after	10, 11
enumet*	21	align	7, 21, 23
enumext*	4-15, 18	base-fix	8
enumext	4-16, 18, 21, 22	before*	9, 10
kenyans*	9	before	9
keyans*	4-15, 22	below*	9
keyanspic	4, 7, 8, 11-14, 16, 22	below	9
keyans	4-16, 22	check-ans	13
Environments:		columns-sep	4, 10, 21
Verbatim	14	columns	4, 9, 10, 21
center	5	first	10
description	5, 22	font	7
enumerate	1, 3, 5, 23	item-pos*	5, 6
figure	5	item-sym*	5, 6
flushleft	5	itemindent	9, 10
flushright	5	itemsep	8
itemize	5, 22	label-indent	9
list	3, 5, 9, 23	label-pos	16, 17
minipage	3-5, 8-11, 21, 23	label-sep	16
multicols	3, 4, 10, 21	labelsep	3-7, 9, 10, 12, 21, 22
quotation	5	labelwidth	3, 4, 6, 7, 9, 10, 12, 21, 22
quote	5	labelwith	5
tabbing	5	label	7, 8, 10, 15, 21-23
table	5	labewdith	9
task	5	layout-sep	16
trivlist	5	layout-sty	16
verbatim	5	layout-top	16
verse	5	list-indent	3, 9

list-offset	3, 9, 22, 23	Labels provide by enumext:	
listparindent	9, 10	\Alph*	7, 8, 15
mark-ans	12	\Roman*	7, 8
mark-pos	12	\alph*	7, 8
mark-ref	12	\arabic*	7, 8
mini-env	4, 9, 11, 21	\roman*	7, 8
mini-right*	7, 11	\labelsep	3, 7
mini-right	7, 11, 21	\labelwidth	3, 7
mini-sep	4, 11	\linewidth	11
mode-box	7	\listparindent	9
no-store	11, 13, 14, 22		
noitemsep	8		
nosep	8, 22	P	
overwrite	14	Packages:	
parsep	8, 10, 16	enumerate	23
partopsep	8	enumext	1-5, 7, 16, 21, 23, 24
ref	4, 8, 21	enumitem	3-5, 23
resume*	7, 10, 11	fancyvrb	14
resume	7, 10, 11	footnotehyper	5
rightmargin	9	geometry	21
save-ans	4, 6, 10-18, 22	graphicx	21
save-key	10-12, 18	hyperref	4, 5, 10-14, 21, 23
save-ref	4, 7, 12-14, 17, 22	l3keys	7
save-sep	12, 22	l3prop	1, 23
series	7, 10, 11	l3seq	1, 23
show-ans	12, 22	multicol	1, 2, 4, 21, 23
show-length	8	scontents	1, 2, 14, 21
show-pos	12, 13, 17	task	5, 6
start*	10	unicode-math	21
start	10	xsim	2
topsep	8, 9, 16	\parsep	8
widest	7	\partopsep	8
wrap-ans	12		
wrap-label*	7, 23	R	
wrap-label	7, 21, 23	\raggedcolumns	4
wrap-opt	12	\ref	4
write-env	14	\rightmargin	9
L		T	
\label	4	\topsep	8

13 Implementation

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

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

13.1 General conventions

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

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

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

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

13.2 Initial set up

Start the DocStrip guards.

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

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

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

13.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-10-06}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

13.4 Definition of variables

Variables that do not appear in this section are created by means of `\keys_define:nn` or some function described below.

```

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

```

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

```

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

```

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

```

\l__enumext_starred_bool
\g__enumext_starred_bool
\l__enumext_starred_first_bool
\l__enumext_standar_bool
\g__enumext_standar_bool
\l__enumext_standar_first_bool
\l__enumext_anskey_env_bool
\l__enumext_keyans_env_bool
\g__enumext_start_line_tl
\g__enumext_envir_name_tl
\l__enumext_envir_name_tl

```

Internal variables used by functions `__enumext_is_not_nested:`, `__enumext_is_on_first_level:` and `__enumext_keyans_name_and_start:` (§13.5.1).

```

34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_anskey_env_bool
41 \bool_new:N \l__enumext_keyans_env_bool
42 \tl_new:N \g__enumext_start_line_tl
43 \tl_new:N \g__enumext_envir_name_tl
44 \tl_new:N \l__enumext_envir_name_tl

```

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

```

\l__enumext_counter_i_tl
\l__enumext_counter_ii_tl
\l__enumext_counter_iii_tl
\l__enumext_counter_iv_tl
\l__enumext_counter_v_tl
\l__enumext_counter_vi_tl
\l__enumext_counter_vii_tl
\l__enumext_counter_viii_tl

```

Variables to store the “*name of the counters*” `enumXi`, `enumXii`, `enumXiii` and `enumXiv` for `enumext` environment, `enumXv` for `keyans` environment and `enumXvi` for the `keyanspic` environment. The counters `enumXvii` and `enumXviii` are used by `enumext*` and `keyans*` environments.

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§13.11) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.14).

```

45 \cs_set_protected:Npn \__enumext_tmp:n #1
46 {
47   \tl_new:c { \l__enumext_counter_#1_tl }
48 }
49 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

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

```

50 \tl_const:Nn \c__enumext_counter_style_tl
51 { { arabic } { roman } { Roman } { alph } { Alph } }
52 \tl_new:N \l__enumext_ref_key_arg_tl
53 \tl_new:N \l__enumext_ref_the_count_tl
54 \cs_set_protected:Npn \__enumext_tmp:n #1
55 {
56   \tl_new:c { \l__enumext_renew_the_count_#1_tl }
57   \tl_new:c { \l__enumext_the_counter_#1_tl }
58   \tl_set:ce { \l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
59 }
60 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_resume_int
\g__enumext_resume_vii_int
\l__enumext_resume_name_tl
\l__enumext_resume_active_bool
\g__enumext_starred_series_tl
\g__enumext_standar_series_tl

```

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

```

61 \int_new:N \g__enumext_resume_int
62 \int_new:N \g__enumext_resume_vii_int
63 \tl_new:N \l__enumext_resume_name_tl
64 \bool_new:N \l__enumext_resume_active_bool
65 \tl_new:N \g__enumext_standar_series_tl
66 \tl_new:N \g__enumext_starred_series_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default *label style* and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§13.15) and `label` (§13.13) keys.

```

67 \dim_new:N \l__enumext_current_widest_dim
68 \tl_new:N \g__enumext_counter_styles_tl
69 \tl_new:N \g__enumext_widest_label_tl
70 \box_new:N \l__enumext_label_width_by_box

```

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

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

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace`:NNNNNNNNNN (§13.38.1).

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

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

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

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

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

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

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

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

```
87 \int_new:N \g__enumext_minipage_stat_int
88 \skip_new:N \l__enumext_minipage_temp_skip
89 \skip_new:N \l__enumext_minipage_left_skip
90 \skip_new:N \l__enumext_minipage_right_skip
91 \skip_new:N \l__enumext_minipage_after_skip
92 \skip_new:N \g__enumext_minipage_right_skip
93 \skip_new:N \g__enumext_minipage_after_skip
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95 {
96   \dim_new:c { \l__enumext_minipage_left_#1_dim }
97   \bool_new:c { \l__enumext_minipage_active_#1_bool }
98 }
99 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

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

```
100 \cs_set_protected:Npn \__enumext_tmp:n #1
101 {
102   \bool_new:c { \l__enumext_wrap_label_#1_bool }
103   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
104   \int_new:c { \l__enumext_start_#1_int }
105   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
106   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
107   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
108   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
109   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
110 }
111 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```

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

```

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

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

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

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

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

```

112 \bool_new:N \l__enumext_store_active_bool
113 \tl_new:N \l__enumext_store_name_tl
114 \tl_new:N \g__enumext_store_name_tl
115 \tl_new:N \l__enumext_store_anskey_arg_tl
116 \tl_new:N \l__enumext_store_anskey_env_tl
117 \tl_new:N \l__enumext_store_anskey_opt_tl
118 \tl_new:N \l__enumext_store_current_label_tl
119 \tl_new:N \l__enumext_store_current_opt_arg_tl
120 \tl_new:N \l__enumext_store_current_label_tmp_tl

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

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

```

121 \tl_new:N \l__enumext_setkey_tmpa_tl
122 \tl_new:N \l__enumext_setkey_tmpb_tl
123 \int_new:N \l__enumext_setkey_tmpa_int
124 \seq_new:N \l__enumext_setkey_tmpa_seq
125 \seq_new:N \l__enumext_setkey_tmpb_seq

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

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

```

126 \tl_new:N \l__enumext_meta_path_tl
127 \seq_new:N \l__enumext_foreach_print_seq
128 \tl_new:N \l__enumext_foreach_name_prop_tl
129 \tl_new:N \g__enumext_foreach_default_keys_tl

```

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

```

\l__enumext_print_keyans_starred_tl
\l__enumext_print_keyans_star_bool
\l__enumext_mark_position_str
\g__enumext_item_symbol_aux_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool

```

Internal variables used by command `\printkeyans` (§13.47), `show-pos` key (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.2) and “*storing structure*”.

```

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

```

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

```

\l__enumext_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_anspic_label_above_bool
\l__enumext_anspic_mini_pos_str
\g__enumext_keyans_pic_parsep_skip
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim

```

Internal variables used by `keyanspic` environment and `\anspic` command (§13.42.1).

```

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

```

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

Internal variables used by “*internal check answer*” mechanism (§13.26.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

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

```

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

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

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

```

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

Internal variables used by `save-ref` key (§13.27). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the $\langle labels \rangle$ defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§13.7) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

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

```

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

Internal variables used for redefinition of `\footnote` (§13.8).

```

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

```

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

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

```

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

```

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

`\c__enumext_all_envs_clist` An internal `clist-var` variable to run with `__enumext_tmp:n`.

```
195 \clist_const:Nn \c__enumext_all_envs_clist
196 {
197   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
198   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
199 }
```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

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

`\seq_use:NV`

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

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

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

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

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

`__enumext_after_env:nn` A internal “hook” functions for execute code `mini-right` and `mini-right*` keys outside the `enumext*` and `keyans*` environments and print `check-ans` outside the `enumext` and `enumext*` environments.

`__enumext_before_env:nn`

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

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

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

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

(End of definition for `__enumext_level:`.)

`__enumext_if_is_int:nT` A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys.

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

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?](#).

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

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

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

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


(End of definition for `__enumext_regex_counter_style:`.)

`__enumext_show_length:nnn`

Internal function used by `show-length` key to show “*all lengths*” calculated and use in `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

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

```

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

`__enumext_unskip_unkern:`

The function `__enumext_unskip_unkern:` will remove the last `<skip>` or `<kern>` at execution time using the values `11` and `12` of `\lastnodetype` to apply `\unskip` or `\unkern` according to the case.

```

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

```

(End of definition for `__enumext_unskip_unkern:`.)

13.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`

The function `__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “*true*” only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

`__enumext_is_on_first_level:`

```

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

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) to “*true*” only if the environment is not nested and we are in the “*first level*” of it . We will also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name* of each environment in the variable `\g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

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

```

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

__enumext_keyans_name_and_start:

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

```

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

```

(End of definition for __enumext_keyans_name_and_start:.)

13.5.2 Utilities for log and terminal

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

```

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

```

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

The function `__enumext_log_global_vars:` will be passed to the function `__enumext_execute_after_env:` and write to the `.log` file the number of elements saved in the *prop list* and *sequence* created by the *save-ans* key along with the value of the integer variable created for the *resume* key.

```

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

```

The function `__enumext_log_answer_vars:` will be passed to the function `__enumext_execute_after_env:` and write to the `.log` file the number of items and answers along with the difference between them.

```

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

```

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

13.6 Copying list and minipage environments

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

```

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

```

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

```

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

```

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

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

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w
\__enumext_minipage:w
\__enumext_endminipage:

```

The functions `__enumext_start_list:nn` and `__enumext_stop_list:` correspond to copies of `\list` and `\endlist` from plain definition of `list`, the function `__enumext_item_std:w` is a copy of the `\item` command.

```

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

```

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

```

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

```

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

13.7 Compatibility with hyperref and footnotehyper

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

```

393 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
394 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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

```

395 \cs_new_protected:Nn \__enumext_after_hyperref:
396 {
397     \IfPackageLoadedTF { hyperref }
398     {
399         \msg_info:nnn { enumext } { package-load } { hyperref }
400         \bool_set_true:N \l__enumext_hyperref_bool
401         \IfHyperBoolean{hyperfootnotes}
402         {
403             \bool_set_true:N \l__enumext_footnotes_key_bool
404         }
405     }
406 }
407 { }

```

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`.

```

408 \bool_if:NT \l__enumext_footnotes_key_bool
409 {
410     \IfPackageLoadedTF { footnotehyper }
411     {
412         \msg_info:nnn { enumext } { package-load } { footnotehyper }
413     }
414     {
415         \bool_set_false:N \l__enumext_footnotes_key_bool
416     }
417 }

```

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.

```

418 \bool_if:NTF \l__enumext_hyperref_bool
419 {
420     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
421     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
422 }
423 {
424     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
425     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
426 }
427 }

```

(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: `__enumext_newlabel_arg_one_tl`

#2: `__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.

```

428 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
429 {
430   \protected@write \@auxout { }
431   {
432     \token_to_str:N \newlabel {#1}
433     {
434       {#2}
435       \bool_if:NT \__enumext_hyperref_bool
436       { { \thepage } {#2} {#1} }
437       { }
438     }
439   }
440   \__enumext_hypertarget:nn {#1} { }
441   \__enumext_phantomsection:
442 }

```

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

13.8 Redefining `\footnote` command

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

`__enumext_renew_footnote:`
`__enumext_print_footnote:`

```

443 \cs_new_protected:Nn \__enumext_footnotetext:nn
444 {
445   \footnotetext[#1]{#2}
446 }
447 \cs_new_protected:Nn \__enumext_renew_footnote:
448 {
449   \seq_gclear:N \g__enumext_footnote_arg_seq
450   \seq_gclear:N \g__enumext_footnote_int_seq
451   \RenewDocumentCommand \footnote { o +m }
452   {
453     \tl_if_novalue:nTF {##1}
454     {
455       \stepcounter{footnote}
456       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
457     }
458     {
459       \int_gset:Nn \g__enumext_footnote_int { ##1 }
460     }
461     \footnotemark [ \g__enumext_footnote_int ]
462     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
463     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
464   }
465 }
466 \cs_new_protected:Nn \__enumext_print_footnote:
467 {
468   \seq_if_empty:NF \g__enumext_footnote_int_seq
469   {
470     \seq_map_pairwise_function:NNN
471     \g__enumext_footnote_int_seq
472     \g__enumext_footnote_arg_seq
473     \__enumext_footnotetext:nn
474   }
475 }

```

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

`__enumext_renew_footnote_standar:`

`__enumext_print_footnote_standar:`

```

476 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
477 {

```

```

478     \bool_if:NT \g__enumext_standar_bool
479     {
480         \IfDocumentMetadataTF
481         {
482             \__enumext_renew_footnote:
483         }
484         {
485             \bool_if:NF \l__enumext_footnotes_key_bool
486             {
487                 \__enumext_renew_footnote:
488             }
489         }
490     }
491 }
492 \cs_new_protected:Nn \__enumext_print_footnote_standar:
493 {
494     \bool_if:NT \g__enumext_standar_bool
495     {
496         \IfDocumentMetadataTF
497         {
498             \__enumext_print_footnote:
499         }
500         {
501             \bool_if:NF \l__enumext_footnotes_key_bool
502             {
503                 \__enumext_print_footnote:
504             }
505         }
506     }
507 }
508 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
509 {
510     \bool_if:NT \g__enumext_starred_bool
511     {
512         \IfDocumentMetadataTF
513         {
514             \__enumext_renew_footnote:
515         }
516         {
517             \bool_if:NF \l__enumext_footnotes_key_bool
518             {
519                 \__enumext_renew_footnote:
520             }
521         }
522     }
523 }
524 \cs_new_protected:Nn \__enumext_print_footnote_starred:
525 {
526     \bool_if:NT \g__enumext_starred_bool
527     {
528         \IfDocumentMetadataTF
529         {
530             \__enumext_print_footnote:
531         }
532         {
533             \bool_if:NF \l__enumext_footnotes_key_bool
534             {
535                 \__enumext_print_footnote:
536             }
537         }
538     }
539 }

```

(End of definition for __enumext_renew_footnote_standar: and __enumext_print_footnote_standar:.)

13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

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

use only. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§13.39) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.44)

```

540 \cs_new_protected:Nn \__enumext_internal_mini_page:
541 {
542   \int_compare:nNtT { \__enumext_level_int } = { 0 }
543   {
544     \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
545     {
546       \__enumext_renew_footnote_standar:
547       \__enumext_minipage:w [ t ] { ##1 }
548       \legacy_if_gset_false:n { @minipage }
549       \skip_vertical:N \c_zero_skip
550     }
551     {
552       \skip_vertical:N \c_zero_skip
553       \__enumext_endminipage:
554       \__enumext_print_footnote_standar:
555     }
556   }
557 }

```

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

13.10 Definition of public dimension

The package `enumext` only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. This dimension is set in all environments and is only used by the `wrap-ans` key at its default value.

```

558 \dim_zero_new:N \itemwidth

```

13.11 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.

```

559 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
560 {
561   \cs_if_exist:cTF { c@ #2 }
562   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
563   {
564     \tl_set:Nn #1 { #2 }
565     \newcounter { #2 }
566   }
567 }

```

(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      568 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     569 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    570 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     571 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXvii    572 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXviii   573 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
             574 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
             575 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

13.12 Definition of labels

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

`__enumext_register_counter_style:Nn`

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

```
576 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
577 {
578   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
579   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
580 }
581 \__enumext_register_counter_style:Nn \arabic { 0 }
582 \__enumext_register_counter_style:Nn \Alph { M }
583 \__enumext_register_counter_style:Nn \alph { m }
584 \__enumext_register_counter_style:Nn \Roman { VIII }
585 \__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.

```
586 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
587 {
588   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
589   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
590 }
591 \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.

```
592 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
593 {
594   \tl_clear_new:N #1
595   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
596   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
597   \tl_map_inline:Nn \g__enumext_counter_styles_tl
598   {
599     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
600     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
601     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
602   }
603   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
604   { \tl_use:N \g__enumext_widest_label_tl }
605   \tl_set_eq:cN { the #2 } #1
606 }
607 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

13.13 Setting keys associated with label

When *tagged* PDF is active `\makelabel` is redefined using `\makebox` to work correctly (§13.34). From the user side it is convenient to have a key that allows using this redefinition with `\makebox` without having `\IfDocumentMetadataTF` active.

`mode-box`

We define the key `mode-box` only for the “first level” of `enumext` and `enumext*` environments.

```
608 \cs_set_protected:Npn \__enumext_tmp:n #1
609 {
610   \keys_define:nn { enumext / #1 }
611   {
612     mode-box .bool_set:N = \l__enumext_mode_box_bool,
613     mode-box .initial:n = false,
614     mode-box .value_forbidden:n = true,
615   }
616 }
617 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(End of definition for mode-box.)

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

```

618 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
619 {
620   \keys_define:nn { enumext / #1 }
621   {
622     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
623     font      .value_required:n = true,
624     labelsep   .dim_set:c = { l__enumext_labelsep_#2_dim },
625     labelsep   .initial:n  = {0.3333em},
626     labelsep   .value_required:n = true,
627     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
628     labelwidth .value_required:n = true,
629     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
630     wrap-label .initial:n  = {##1},
631     wrap-label .value_required:n = true,
632     wrap-label* .code:n = {
633       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
634       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
635     },
636     wrap-label* .value_required:n = true,
637   }
638 }
639 \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. For compatibility with *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`.

```

640 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
641 {
642   \keys_define:nn { enumext / #1 }
643   {
644     align .choice:,
645     align / left .code:n =
646       {
647         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
648         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
649         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
650       },
651     align / right .code:n =
652       {
653         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
654         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
655         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
656       },
657     align / center .code:n =
658       {
659         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
660         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
661         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
662       },
663     align / unknown .code:n =
664       \msg_error:nneee { enumext } { unknown-choice }
665       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
666     align .initial:n = left,
667     align .value_required:n = true,
668   }
669 }
670 \clist_map_inline:nn
671 {
672   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
673 }
674 { \__enumext_tmp:nn #1 }

```

```

675 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
676 {
677   \keys_define:nn { enumext / #1 }
678   {
679     align .choice:,
680     align / left .code:n = \str_set:cn { \__enumext_align_label#2_str } { l },
681     align / right .code:n = \str_set:cn { \__enumext_align_label#2_str } { r },
682     align / center .code:n = \str_set:cn { \__enumext_align_label#2_str } { c },
683     align / unknown .code:n =
684       \msg_error:nneee { enumext } { unknown-choice }
685       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
686     align .initial:n = left,
687     align .value_required:n = true,
688   }
689 }
690 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

13.14 Setting label and ref keys

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

13.14.1 Define and set label and ref keys for enumext environment

Here we set the default `<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
691 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
692 {
693   \keys_define:nn { enumext / #1 }
694   {
695     label .code:n = {
696       \__enumext_label_style:cvn { \__enumext_label#2_tl }
697       { \__enumext_counter#2_tl } {##1}
698       \dim_set_eq:cN { \__enumext_labelwidth#2_dim }
699       \__enumext_current_widest_dim
700     },
701     label .initial:n = #3,
702     label .value_required:n = true,
703     ref .code:n = \__enumext_standar_ref:n {##1},
704     ref .value_required:n = true,
705   }
706 }
707 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
708 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
709 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
710 \__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.

```

711 \cs_new_protected:Npn \__enumext_standar_ref:n #1
712 {
713   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
714   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
715   {
716     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
717   }
718   {
719     \tl_set_eq:Nc
720       \l__enumext_ref_the_count_tl { \__enumext_counter_ \__enumext_level: _tl }
721     \__enumext_regex_counter_style:
722     \tl_set_eq:Nc
723       \l__enumext_ref_the_count_tl { \__enumext_the_counter_ \__enumext_level: _tl }
724     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }

```

```

725         {
726         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
727         }
728     }
729 }

```

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

```

730 \cs_new_protected:Nn \__enumext_standar_ref:
731 {
732     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
733     {
734         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
735     }
736 }

```

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

13.14.2 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
737 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
738 {
739     \keys_define:nn { enumext / #1 }
740     {
741         label .code:n = {
742             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
743             { l__enumext_counter_#2_tl } {##1}
744             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
745             \l__enumext_current_widest_dim
746         },
747         label .initial:n = #3,
748         label .value_required:n = true,
749         ref .code:n = \__enumext_starred_ref:n {##1},
750         ref .value_required:n = true,
751     }
752 }
753 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
754 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

\__enumext_starred_ref:
755 \cs_new_protected:Npn \__enumext_starred_ref:n #1
756 {
757     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
758     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
759     {
760         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
761         {
762             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
763         }
764         {
765             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
766             \__enumext_regex_counter_style:
767             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
768             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
769             {
770                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
771                 }
772             }
773         }
774     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
775     {
776         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
777         {
778             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
779         }
780         {
781             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
782             \__enumext_regex_counter_style:

```

```

783         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
784         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
785         {
786             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
787         }
788     }
789 }
790 }

```

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.

```

791 \cs_new_protected:Nn \__enumext_starred_ref:
792 {
793     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
794     {
795         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
796         {
797             \tl_use:N \l__enumext_renew_the_count_vii_tl
798         }
799     }
800     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
801     {
802         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
803         {
804             \tl_use:N \l__enumext_renew_the_count_viii_tl
805         }
806     }
807 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
808 \keys_define:nn { enumext / keyans }
809 {
810     label .code:n = {
811         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
812         { \l__enumext_counter_v_tl } {#1}
813         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
814         \l__enumext_current_widest_dim
815         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
816         { \l__enumext_counter_vi_tl } {#1}
817         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
818         \l__enumext_current_widest_dim
819     },
820     label .initial:n = \Alph*,
821     label .value_required:n = true,
822     ref .code:n = \__enumext_keyans_ref:n {#1},
823     ref .value_required:n = true,
824 }

```

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

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

```

825 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
826 {
827     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
828     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
829     {
830         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
831     }
832     {
833         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
834         \__enumext_regex_counter_style:
835         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
836         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
837         {
838             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
839         }
840     }
841 }

```


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

```

842 \cs_new_protected:Nn \__enumext_keyans_ref:
843 {
844     \tl_if_empty:NF \__enumext_renew_the_count_v_tl
845     {
846         \tl_use:N \__enumext_renew_the_count_v_tl
847     }
848 }

```

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

13.15 Setting start, start* and widest keys

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

```

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

```

```

#1: \__enumext_label_X_tl
#2: \__enumext_start_X_int
#3: <integer or string>

```

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

```

849 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
850 {
851     \__enumext_if_is_int:nTF { #3 }
852     {
853         \int_set:Nn #2 {#3}
854     }
855     {
856         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
857         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
858         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
859         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
860     }
861 }
862 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

```

#1: The counter associated with the environment level
#2: \__enumext_label_X_tl
#3: \__enumext_labelwidth_X_dim
#4: <integer or string>

```

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

```

863 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
864 {
865     \__enumext_if_is_int:nTF {#4}
866     {
867         \setcounter{enumX#1} { #4 }
868     }
869     {
870         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
871         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
872         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
873         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
874     }
875     \__enumext_label_width_by_box:cv
876     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
877 }
878 \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.

`widest`

```

879 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
880 {

```

```

881 \keys_define:nn { enumext / #1 }
882 {
883   start* .code:n = {
884     \__enumext_start_from:ccn
885     { l__enumext_label_#2_tl }
886     { l__enumext_start_#2_int } {##1}
887   },
888   start* .value_required:n = true,
889   start .code:n = {
890     \__enumext_start_from:cce
891     { l__enumext_label_#2_tl }
892     { l__enumext_start_#2_int } { \int_eval:n {##1} }
893   },
894   start .initial:n = 1,
895   start .value_required:n = true,
896   widest .code:n = {
897     \__enumext_widest_from:nccn {#2}
898     { l__enumext_label_#2_tl }
899     { l__enumext_labelwidth_#2_dim } {##1}
900   },
901   widest .value_required:n = true,
902 }
903 }
904 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.16 Setting keys for vertical spaces

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

```

topsep  Define and set topsep, partopsep, parsep, itemsep, noitemsep and nosep keys for enumext, enumext*,
partopsep keyans and keyans* environments.
parsep
noitemsep
nosep
905 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
906 {
907   \keys_define:nn { enumext / #1 }
908   {
909     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
910     topsep .initial:n = {#3},
911     topsep .value_required:n = true,
912     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
913     partopsep .initial:n = {#4},
914     partopsep .value_required:n = true,
915     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
916     parsep .initial:n = {#5},
917     parsep .value_required:n = true,
918     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
919     itemsep .initial:n = {#6},
920     itemsep .value_required:n = true,
921     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
922     noitemsep .value_forbidden:n = true,
923     nosep .meta:n = {
924       itemsep = 0pt, parsep = 0pt,
925       topsep = 0pt, partopsep = 0pt,
926     },
927     nosep .value_forbidden:n = true,
928   }
929 }

```

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

```

930 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
931 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
932 { 4.0pt plus 2.0pt minus 1.0pt }
933 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
934 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
935 { 2.0pt plus 1.0pt minus 1.0pt }
936 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
937 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
938 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
939 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
940 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
941 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
942 { 2.0pt plus 1.0pt minus 1.0pt }
943 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }

```

```

944 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
945 { 4.0pt plus 2.0pt minus 1.0pt }
946 \__enumext_tmp:nnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
947 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
948 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.17 Setting base-fix key

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

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

```

base-fix
\__enumext_nested_base_line_fix:
949 \keys_define:nn { enumext / level-1 }
950 {
951   base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
952   base-fix .initial:n = false,
953   base-fix .value_forbidden:n = true,
954 }

```

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

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

```

955 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
956 {
957   \bool_lazy_all:nT
958   {
959     { \bool_if_p:N \l__enumext_starred_first_bool }
960     { \bool_if_p:N \l__enumext_base_line_fix_bool }
961     { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
962   }
963   {
964     \mode_leave_vertical:
965     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
966   }

```

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

```

967   \bool_lazy_and:nnT
968   { \bool_if_p:N \l__enumext_starred_first_bool }
969   { \bool_if_p:N \l__enumext_print_keyans_star_bool }
970   {
971     \mode_leave_vertical:
972     \skip_vertical:n { -\baselineskip }
973     \skip_vertical:N \c_zero_skip
974   }

```

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

```

975   \keys_set:nn { enumext / level-1 }
976   {
977     topsep = 0pt, above = 0pt, above* = 0pt,
978   }
979   \bool_set_false:N \l__enumext_base_line_fix_bool
980 }

```

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

13.18 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```

981 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
982 {
983   \keys_define:nn { enumext / #1 }
984   {
985     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
986     itemindent .value_required:n = true,
987     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
988     rightmargin .value_required:n = true,
989     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
990     listparindent .value_required:n = true,
991     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
992     list-offset .value_required:n = true,
993     list-indent .code:n =
994       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
995       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
996     list-indent .value_required:n = true,
997   }
998 }
999 \clist_map_inline:nn
1000 {
1001   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1002 }
1003 { \__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.

```

1004 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1005 {
1006   \keys_define:nn { enumext / #1 }
1007   {
1008     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1009     itemindent .value_required:n = true,
1010     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1011     rightmargin .value_required:n = true,
1012     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1013     listparindent .value_required:n = true,
1014     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1015     list-offset .value_required:n = true,
1016     list-indent .meta:n = { list-offset = ##1 },
1017     list-indent .value_required:n = true,
1018   }
1019 }
1020 \clist_map_inline:nn
1021 {
1022   {enumext*}{vii}, {keyans*}{viii}
1023 }
1024 { \__enumext_tmp:nn #1 }

```

13.18.1 Functions for setting the fake `itemindent`

__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_viii:

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.

```

1025 \cs_set_protected:Nn \__enumext_fake_item_indent:
1026 {
1027   \dim_compare:nNnT
1028     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1029     >
1030     { \c_zero_dim }
1031   {
1032     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1033     {
1034       \exp_not:N \mode_leave_vertical:
1035       \exp_not:n { \skip_horizontal:n }
1036       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }

```

```

1037         \exp_not:N \ignorespaces
1038     }
1039 }
1040 }
1041 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1042 {
1043     \dim_compare:nNnT
1044     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1045     {
1046         \tl_set:Nc \l__enumext_fake_item_indent_v_tl
1047         {
1048             \exp_not:N \mode_leave_vertical:
1049             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1050             \exp_not:N \ignorespaces
1051         }
1052     }
1053 }
1054 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1055 {
1056     \dim_compare:nNnT
1057     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1058     {
1059         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
1060         {
1061             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1062             \exp_not:N \ignorespaces
1063         }
1064     }
1065 }
1066 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1067 {
1068     \dim_compare:nNnT
1069     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1070     {
1071         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
1072         {
1073             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1074             \exp_not:N \ignorespaces
1075         }
1076     }
1077 }

```

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

13.19 Setting show-length key

show-length

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

```

1078 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1079 {
1080     \keys_define:nn { enumext / #1 }
1081     {
1082         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
1083         show-length .initial:n = false,
1084     }
1085 }
1086 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20 Setting before, after and first keys

before

before*

after

first

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

```

1087 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1088 {
1089     \keys_define:nn { enumext / #1 }
1090     {
1091         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
1092         before .value_required:n = true,

```

```

1093     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1094     before* .value_required:n = true,
1095     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1096     after .value_required:n = true,
1097     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1098     first .value_required:n = true,
1099   }
1100 }
1101 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20.1 Functions for before, after and first keys in enumext

The function `__enumext_before_args_exec:` executes the `{⟨code⟩}` set by the *before** key “before” the `enumext` environment is started. The `{⟨code⟩}` is executed “without” knowing any definition of the `{⟨arg two⟩}` of the list: `{⟨code⟩}\list{⟨arg one⟩}{⟨arg two⟩}`.

```

1102 \cs_new_protected:Nn \__enumext_before_args_exec:
1103 {
1104   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1105 }

```

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⟩}`

```

1106 \cs_new_protected:Nn \__enumext_before_keys_exec:
1107 {
1108   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1109 }

```

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

```

1110 \cs_new_protected:Nn \__enumext_after_stop_list:
1111 {
1112   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1113 }

```

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.`

```

1114 \cs_new_protected:Nn \__enumext_after_args_exec:
1115 {
1116   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1117 }

```

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

13.20.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
1118 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1119 {
1120   \tl_use:N \l__enumext_before_starred_key_v_tl
1121 }
1122 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1123 {
1124   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1125 }
1126 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1127 {
1128   \tl_use:N \l__enumext_after_stop_list_v_tl
1129 }
1130 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1131 {
1132   \tl_use:N \l__enumext_after_list_args_v_tl
1133 }

```

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

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

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1134 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1135 {
1136   \tl_use:N \l__enumext_before_starred_key_vii_tl
1137 }
1138 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1139 {
1140   \tl_use:N \l__enumext_before_starred_key_viii_tl
1141 }
1142 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1143 {
1144   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1145 }
1146 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1147 {
1148   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1149 }
1150 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1151 {
1152   \tl_use:N \l__enumext_after_stop_list_vii_tl
1153 }
1154 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1155 {
1156   \tl_use:N \l__enumext_after_stop_list_viii_tl
1157 }
1158 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1159 {
1160   \tl_use:N \l__enumext_after_list_args_vii_tl
1161 }
1162 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1163 {
1164   \tl_use:N \l__enumext_after_list_args_viii_tl
1165 }

```

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

13.21 Setting keys for multicol and minipage

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

```

1166 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1167 {
1168   \keys_define:nn { enumext / #1 }
1169   {
1170     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
1171     mini-env .value_required:n = true,
1172     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
1173     mini-sep .initial:n = 0.3333em,
1174     mini-sep .value_required:n = true,
1175     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
1176     columns-sep .value_required:n = true,
1177     columns .int_set:c = { \l__enumext_columns_#2_int },
1178     columns .initial:n = 1,
1179     columns .value_required:n = true,
1180   }
1181 }
1182 \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.

```

1183 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1184 {
1185   \keys_define:nn { enumext / #1 }
1186   {
1187     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1188     mini-right .value_required:n = true,
1189     mini-right* .code:n = {

```

```

1190         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1191         \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1192     },
1193     mini-right* .value_required:n = true,
1194 }
1195 }
1196 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.22 Adjustment of vertical spaces for 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.

13.22.1 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*.

```

1197 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1198 {
1199     \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1200     {
1201         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1202     }
1203     \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1204     {
1205         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1206     }
1207     \__enumext_add_pre_parsep:
1208 }

```

(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*.

```

1209 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1210 {
1211     \int_case:nn { \l__enumext_level_int }
1212     {
1213         { 2 }{
1214             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1215             {
1216                 \skip_add:Nn \l__enumext_multicols_above_ii_skip
1217                 {
1218                     \l__enumext_parsep_i_skip
1219                 }
1220             }
1221         }

```

```

1222     { 3 }{
1223         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1224         {
1225             \skip_add:Nn \l__enumext_multicols_above_iii_skip
1226             {
1227                 \l__enumext_parsep_ii_skip
1228             }
1229         }
1230     }
1231     { 4 }{
1232         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1233         {
1234             \skip_add:Nn \l__enumext_multicols_above_iv_skip
1235             {
1236                 \l__enumext_parsep_iii_skip
1237             }
1238         }
1239     }
1240 }
1241 }

```

(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*.

```

1242 \cs_new_protected:Nn \l__enumext_multi_addvspace:
1243 {
1244     \l__enumext_multi_set_vskip:
1245     \mode_if_vertical:T
1246     {
1247         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1248         {
1249             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1250         }
1251         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1252         {
1253             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1254         }
1255     }
1256     \par\nopagebreak
1257     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1258 }

```

(End of definition for `\l__enumext_multi_addvspace:`)

13.22.2 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:`

```

1259 \cs_new_protected:Nn \l__enumext_keyans_multi_set_vskip:
1260 {
1261     \skip_set:Nn \l__enumext_multicols_above_v_skip
1262     {
1263         \l__enumext_topsep_v_skip
1264     }
1265     \skip_set:Nn \l__enumext_multicols_below_v_skip
1266     {
1267         \l__enumext_topsep_v_skip
1268     }
1269 }
1270 \cs_new_protected:Nn \l__enumext_keyans_multi_addvspace:
1271 {
1272     \l__enumext_keyans_multi_set_vskip:
1273     \mode_if_vertical:T
1274     {
1275         \skip_add:Nn \l__enumext_multicols_above_v_skip
1276         {
1277             \skip_use:N \l__enumext_partopsep_v_skip
1278         }
1279         \skip_add:Nn \l__enumext_multicols_below_v_skip

```

```

1280     {
1281         \skip_use:N \l__enumext_partopsep_v_skip
1282     }
1283 }
1284 \par\nopagebreak
1285 \addvspace{ \l__enumext_multicols_above_v_skip }
1286 }

```

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

13.23 Adjustment of vertical spaces for minipage

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



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 $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$. Depending on these cases, small adjustments must be made using `\vspace` and `\addvspace` to obtain the “desired vertical spacing”.

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

13.23.1 Adjustment of vertical spaces for minipage in enumext

```

\__enumext_minipage_set_skip:
\__enumext_minipage_add_space:

```

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

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

```

1287 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1288 {
1289     \skip_set:Nn \l__enumext_minipage_right_skip
1290     {
1291         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1292     }
1293     \mode_if_vertical:T
1294     {
1295         \skip_add:Nn \l__enumext_minipage_right_skip
1296         {
1297             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1298         }
1299     }
1300     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip

```

We will adjust the values `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` and call the function `__enumext_pre_itemsep_skip:`.

```

1301     \skip_set_eq:cN
1302     { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1303     \skip_set_eq:cN
1304     { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1305     \__enumext_pre_itemsep_skip:

```

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

```

1306     \int_compare:nNnT
1307     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1308     {
1309         \skip_zero:N \topskip

```

```

1310     \skip_set_eq:Nc \multicolsep { \__enumext_multicols_above_ \__enumext_level: _skip }
1311   }
1312 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_mini_page` environment, taking into account whether T_EX is in *horizontal mode* or *vertical mode*. Here we use the plain T_EX 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*.

```

1313 \cs_new_protected:Nn \__enumext_minipage_add_space:
1314 {
1315   \__enumext_minipage_set_skip:
1316   \__enumext_unskip_unkern:
1317   \mode_if_vertical:TF
1318   {
1319     \nopagebreak\nointerlineskip
1320   }
1321   {
1322     \par\nopagebreak\nointerlineskip
1323     \skip_zero:c { \__enumext_partopsep_ \__enumext_level: _skip }
1324   }
1325   \int_compare:nNnTF
1326   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
1327   {
1328     \addvspace{ 0.445\box_ht:N \strutbox }
1329   }
1330   {
1331     \addvspace{ 0.250\box_ht:N \strutbox }
1332   }
1333 }

```

(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.

```

1334 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1335 {
1336   \int_case:nn { \__enumext_level_int }
1337   {
1338     { 2 }{
1339       \skip_if_eq:nnTF
1340       { \__enumext_itemsep_i_skip } { \__enumext_minipage_after_skip }
1341       {
1342         \skip_set:Nn \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1343         \skip_set:Nn \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1344       }
1345       {
1346         \dim_compare:nNnT
1347         { \__enumext_itemsep_i_skip } < { \__enumext_minipage_after_skip }
1348         {
1349           \skip_sub:Nn
1350           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1351           \skip_sub:Nn
1352           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }
1353           \skip_add:Nn
1354           \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1355           \skip_add:Nn
1356           \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1357         }
1358         \dim_compare:nNnT
1359         { \__enumext_itemsep_i_skip } > { \__enumext_minipage_after_skip }
1360         {
1361           \skip_set:Nn \__enumext_minipage_temp_skip
1362           {
1363             \__enumext_itemsep_i_skip - \__enumext_minipage_after_skip
1364           }
1365           \skip_sub:Nn
1366           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1367           \skip_sub:Nn
1368           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }

```

```

1369         \skip_add:Nn
1370         \l__enumext_minipage_after_skip
1371         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1372     \skip_add:Nn
1373     \l__enumext_multicols_below_ii_skip
1374     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1375 }
1376 }
1377 }
1378 { 3 }{
1379     \skip_if_eq:nnTF
1380     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1381     {
1382         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1383         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1384     }
1385     {
1386         \dim_compare:nNnT
1387         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1388         {
1389             \skip_sub:Nn
1390             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1391             \skip_sub:Nn
1392             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1393             \skip_add:Nn
1394             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1395             \skip_add:Nn
1396             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1397         }
1398         \dim_compare:nNnT
1399         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1400         {
1401             \skip_set:Nn \l__enumext_minipage_temp_skip
1402             {
1403                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1404             }
1405             \skip_sub:Nn
1406             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1407             \skip_sub:Nn
1408             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1409             \skip_add:Nn
1410             \l__enumext_minipage_after_skip
1411             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1412             \skip_add:Nn
1413             \l__enumext_multicols_below_iii_skip
1414             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1415         }
1416     }
1417 }
1418 { 4 }{
1419     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1420     {
1421         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1422         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1423     }
1424     {
1425         \dim_compare:nNnT
1426         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1427         {
1428             \skip_sub:Nn
1429             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1430             \skip_sub:Nn
1431             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1432             \skip_add:Nn
1433             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1434             \skip_add:Nn
1435             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1436         }
1437         \dim_compare:nNnT
1438         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1439         {

```



```

1440         \skip_set:Nn \l__enumext_minipage_temp_skip
1441         {
1442             \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1443         }
1444         \skip_sub:Nn
1445             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1446         \skip_sub:Nn
1447             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1448         \skip_add:Nn
1449             \l__enumext_minipage_after_skip
1450             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1451         \skip_add:Nn
1452             \l__enumext_multicols_below_iv_skip
1453             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1454     }
1455 }
1456 }
1457 }
1458 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

13.23.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

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

```

1459 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1460 {
1461     \skip_zero:N \l__enumext_minipage_after_skip
1462     \skip_zero:N \l__enumext_minipage_left_skip
1463     \skip_zero:N \l__enumext_minipage_right_skip
1464     \skip_set:Nn \l__enumext_minipage_right_skip
1465     {
1466         \l__enumext_topsep_v_skip
1467     }
1468     \mode_if_vertical:T
1469     {
1470         \skip_add:Nn \l__enumext_minipage_right_skip
1471         {
1472             \l__enumext_partopsep_v_skip
1473         }
1474     }
1475     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1476     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1477     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1478     \__enumext_keyans_pre_itemsep_skip:
1479     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1480     {
1481         \skip_zero:N \topskip
1482         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1483     }
1484 }
1485 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1486 {
1487     \__enumext_keyans_minipage_set_skip:
1488     \__enumext_unskip_unkern:
1489     \mode_if_vertical:TF
1490     {
1491         \nopagebreak\nointerlineskip
1492     }
1493     {
1494         \par\nopagebreak\nointerlineskip
1495         \skip_zero:N \l__enumext_partopsep_v_skip
1496     }
1497     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1498     {
1499         \addvspace{ 0.445\box_ht:N \strutbox }
1500     }
1501     {
1502         \addvspace{ 0.250\box_ht:N \strutbox }
1503     }

```

```

1504 }
1505 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1506 {
1507   \skip_if_eq:nnTF
1508   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1509   {
1510     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1511     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1512   }
1513   {
1514     \dim_compare:nNnT
1515     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1516     {
1517       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1518       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1519       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1520       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1521     }
1522     \dim_compare:nNnT
1523     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1524     {
1525       \skip_set:Nn \l__enumext_minipage_temp_skip
1526       {
1527         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1528       }
1529       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1530       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1531       \skip_add:Nn \l__enumext_minipage_after_skip
1532       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1533       \skip_add:Nn \l__enumext_multicols_below_v_skip
1534       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1535     }
1536   }
1537 }

```

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

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1538 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1539 {
1540   \skip_zero_new:N \l__enumext_minipage_left_skip
1541   \skip_gzero_new:N \g__enumext_minipage_right_skip
1542   \skip_gzero_new:N \g__enumext_minipage_after_skip
1543   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1544   {
1545     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1546     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1547   }
1548   {
1549     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1550     \skip_gset:Nn \g__enumext_minipage_right_skip
1551     {
1552       \l__enumext_topsep_vii_skip
1553     }
1554     \skip_gset:Nn \g__enumext_minipage_after_skip
1555     {
1556       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1557     }
1558   }
1559 }
1560 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1561 {
1562   \skip_zero_new:N \l__enumext_minipage_after_skip
1563   \skip_zero_new:N \l__enumext_minipage_left_skip
1564   \skip_zero_new:N \l__enumext_minipage_right_skip
1565   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1566   {

```

```

1567     \skip_set:Nn \l__enumext_minipage_left_skip
1568     {
1569         0.5\box_dp:N \strutbox
1570     }
1571     \skip_set:Nn \l__enumext_minipage_right_skip
1572     {
1573         \l__enumext_partopsep_viii_skip
1574     }
1575     \skip_set:Nn \l__enumext_minipage_after_skip
1576     {
1577         1.6\box_dp:N \strutbox
1578     }
1579 }
1580 {
1581     \skip_set:Nn \l__enumext_minipage_left_skip
1582     {
1583         0.5875\box_dp:N \strutbox
1584     }
1585     \skip_set:Nn \l__enumext_minipage_right_skip
1586     {
1587         \l__enumext_topsep_viii_skip
1588     }
1589     \skip_set:Nn \l__enumext_minipage_after_skip
1590     {
1591         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1592     }
1593 }
1594 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1595 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1596 {
1597     \__enumext_mini_set_vskip_vii:
1598     \par\nopagebreak
1599     \addvspace { \l__enumext_minipage_left_skip }
1600 }
1601 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1602 {
1603     \__enumext_mini_set_vskip_viii:
1604     \par\nopagebreak
1605     \addvspace { \l__enumext_minipage_left_skip }
1606 }

```

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

13.23.4 The command `\miniright`

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

`\miniright`

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

```

1607 \NewDocumentCommand \miniright { s }
1608 {
1609     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1610     {
1611         \msg_error:nnn { enumext } { wrong-miniright-place }
1612     }
1613     % outside
1614     \bool_lazy_and:nnT
1615     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }

```

```

1616     { \int_compare:p:nNn { \l__enumext_level_h_int } = { 0 } }
1617     {
1618       \msg_error:nnn { enumext } { wrong-miniright-place }
1619     }
1620     % starred env
1621     \bool_if:NT \l__enumext_starred_bool
1622     {
1623       \msg_error:nnn { enumext } { wrong-miniright-starred }
1624     }
1625     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1626     {
1627       \__enumext_keyans_mini_right_cmd:n {#1}
1628     }
1629     { \__enumext_mini_right_cmd:n {#1} }
1630   }

```

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

`__enumext_mini_right_cmd:n`

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

```

1631 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1632 {
1633   \dim_compare:nNnTF
1634   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1635   {
1636     \__enumext_multicols_stop:
1637     \int_compare:nNnT
1638     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1639     {
1640       \par\addvspace{ \l__enumext_minipage_after_skip }
1641     }
1642     \end__enumext_mini_page
1643     \hfill
1644     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1645     \par\nointerlineskip
1646     \addvspace { \l__enumext_minipage_right_skip }
1647     \bool_if:nF {#1}
1648     {
1649       \centering
1650     }
1651     \int_gzero:N \g__enumext_minipage_stat_int
1652   }
1653   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1654   % paranoia
1655   \RenewDocumentCommand \miniright { s }
1656   {
1657     \msg_error:nn { enumext } { many-miniright-used }
1658   }
1659 }

```

(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.

```

1660 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1661 {
1662   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1663   {
1664     \__enumext_keyans_multicols_stop:
1665     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1666     {
1667       \par\addvspace{ \l__enumext_minipage_after_skip }
1668     }
1669     \end__enumext_mini_page
1670     \hfill
1671     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }

```

```

1672         \par\nointerlineskip
1673         \addvspace { \l__enumext_minipage_right_skip }
1674         \bool_if:nF {#1}
1675         {
1676             \centering
1677         }
1678         \int_gzero:N \g__enumext_minipage_stat_int
1679     }
1680     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1681 % paranoia
1682 \RenewDocumentCommand \miniright { s }
1683 {
1684     \msg_error:nn { enumext } { many-miniright-used }
1685 }
1686 }

```

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

13.24 Setting above and below keys

While having controlled the *vertical spaces* within the `enumext` and `keyans` environments when using the `columns` or `mini-env` keys, sometimes the “*vertical spaces above*” or “*vertical spaces below*” the environments are not as expected and it is necessary to be able to apply a “*fine correction*” to these. As I have not been able to correct these *glitches*, the best option is to leave a couple of `\keys` dedicated to this purpose, in this case it is best to use `\vspace` or `\vspace*` when convenient.

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

```

1687 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1688 {
1689     \keys_define:nn { enumext / #1 }
1690     {
1691         above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1692         above .value_required:n = true,
1693         above* .code:n = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1694             \keys_set:nn { enumext / #1 } { above = {##1} },
1695         above* .value_required:n = true,
1696         below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1697         below .value_required:n = true,
1698         below* .code:n = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1699             \keys_set:nn { enumext / #1 } { below = {##1} },
1700         below* .value_required:n = true,
1701     }
1702 }
1703 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.24.1 Functions for above and below keys in enumext

`__enumext_vspace_above:` The function `__enumext_vspace_above:` apply the *vertical space above* the `enumext` environment set by the `above*` and `above` keys.

```

1704 \cs_new_protected:Nn \__enumext_vspace_above:
1705 {
1706     \skip_if_eq:nnF
1707     { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1708     {
1709         \bool_if:cTF { \l__enumext_vspace_a_star_ \__enumext_level: _bool }
1710         {
1711             \vspace*{ \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1712         }
1713         {
1714             \vspace { \skip_use:c { \l__enumext_vspace_above_ \__enumext_level: _skip } }
1715         }
1716     }
1717 }

```

(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.

```

1718 \cs_new_protected:Nn \__enumext_vspace_below:
1719 {

```

```

1720 \skip_if_eq:nnF
1721 { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1722 {
1723   \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1724   {
1725     \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1726   }
1727   {
1728     \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1729   }
1730 }
1731 }

```

(End of definition for `__enumext_vspace_below:`.)

13.24.2 Functions for above and below keys in keyans

`__enumext_vspace_above_v:`

The function `__enumext_vspace_above_v:` apply the *vertical space above* the `keyans` environment set by the `above` and `above*` keys.

```

1732 \cs_new_protected:Nn \__enumext_vspace_above_v:
1733 {
1734   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1735   {
1736     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1737     {
1738       \vspace*{ \l__enumext_vspace_above_v_skip }
1739     }
1740     { \vspace { \l__enumext_vspace_above_v_skip } }
1741   }
1742 }

```

(End of definition for `__enumext_vspace_above_v:`.)

`__enumext_vspace_below_v:`

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

```

1743 \cs_new_protected:Nn \__enumext_vspace_below_v:
1744 {
1745   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1746   {
1747     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1748     {
1749       \vspace*{ \l__enumext_vspace_below_v_skip }
1750     }
1751     { \vspace { \l__enumext_vspace_below_v_skip } }
1752   }
1753 }

```

(End of definition for `__enumext_vspace_below_v:`.)

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

`__enumext_vspace_above_vii:`

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

`__enumext_vspace_above_viii:`

```

1754 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1755 {
1756   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1757   {
1758     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1759     {
1760       \vspace*{ \l__enumext_vspace_above_vii_skip }
1761     }
1762     { \vspace { \l__enumext_vspace_above_vii_skip } }
1763   }
1764 }
1765 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1766 {
1767   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1768   {
1769     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1770     {
1771       \vspace*{ \l__enumext_vspace_above_viii_skip }
1772     }
1773     { \vspace { \l__enumext_vspace_above_viii_skip } }

```



```

1774     }
1775 }

```

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

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_viii:

```

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

```

1776 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1777 {
1778   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1779   {
1780     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1781     {
1782       \vspace*{ \l__enumext_vspace_below_vii_skip }
1783     }
1784     { \vspace { \l__enumext_vspace_below_vii_skip } }
1785   }
1786 }
1787 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1788 {
1789   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1790   {
1791     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1792     {
1793       \vspace*{ \l__enumext_vspace_below_viii_skip }
1794     }
1795     { \vspace { \l__enumext_vspace_below_viii_skip } }
1796   }
1797 }

```

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

13.25 Setting series, resume and resume* keys

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

```

series  We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1798 \cs_set_protected:Npn \__enumext_tmp:n #1
1799 {
1800   \keys_define:nn { enumext / #1 }
1801   {
1802     series .str_set:N = \l__enumext_series_str,
1803     series .value_required:n = true,
1804     resume .code:n = \__enumext_resume_series:n {##1},
1805     resume* .code:n = \__enumext_resume_starred:,
1806     resume* .value_forbidden:n = true,
1807   }
1808 }
1809 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

13.25.1 Internal functions for series key

```

\__enumext_filter_series:n
\__enumext_filter_series_key:n
\__enumext_filter_series_pair:nn

```

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

```

1810 \cs_new:Npn \__enumext_filter_series:n #1
1811 {
1812   \use:e
1813   {
1814     \keyval_parse:NNn
1815     \__enumext_filter_series_key:n
1816     \__enumext_filter_series_pair:nn {#1}
1817   }
1818 }

```

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

```

1819 \cs_new:Npn \__enumext_filter_series_key:n #1
1820 {
1821   \str_case:nnF {#1}
1822   {
1823     { resume } {} { resume* } {} { base-fix } {}
1824   }
1825   { , { \exp_not:n {#1} } }
1826 }

```

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

```

1827 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1828 {
1829   \str_case:nnF {#1}
1830   {
1831     { series } {} { resume } {} { start } {}
1832     { start* } {} { save-ans } {} { save-key } {}
1833   }
1834   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
1835 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

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

```

1836 \cs_new_protected:Npn \__enumext_parse_series:n #1
1837 {
1838   \str_if_empty:NTF \l__enumext_series_str
1839   {
1840     \bool_if:NF \l__enumext_resume_active_bool
1841     {
1842       \__enumext_resume_last:n {#1}
1843     }
1844   }
1845   {
1846     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1847     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1848     { \__enumext_filter_series:n {#1} }
1849     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1850     {
1851       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1852     }
1853   }
1854 }

```

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.

```

1855 \cs_new_protected:Npn \__enumext_resume_last:n #1
1856 {
1857   \bool_if:NT \l__enumext_standar_first_bool
1858   {
1859     \tl_gclear:N \g__enumext_standar_series_tl
1860     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1861   }
1862   \bool_if:NT \l__enumext_starred_first_bool
1863   {
1864     \tl_gclear:N \g__enumext_starred_series_tl
1865     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1866   }
1867 }

```

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

13.25.2 Internal function to save counter value

`__enumext_resume_save_counter:`

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

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

```

1868 \cs_new_protected:Nn \__enumext_resume_save_counter:
1869 {
1870     \bool_if:NT \g__enumext_standar_bool
1871     {
1872         \tl_if_empty:NF \l__enumext_series_str
1873         {
1874             \int_gset_eq:cN
1875             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1876         }
1877         \tl_if_empty:NTF \l__enumext_resume_name_tl
1878         {
1879             \str_if_empty:NT \l__enumext_series_str
1880             {
1881                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1882             }
1883         }
1884         {
1885             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1886             {
1887                 \int_gset_eq:cN
1888                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1889             }
1890         }
1891         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1892         {
1893             \int_gset_eq:cN
1894             { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1895         }
1896     }
1897     \bool_if:NT \g__enumext_starred_bool
1898     {
1899         \tl_if_empty:NF \l__enumext_series_str
1900         {
1901             \int_gset_eq:cN
1902             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1903         }
1904         \tl_if_empty:NTF \l__enumext_resume_name_tl
1905         {
1906             \str_if_empty:NT \l__enumext_series_str
1907             {
1908                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1909             }
1910         }
1911         {
1912             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1913             {
1914                 \int_gset_eq:cN
1915                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1916             }
1917         }
1918         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1919         {
1920             \int_gset_eq:cN
1921             { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1922         }
1923     }
1924 }

```

(End of definition for `__enumext_resume_save_counter:`.)

13.25.3 Internal functions for resume key

`__enumext_resume_series:n`

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

```

1925 \cs_new_protected:Npn \__enumext_resume_series:n #1
1926 {
1927   \tl_if_empty:nTF {#1}
1928   {
1929     \__enumext_resume_counter:n { }
1930   }
1931   {
1932     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1933     {
1934       \__enumext_resume_counter:n {#1}
1935       \bool_if:NT \g__enumext_standar_bool
1936       {
1937         \keys_set:nv { enumext / level-1 }
1938         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1939       }
1940       \bool_if:NT \g__enumext_starred_bool
1941       {
1942         \keys_set:nv { enumext / enumext* }
1943         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1944       }
1945     }
1946     {
1947       \bool_if:NT \g__enumext_standar_bool
1948       {
1949         \msg_error:nnn { enumext } { unknown-series } {#1}
1950       }
1951       \bool_if:NT \g__enumext_starred_bool
1952       {
1953         \msg_error:nnn { enumext } { unknown-series } {#1}
1954       }
1955     }
1956   }
1957 }

```

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

`__enumext_resume_counter:n`

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`.

```

1958 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1959 {
1960   \bool_set_true:N \l__enumext_resume_active_bool
1961   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1962   \tl_if_empty:NTF \l__enumext_resume_name_tl
1963   {
1964     \__enumext_resume_counter:
1965   }
1966   {
1967     \__enumext_resume_counter_series:
1968   }
1969   \__enumext_resume_counter_save_ans:
1970 }

```

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.

```

1971 \cs_new_protected:Nn \__enumext_resume_counter:

```

```

1972 {
1973   \bool_if:NT \g__enumext_standar_bool
1974   {
1975     \int_gincr:N \g__enumext_resume_int
1976     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1977   }
1978   \bool_if:NT \g__enumext_starred_bool
1979   {
1980     \int_gincr:N \g__enumext_resume_vii_int
1981     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1982   }
1983 }

```

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.

```

1984 \cs_new_protected:Nn \__enumext_resume_counter_series:
1985 {
1986   \bool_if:NT \g__enumext_standar_bool
1987   {
1988     \int_set:Nn \l__enumext_start_i_int
1989     {
1990       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1991     }
1992   }
1993   \bool_if:NT \g__enumext_starred_bool
1994   {
1995     \int_set:Nn \l__enumext_start_vii_int
1996     {
1997       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1998     }
1999   }
2000 }

```

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.

```

2001 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2002 {
2003   \bool_lazy_and:nnT
2004   { \bool_if_p:N \l__enumext_standar_first_bool }
2005   { \bool_if_p:N \l__enumext_store_active_bool }
2006   {
2007     \int_set:Nn \l__enumext_start_i_int
2008     {
2009       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2010     }
2011   }
2012   \bool_lazy_and:nnT
2013   { \bool_if_p:N \l__enumext_starred_first_bool }
2014   { \bool_if_p:N \l__enumext_store_active_bool }
2015   {
2016     \int_set:Nn \l__enumext_start_vii_int
2017     {
2018       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2019     }
2020   }
2021 }

```

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

13.25.4 Internal function for `resume*` key

`__enumext_resume_starred:`

The function `__enumext_resume_starred:` will handle the `resume*` key in the `enumext` and `enumext*` environments. This function will execute the filtered `⟨keys⟩` in the last one and will continue with the numbering according to the last execution of the environment `enumext` or `enumext*` in which the keys `resume={⟨series name⟩}` or `series={⟨series name⟩}` were not active.

```

2022 \cs_new_protected:Nn \__enumext_resume_starred:
2023 {
2024   \bool_if:NT \g__enumext_standar_bool
2025   {
2026     \tl_if_empty:NF \g__enumext_standar_series_tl

```

```

2027         {
2028             \__enumext_resume_counter:n { }
2029             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2030         }
2031     }
2032     \bool_if:NT \g__enumext_starred_bool
2033     {
2034         \tl_if_empty:NF \g__enumext_starred_series_tl
2035         {
2036             \__enumext_resume_counter:n { }
2037             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2038         }
2039     }
2040 }

```

(End of definition for __enumext_resume_starred:.)

13.26 Setting save-ans, check-ans and no-store keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

13.26.1 Setting save-ans key

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

```

2041 \cs_set_protected:Npn \__enumext_tmp:n #1
2042 {
2043     \keys_define:nn { enumext / #1 }
2044     {
2045         save-ans .code:n = \__enumext_storing_set:n {##1},
2046         save-ans .value_required:n = true,
2047     }
2048 }
2049 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.26.2 Internal functions for save-ans key

```

\__enumext_start_save_ans_msg:
\__enumext_stop_save_ans_msg:

```

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

```

2050 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2051 {
2052     \msg_term:nnVV { enumext } { save-ans-log }
2053     \g__enumext_envir_name_tl \l__enumext_store_name_tl
2054 }
2055 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2056 {
2057     \msg_term:nnVV { enumext } { save-ans-log-hook }
2058     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2059 }

```

(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.

```

2060 \cs_new_protected:Npn \__enumext_storing_set:n #1
2061 {
2062     \tl_set:Ne \l__enumext_store_name_tl {#1}
2063     \tl_if_empty:NTF \l__enumext_store_name_tl
2064     {
2065         \bool_lazy_or:nnT
2066         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2067         {
2068             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2069         }
2070     }

```



```

2071     {
2072         \bool_lazy_or:nnT
2073         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2074         {
2075             \__enumext_start_save_ans_msg:
2076             \__enumext_storing_exec:
2077         }
2078     }
2079 }

```

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

```

2080 \cs_new_protected:Nn \__enumext_storing_exec:
2081 {
2082     \bool_set_true:N \l__enumext_store_active_bool
2083     \bool_set_true:N \l__enumext_check_answers_bool
2084     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2085     \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

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

```

2086     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2087     {
2088         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2089         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2090     }
2091     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2092     {
2093         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2094         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2095     }
2096     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2097     {
2098         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2099         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2100     }
2101 }

```

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

13.26.3 The check answer mechanism

The internal mechanism for “checking answers” follows this logic:

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

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

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

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

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

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

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

13.26.4 Setting check-ans and no-store keys

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

```

2102 \cs_set_protected:Npn \__enumext_tmp:n #1
2103 {
2104   \keys_define:nn { enumext / #1 }
2105   {
2106     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2107     check-ans .initial:n = false,
2108     check-ans .value_required:n = true,
2109     no-store .code:n = {
2110       \bool_set_false:N \l__enumext_check_answers_bool
2111       \bool_set_false:N \l__enumext_check_ans_key_bool
2112     },
2113     no-store .value_forbidden:n = true,
2114   }
2115 }
2116 \clist_map_inline:nn
2117 {
2118   level-1, level-2, level-3, level-4, enumext*
2119 }
2120 { \__enumext_tmp:n {#1} }
```

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

13.26.5 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.

```

2121 \cs_new_protected:Nn \__enumext_check_ans_active:
2122 {
2123   \tl_if_empty:NF \l__enumext_store_name_tl
2124   {
2125     \bool_if:NT \l__enumext_check_answers_bool
2126     {
2127       \__enumext_check_ans_level:
2128     }
2129   }
2130 }
```

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”.

```

2131 \cs_new_protected:Nn \__enumext_check_ans_level:
2132 {
2133   \int_case:nn { \l__enumext_level_int }
2134   {
2135     { 1 }{
2136       \bool_lazy_all:nT
2137       {
2138         { \bool_if_p:N \g__enumext_starred_bool }
2139         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2140       }
2141       {
2142         \int_gdecr:N \g__enumext_item_number_int
2143         \bool_set_false:N \l__enumext_item_number_bool
2144       }
2145     }
2146     { 2 }{
2147       \int_gdecr:N \g__enumext_item_number_int
2148       \bool_set_false:N \l__enumext_item_number_bool
2149     }
2150     { 3 }{
2151       \int_gdecr:N \g__enumext_item_number_int
2152       \bool_set_false:N \l__enumext_item_number_bool
2153     }
2154   }
```

```

2154         { 4 }{
2155             \int_gdecr:N \g__enumext_item_number_int
2156             \bool_set_false:N \l__enumext_item_number_bool
2157         }
2158     }

```

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.

```

2159     \int_case:nn { \l__enumext_level_h_int }
2160     {
2161         { 1 }{
2162             \bool_lazy_all:nnT
2163             {
2164                 { \bool_if_p:N \g__enumext_standar_bool }
2165                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2166             }
2167             {
2168                 \int_gdecr:N \g__enumext_item_number_int
2169                 \bool_set_false:N \l__enumext_item_number_bool
2170             }
2171         }
2172     }
2173 }

```

(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.

```

2174 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2175 {
2176     \bool_lazy_and:nnT
2177     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2178     { \bool_if_p:N \g__enumext_standar_bool }
2179     {
2180         \bool_gset_true:N \g__enumext_check_ans_key_bool
2181     }
2182     \bool_lazy_and:nnT
2183     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2184     { \bool_if_p:N \g__enumext_starred_bool }
2185     {
2186         \bool_gset_true:N \g__enumext_check_ans_key_bool
2187     }
2188 }

```

(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:`.

```

2189 \cs_new_protected:Nn \__enumext_item_answer_diff:
2190 {
2191     \int_gset:Nn \g__enumext_item_answer_diff_int
2192     {
2193         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2194     }
2195 }

```

(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:`.

```

2196 \cs_new_protected:Nn \__enumext_check_ans_show:
2197 {
2198     \int_case:nn { \g__enumext_item_answer_diff_int }
2199     {
2200         { -1 }{ \__enumext_check_ans_msg_less: }
2201         { 0 }{ \__enumext_check_ans_msg_same_ok: }

```

```

2202         { 1 } { \__enumext_check_ans_msg_greater: }
2203     }
2204 }
2205 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2206 {
2207     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2208     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2209 }
2210 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2211 {
2212     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2213     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2214 }
2215 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2216 {
2217     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2218     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2219 }

```

(End of definition for __enumext_check_ans_show: and others.)

```

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

```

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

```

2220 \cs_new_protected:Nn \__enumext_check_ans_log:
2221 {
2222     \int_case:nn { \g__enumext_item_answer_diff_int }
2223     {
2224         { -1 } { \__enumext_check_ans_log_msg_less: }
2225         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2226         { 1 } { \__enumext_check_ans_log_msg_greater: }
2227     }
2228 }
2229 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2230 {
2231     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2232     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2233 }
2234 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2235 {
2236     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2237     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2238 }
2239 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2240 {
2241     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2242     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2243 }

```

(End of definition for __enumext_check_ans_log: and others.)

13.26.6 Check for \item* and \anspic* commands

```
\__enumext_check_starred_cmd:n
```

The function __enumext_check_starred_cmd:n performs an *extra check* for the `keyans`, `keyans*` and `keyanspic` environments. Unlike the *check* executed by `check-ans` key this one is not controlled by any key, it is intended to prevent the forgetting of `\item*` or `\anspic*` in these environments.

```

2244 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2245 {
2246     \int_compare:nNnT
2247     { \g__enumext_check_starred_cmd_int } = { 0 }
2248     {
2249         \msg_warning:nnnV
2250         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2251     }
2252     \int_compare:nNnT
2253     { \g__enumext_check_starred_cmd_int } > { 1 }
2254     {
2255         \msg_warning:nnnV
2256         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2257     }
2258     \int_gzero:N \g__enumext_check_starred_cmd_int

```

```

2259   \tl_clear:N \l__enumext_check_start_line_env_tl
2260 }

```

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

13.27 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*`.

```

2261 \cs_set_protected:Npn \__enumext_tmp:n #1
2262 {
2263   \keys_define:nn { enumext / #1 }
2264   {
2265     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2266     wrap-ans .initial:n =
2267       {
2268         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2269       },
2270     wrap-ans .value_required:n = true,
2271     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2272     wrap-opt .initial:n = [{##1}],
2273     wrap-opt .value_required:n = true,
2274     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2275     save-sep .initial:n = {, ~ },
2276     save-sep .value_required:n = true,
2277     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2278     mark-ans .initial:n = \textasteriskcentered,
2279     mark-ans .value_required:n = true,
2280     mark-pos .choice:,
2281     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2282     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2283     mark-pos / unknown .code:n =
2284       \msg_error:nneee { enumext } { unknown-choice }
2285       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2286     mark-pos .initial:n = right,
2287     mark-pos .value_required:n = true,
2288     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2289     show-ans .initial:n = false,
2290     show-ans .value_required:n = true,
2291     show-pos .bool_set:N = \l__enumext_show_position_bool,
2292     show-pos .initial:n = false,
2293     show-pos .value_required:n = true,
2294     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2295     mark-ref .initial:n = \textreferencemark,
2296     mark-ref .value_required:n = true,
2297     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2298     save-ref .initial:n = false,
2299     save-ref .value_required:n = true,
2300   }
2301 }
2302 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

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

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

```

2303 \cs_set_protected:Npn \__enumext_tmp:n #1
2304 {
2305   \keys_define:nn { enumext / #1 }
2306   {
2307     mark-pos .choice:,
2308     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2309     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2310     mark-pos .initial:n = right,
2311     mark-pos .value_required:n = true,
2312     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2313     show-ans .initial:n = false,
2314     show-ans .value_required:n = true,
2315     show-pos .bool_set:N = \l__enumext_show_position_bool,
2316     show-pos .initial:n = false,
2317     show-pos .value_required:n = true,
2318   }

```

```

2319 }
2320 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

13.27.1 Storing structure of the environments

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

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n` will be responsible for the “*storing keys*” filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\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*.

```

2321 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2322 {
2323   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2324   {
2325     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2326     \tl_set:ce
2327       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2328       { \__enumext_filter_save_key:n {#1} }
2329   }
2330 }
2331 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2332 {
2333   \bool_if:NF \l__enumext_store_save_key_vii_bool
2334   {
2335     \tl_clear:N \l__enumext_store_save_key_vii_tl
2336     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2337   }
2338 }

```

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

13.27.2 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the `save-ans` key when executing `\anskey` or `anskey*`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the *optional argument* of the “*storing structure*” in the *sequence*.

save-key

The values set by this key passed in the *optional argument* of the `enumext` and `enumext*` environments will override the values of the `\l__enumext_store_save_key_X_tl` variable set by the functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n`. Now define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2339 \cs_set_protected:Npn \__enumext_tmp:n #1
2340 {
2341   \keys_define:nn { enumext / enumext* }
2342   {
2343     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2344     save-key .value_required:n = true,
2345   }
2346   \keys_define:nn { enumext / #1 }
2347   {
2348     save-key .code:n = \__enumext_parse_save_key:n {##1},
2349     save-key .value_required:n = true,
2350   }
2351 }
2352 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for save-key.)

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for “*storing keys*” in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2353 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2354 {
2355   \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2356   \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2357   \tl_set:ce

```

```

2358     { l__enumext_store_save_key_ \__enumext_level: _tl }
2359     { \__enumext_filter_save_key:n {#1} }
2360   }
2361   \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2362   {
2363     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2364     \tl_clear:N \l__enumext_store_save_key_vii_tl
2365     \tl_set:Nx \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2366   }

```

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

13.27.3 Internal functions to store optional arguments

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

```

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

```

```

2367 \cs_new:Npn \__enumext_filter_save_key:n #1
2368 {
2369   \use:e
2370   {
2371     \keyval_parse:NNn
2372     \__enumext_filter_save_key_key:n
2373     \__enumext_filter_save_key_pair:nn {#1}
2374   }
2375 }

```

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

```

2376 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2377 {
2378   \str_case:nnF {#1}
2379   {
2380     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2381   }
2382   { , { \exp_not:n {#1} } }
2383 }

```

The function __enumext_filter_save_key_pair:nn will be responsible for “*filtering keys*” that are passed “*with value*” by excluding the `series`, `resume`, `save-ans`, `save-ref`, `check-ans`, `show-ans`, `save-pos`, `wrap-ans`, `mark-ans`, `wrap-opt`, `save-sep`, `mark-ref`, `mini-env`, `mini-sep`, `mini-right` and `mini-right*` keys.

```

2384 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2385 {
2386   \str_case:nnF {#1}
2387   {
2388     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2389     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2390     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2391     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2392     { mini-right* } {}
2393   }
2394   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2395 }

```

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

13.27.4 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2396 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2397 {
2398   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2399   {
2400     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2401   }
2402   { #1 }
2403 }
2404 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```


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

13.27.5 Function for storing content in sequence

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

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

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

```
2405 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2406 {
2407   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2408 }
2409 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.27.6 Functions for storing structure in the sequence

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

```
2410 \cs_new_protected:Nn \__enumext_store_level_open:
2411 {
2412   \bool_if:NT \l__enumext_check_answers_bool
2413   {
2414     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2415     {
2416       \__enumext_store_addto_seq:n
2417       {
2418         \item \begin{enumext}
2419       }
2420     }
2421     {
2422       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2423       {
2424         \item \begin{enumext} [
2425       }
2426       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2427       {
2428         ]
2429       }
2430       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2431     }
2432   }
2433 }
2434 \cs_new_protected:Nn \__enumext_store_level_close:
2435 {
2436   \bool_if:NT \l__enumext_check_answers_bool
2437   {
2438     \__enumext_store_addto_seq:n { \end{enumext} }
2439   }
2440 }
```

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

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

```
2441 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2442 {
2443   \bool_if:NT \l__enumext_check_answers_bool
2444   {
2445     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2446     {
2447       \__enumext_store_addto_seq:n
2448       {
2449         \item \begin{enumext*}
2450       }
2451     }
2452     {
2453       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2454       {
2455         \item \begin{enumext*}[
```

```

2456         }
2457         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2458         {
2459             ]
2460         }
2461         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2462     }
2463 }
2464 }
2465 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2466 {
2467     \bool_if:NT \l__enumext_check_answers_bool
2468     {
2469         \__enumext_store_addto_seq:n { \end{enumext*} }
2470     }
2471 }

```

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

13.27.7 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

```

2472 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2473 {
2474     \mode_leave_vertical:
2475     \skip_horizontal:n { -\dim_use:N #2 }
2476     \makebox[0pt][ r ]
2477     {
2478         \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2479         {
2480             \tl_use:N \l__enumext_mark_answer_sym_tl
2481         }
2482     }
2483     \skip_horizontal:n { \dim_use:N #2 }
2484 }
2485 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

13.28 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.

```

2486 \cs_new_protected:Nn \__enumext_store_internal_ref:
2487 {
2488     \cs_set_protected:Npn \__enumext_tmp:n ##1
2489     {
2490         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2491         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2492         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2493         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2494     }
2495     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2496     \cs_set:Npn \__enumext_tmp:n ##1
2497     { . \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.

```

2498     \bool_lazy_all:nT
2499     {
2500         { \bool_if_p:N \g__enumext_starred_bool }
2501         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2502     }
2503     {
2504         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl

```

```

2505         { \tl_use:N \l__enumext_label_copy_vii_tl }
2506     }
2507     \bool_lazy_all:nT
2508     {
2509         { \bool_not_p:n { \g__enumext_standar_bool } }
2510         { \bool_if_p:N \l__enumext_standar_bool }
2511         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2512     }
2513     {
2514         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2515         {
2516             \tl_use:N \l__enumext_label_copy_vii_tl
2517             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2518         }
2519     }

```

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.

```

2520     \bool_lazy_all:nT
2521     {
2522         { \bool_if_p:N \g__enumext_standar_bool }
2523         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2524         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2525     }
2526     {
2527         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2528         {
2529             \tl_use:N \l__enumext_label_copy_i_tl
2530             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2531         }
2532     }
2533     \cs_set:Npn \__enumext_tmp:n ##1
2534     { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2535     \bool_lazy_all:nT
2536     {
2537         { \bool_if_p:N \g__enumext_standar_bool }
2538         { \bool_if_p:N \l__enumext_starred_bool }
2539         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2540     }
2541     {
2542         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2543         {
2544             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2545             \tl_use:N \l__enumext_label_copy_vii_tl
2546         }
2547     }

```

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

```

2548     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2549     {
2550         \l__enumext_store_name_tl \c_colon_str
2551         \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2552     }

```

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

```

2553     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2554     {
2555         \__enumext_newlabel:nn
2556         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2557         { \l__enumext_newlabel_arg_two_tl }
2558     }
2559     \l__enumext_write_aux_file_tl
2560 }

```

(End of definition for `__enumext_store_internal_ref:`)

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

`__enumext_store_anskey_code:n`

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

```

2561 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2562 {
2563   \int_gincr:N \g__enumext_item_anskey_int
2564   \__enumext_store_addto_prop:n {#1}
2565   \bool_if:NT \l__enumext_store_ref_key_bool
2566   {
2567     \__enumext_store_internal_ref:
2568   }
2569   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2570 \tl_clear:N \l__enumext_store_anskey_arg_tl
2571 \bool_lazy_and:nnT
2572 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2573 { \bool_not_p:n { \l__enumext_starred_bool } }
2574 {
2575   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2576 }
2577 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2578 \bool_lazy_and:nnT
2579 { \bool_not_p:n { \l__enumext_starred_bool } }
2580 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2581 {
2582   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2583   {
2584     ( \exp_not:V \l__enumext_store_item_join_int )
2585   }
2586 }

```

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

```

2587 \bool_if:NTF \l__enumext_store_item_star_bool
2588 {
2589   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2590   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2591   {
2592     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2593     {
2594       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2595     }
2596   }
2597   \dim_compare:nT
2598   {
2599     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2600   }
2601   {
2602     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2603     {
2604       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2605     }
2606   }
2607   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2608 }
2609 {
2610   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2611 }

```

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

```

2612 \bool_lazy_and:nnT
2613 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2614 { \bool_if_p:N \l__enumext_hyperref_bool }
2615 {
2616   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2617   {
2618     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }

```

```

2619             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2620         }
2621     }
2622     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2623 }

```

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

`__enumext_anskey_show_wrap_arg:n`

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

```

2624 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2625 {
2626     \par
2627     \bool_if:NTF \l__enumext_starred_bool
2628     {
2629         \__enumext_print_keyans_box:NN
2630         \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2631     }
2632     {
2633         \__enumext_print_keyans_box:cc
2634         { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2635         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2636     }
2637     \__enumext_anskey_wrapper:n { #1 }
2638 }

```

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

`__enumext_anskey_show_wrap_left:n`

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

```

2639 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2640 {
2641     \bool_if:NT \l__enumext_show_answer_bool
2642     {
2643         \__enumext_anskey_show_wrap_arg:n { #1 }
2644     }
2645     \bool_if:NT \l__enumext_show_position_bool
2646     {
2647         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2648         {
2649             \group_begin:
2650             \exp_not:N \normalfont
2651             \exp_not:N \footnotesize [ \int_eval:n
2652             {
2653                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2654             }
2655             ]
2656             \group_end:
2657         }
2658         \__enumext_anskey_show_wrap_arg:n { #1 }
2659     }
2660 }

```

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

13.30 The command `\anskey`

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

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

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

```

2661 \keys_define:nn { enumext / anskey }
2662 {
2663     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2664     break-col .default:n = true,
2665     break-col .value_forbidden:n = true,
2666     item-join .int_set:N = \l__enumext_store_item_join_int,

```

```

2667     item-join .value_required:n = true,
2668     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2669     item-star .default:n = true,
2670     item-star .value_forbidden:n = true,
2671     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2672     item-sym* .value_required:n = true,
2673     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2674     item-pos* .value_required:n = true,
2675     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2676 }

```

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

```

2677 \cs_new_protected:Npn \l__enumext_anskey_unknown:n #1
2678 {
2679     \exp_args:NV \l__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2680 }
2681 \cs_new_protected:Npn \l__enumext_anskey_unknown:nn #1 #2
2682 {
2683     \tl_if_blank:nTF {#2}
2684     {
2685         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2686     }
2687     {
2688         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2689     }
2690 }

```

(End of definition for `\l__enumext_anskey_unknown:n` and `\l__enumext_anskey_unknown:nn`.)

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

`\anskey` We will first call the function `\l__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and execute the function `\l__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[<key = val>]` and call the function `\l__enumext_store_anskey_code:n`.

```

2691 \NewDocumentCommand \anskey { o +m }
2692 {
2693     \l__enumext_anskey_safe_outer:
2694     \group_begin:
2695         \bool_if:NT \l__enumext_check_answers_bool
2696         {
2697             \tl_if_novalue:nF {#1}
2698             {
2699                 \keys_set:nn { enumext / anskey } {#1}
2700             }
2701             \tl_if_blank:nTF {#2}
2702             {
2703                 \msg_error:nn { enumext } { anskey-empty-arg }
2704             }
2705             {
2706                 \l__enumext_anskey_safe_inner:
2707                 \l__enumext_store_anskey_code:n {#2}
2708             }
2709         }
2710     \group_end:
2711 }

```

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

13.30.1 Internal functions for the command

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

`\l__enumext_anskey_safe_inner:`

```

2712 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2713 {
2714     \bool_if:NF \l__enumext_store_active_bool
2715     {
2716         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2717     }

```

```

2718 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
2719 {
2720   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2721 }
2722 \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2723 {
2724   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2725 }
2726 \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2727 {
2728   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2729 }
2730 }

```

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.

```

2731 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2732 {
2733   \int_incr:N \__enumext_anskey_level_int
2734   \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2735   {
2736     \msg_error:nn { enumext } { anskey-nested }
2737   }
2738   \bool_if:NF \__enumext_item_number_bool
2739   {
2740     \msg_error:nn { enumext } { anskey-unnumber-item }
2741   }
2742   \mode_if_math:T
2743   {
2744     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2745   }
2746 }

```

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

13.31 The environment `anskey*`

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

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

`__enumext_undefine_anskey_env:`

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

```

2747 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2748 {
2749   \cs_undefine:c { anskey* }
2750   \cs_undefine:c { endanskey* }
2751   \cs_undefine:c { __scontents_anskey*_env_begin: }
2752   \cs_undefine:c { __scontents_anskey*_env_end: }
2753 }

```

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

```

2754 \__enumext_before_env:nn { enumext }
2755 {
2756   \bool_lazy_and:nnT
2757   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2758   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2759   {
2760     \cs_if_free:cf { __scontents_anskey*_env_begin: }
2761     {

```



```

2762         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2763     }
2764 }
2765 }
2766 \__enumext_before_env:nn { enumext* }
2767 {
2768     \bool_lazy_and:nnT
2769     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2770     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2771     {
2772         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2773         {
2774             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2775         }
2776     }
2777 }

```

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.

```

2778 \__enumext_before_env:nn { anskey* }
2779 {
2780     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2781     {
2782         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2783     }
2784     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2785     {
2786         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2787     }
2788     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2789     {
2790         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2791     }
2792     \bool_if:NF \__enumext_item_number_bool
2793     {
2794         \msg_error:nn { enumext } { anskey-unnumber-item }
2795     }
2796     \mode_if_math:T
2797     {
2798         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2799     }
2800 }

```

(End of definition for `__enumext_undefine_anskey_env:`.)

`anskey*`

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

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec: (§13.26.1)` 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`.

```

2801 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2802 {
2803     \bool_if:NT \__enumext_anskey_env_bool
2804     {
2805         \newenvsc{anskey*}[store-env=#1,print-env=false]
2806         \__enumext_anskey_env_exec:
2807     }
2808 }
2809 \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`.

```

2810 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2811 {
2812     \keys_define:nn { scontents / scontents }
2813     {
2814         break-col .bool_gset:N = \__enumext_store_columns_break_bool,
2815         break-col .default:n = true,

```

```

2816         break-col .value_forbidden:n = true,
2817         item-join .int_gset:N = \g__enumext_store_item_join_int,
2818         item-join .value_required:n = true,
2819         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2820         item-star .default:n = true,
2821         item-star .value_forbidden:n = true,
2822         item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2823         item-sym* .value_required:n = true,
2824         item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2825         item-pos* .value_required:n = true,
2826         print-env .undefine:,
2827         store-env .undefine:,
2828         write-out .undefine:,
2829         unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2830     }
2831 }

```

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`.

```

2832 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2833 {
2834     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2835 }
2836 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2837 {
2838     \tl_if_blank:nTF {#2}
2839     {
2840         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2841     }
2842     {
2843         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2844     }
2845 }

```

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`.

```

2846 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2847 {
2848     \keys_define:nn { scontents / scontents }
2849     {
2850         break-col .undefine:,
2851         item-join .undefine:,
2852         item-star .undefine:,
2853         item-sym* .undefine:,
2854         item-pos* .undefine:,
2855         write-out .code:n = {
2856             \bool_set_false:N \l__scontents_storing_bool
2857             \bool_set_true:N \l__scontents_writing_bool
2858             \tl_set:Nn \l__scontents_fname_out_tl {##1}
2859         },
2860         write-out .value_required:n = true,
2861         print-env .meta:nn = { scontents } { print-env = ##1 },
2862         print-env .default:n = true,
2863         store-env .meta:nn = { scontents } { store-env = ##1 },
2864         unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2865     }
2866 }

```

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*.

```

2867 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2868 {
2869     \group_begin:
2870     \int_set:Nn \tex_newlinechar:D { `^^J }
2871     \__scontents_rescan_tokens:x
2872     {
2873         \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2874         #1
2875     }
2876 }

```

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

`__enumext_anskey_env_exec:`

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

```
2877 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2878 {
2879   \__enumext_before_env:nn { anskey* }
2880   {
2881     \__enumext_anskey_env_define_keys:
2882   }
```

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.

```
2883   \hook_if_empty:nF {env/anskey*/after}
2884   {
2885     \hook_gremove_code:nn {env/anskey*/after} { * }
2886   }
2887   \__enumext_after_env:nn { anskey* }
2888   {
2889     \__enumext_anskey_env_save_keys:
2890     \tl_clear:N \l__enumext_store_anskey_env_tl
2891     \tl_clear:N \l__enumext_store_anskey_opt_tl
2892     \bool_if:NT \l__enumext_check_answers_bool
2893     {
2894       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2895       {
2896         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2897       }
2898       \regex_match:nVTF
2899       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2900       \l__enumext_store_anskey_env_tl
2901       {
2902         \msg_error:nn { enumext } { anskey-empty-arg }
2903       }
2904       {
2905         \__enumext_anskey_env_store:
2906       }
2907     }
2908     \__enumext_anskey_env_clean_vars:
2909     \__enumext_anskey_env_reset_keys:
2910   }
2911 }
```

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`.

```
2912 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2913 {
2914   \bool_lazy_and:nnT
2915   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2916   { \bool_not_p:n { \l__enumext_starred_bool } }
2917   {
2918     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2919   }
```

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

```
2920   \bool_lazy_and:nnT
2921   { \bool_not_p:n { \l__enumext_starred_bool } }
2922   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2923   {
2924     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2925     {
2926       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2927     }
2928   }
```

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

```

2929   \bool_if:NT \g__enumext_store_item_star_bool
2930   {
2931     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2932     {
2933       ,item-star,
2934     }
2935     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2936     {
2937       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2938       {
2939         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2940       }
2941     }
2942     \dim_compare:nT
2943     {
2944       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2945     }
2946     {
2947       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2948       {
2949         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2950       }
2951     }
2952   }
2953 }

```

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`.

```

2954 \cs_new_protected:Nn \__enumext_anskey_env_store:
2955 {
2956   \group_begin:
2957   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2958   {
2959     \exp_args:Ne
2960     \__enumext_store_anskey_code:n
2961     {
2962       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2963     }
2964   }
2965   {
2966     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2967     \exp_args:Ne
2968     \__enumext_store_anskey_code:n
2969     {
2970       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2971     }
2972   }
2973   \group_end:
2974 }

```

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

```

2975 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2976 {
2977   \bool_gset_false:N \g__enumext_store_columns_break_bool
2978   \int_gzero:N       \g__enumext_store_item_join_int
2979   \bool_gset_false:N \g__enumext_store_item_star_bool
2980   \tl_gclear:N       \g__enumext_store_item_symbol_tl
2981   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2982 }

```

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

13.32 Executing `anskey*`, `check-ans` and `write .log`

`__enumext_execute_after_env:`

The `__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `__enumext_check_ans_show:` and show the result in the terminal,

otherwise it will execute the function `__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment `anskey*` (§13.31) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

2983 \cs_new_protected:Nn \__enumext_execute_after_env:
2984 {
2985   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
2986   {
2987     \tl_if_empty:NF \g__enumext_store_name_tl
2988     {
2989       \__enumext_stop_save_ans_msg:
2990       \__enumext_item_answer_diff:
2991       \__enumext_log_global_vars:
2992       \__enumext_log_answer_vars:
2993       \bool_if:NTF \g__enumext_check_ans_key_bool
2994       {
2995         \__enumext_check_ans_show:
2996       }
2997       { \__enumext_check_ans_log: }
2998       \__enumext_undefine_anskey_env:
2999     }
3000     \__enumext_reset_global_vars:
3001   }
3002 }

```

(End of definition for `__enumext_execute_after_env:`.)

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

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

13.33.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the the current `<label>` for `\item*` in `keyans` environment and the current `<label>` for `\anspic*` in `keyanspic` environment followed by the `<contents>` of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be stored to the *prop list* defined by the `save-ans` key using the function `__enumext_store_addto_prop:V`.

```

3003 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3004 {
3005   \tl_clear:N \l__enumext_store_current_label_tl
3006   \int_compare:nNtTF { \l__enumext_keyans_pic_level_int } = { 1 }
3007   {
3008     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3009   }
3010   {
3011     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3012   }

```

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

```

3013   \tl_if_novalue:nF { #1 }
3014   {
3015     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3016     {
3017       \tl_put_right:Ne \l__enumext_store_current_label_tl
3018       {
3019         \l__enumext_store_keyans_item_opt_sep_tl
3020       }
3021     }
3022     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3023   }
3024   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3025 }

```

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

13.33.2 The `save-ref` key for `keyans`, `keyans*` and `keyanspic`

The “*internal label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has *slight differences* with the one implemented for `\anskey` basically because in this environments the interest is in the current `<label>` for `\item*` and `\anspic*` with the `<contents>` of the *optional argument*. The mechanism defined here will allow to execute `\ref{<store name : position>}` and will return `1 . (A)`.

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

```

3026 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3027 {
3028   \bool_if:NT \l__enumext_store_ref_key_bool
3029   {
3030     \cs_set_protected:Npn \__enumext_tmp:n ##1
3031     {
3032       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
3033       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
3034       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
3035       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
3036     }
3037     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3038     \__enumext_keyans_store_ref_aux_i:
3039   }
3040 }

```

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

```

3041 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3042 {
3043   \bool_if:NT \g__enumext_starred_bool
3044   {
3045     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3046   }
3047   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
3048   {
3049     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3050     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3051   }
3052   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3053   {
3054     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3055     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3056   }
3057   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3058   {
3059     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3060     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3061   }
3062   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3063   {
3064     \l__enumext_store_name_tl \c_colon_str
3065     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3066   }
3067   \__enumext_keyans_store_ref_aux_ii:
3068 }

```

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.

```

3069 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3070 {
3071   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3072   {
3073     \__enumext_newlabel:nn
3074     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3075     { \l__enumext_newlabel_arg_two_tl }
3076   }
3077   \l__enumext_write_aux_file_tl
3078 }

```

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

13.33.3 Storing content in sequence

`__enumext_keyans_addto_seq:n` The function `__enumext_keyans_addto_seq:n` will pass the contents of the current *(label)* `\l__enumext_label_v_tl` for the `keyans` environment and the `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *(contents)* of the *optional argument* of both

commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3079 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3080 {
3081   \tl_clear:N \l__enumext_store_current_label_tl
3082   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3083   {
3084     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3085   }
3086   {
3087     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3088   }
3089   \tl_if_novalue:nF { #1 }
3090   {
3091     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3092     {
3093       \tl_put_right:Ne \l__enumext_store_current_label_tl
3094       {
3095         \l__enumext_store_keyans_item_opt_sep_tl
3096       }
3097     }
3098     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3099   }
3100   \__enumext_keyans_addto_seq_link:
3101 }

```

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.

```

3102 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3103 {
3104   \bool_lazy_and:nnT
3105   { \bool_if_p:N \l__enumext_store_ref_key_bool }
3106   { \bool_if_p:N \l__enumext_hyperref_bool }
3107   {
3108     \tl_put_right:Ne \l__enumext_store_current_label_tl
3109     {
3110       \hfill \exp_not:N \hyperlink
3111       {
3112         \exp_not:V \l__enumext_newlabel_arg_one_tl
3113       }
3114       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3115     }
3116   }
3117   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3118   \bool_if:NT \l__enumext_check_answers_bool
3119   {
3120     \int_gincr:N \g__enumext_item_anskey_int
3121   }
3122 }

```

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

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

The 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.

```

3123 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3124 {
3125   \tl_if_novalue:nF { #1 }
3126   {
3127     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3128   }
3129   \bool_if:NT \l__enumext_show_answer_bool
3130   {
3131     \__enumext_keyans_show_ans:

```



```

3132     }
3133     \bool_if:NT \l__enumext_show_position_bool
3134     {
3135         \__enumext_keyans_show_pos:
3136     }
3137 }
3138 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3139 {
3140     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3141     {
3142         \bool_lazy_or:nnT
3143         { \bool_if_p:N \l__enumext_show_answer_bool }
3144         { \bool_if_p:N \l__enumext_show_position_bool }
3145         {
3146             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3147         }
3148     }
3149 }
3150 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3151 {
3152     \bool_if:NT \l__enumext_starred_bool
3153     {
3154         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3155         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3156     }
3157     \tl_put_left:Nn \l__enumext_label_v_tl
3158     {
3159         \__enumext_print_keyans_box:NN
3160         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3161     }
3162 }
3163 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3164 {
3165     \bool_if:NT \l__enumext_starred_bool
3166     {
3167         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3168         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3169     }
3170     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3171     {
3172         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3173         {
3174             \group_begin:
3175             \exp_not:N \normalfont
3176             \exp_not:N \footnotesize [ \int_eval:n
3177                 {
3178                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3179                 }
3180             ]
3181             \group_end:
3182         }
3183     }
3184     {
3185         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3186         {
3187             \group_begin:
3188             \exp_not:N \normalfont
3189             \exp_not:N \footnotesize [ \int_eval:n
3190                 {
3191                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3192                 }
3193             ]
3194             \group_end:
3195         }
3196     }
3197     \tl_put_left:Nn \l__enumext_label_v_tl
3198     {
3199         \__enumext_print_keyans_box:NN
3200         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3201     }
3202 }

```

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

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

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

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

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

`__enumext_default_item:n`

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

```

3203 \cs_new_protected:Npn \__enumext_default_item:n #1
3204 {
3205   \tl_if_novalue:nTF {#1}
3206   {
3207     \bool_if:NT \l__enumext_check_answers_bool
3208     {
3209       \int_gincr:N \g__enumext_item_number_int
3210       \bool_set_true:N \l__enumext_item_number_bool
3211     }
3212     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3213     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3214   }
3215   {
3216     \bool_set_eq:cc
3217     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3218     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3219     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3220   }
3221 }

```

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

`__enumext_starred_item:nn`

`__enumext_item_star_exec:`

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

`#1: \l__enumext_item_symbol_X_tl`

`#2: \l__enumext_item_symbol_sep_X_dim`

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

```

3222 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3223 {
3224   \tl_if_novalue:nTF {#1}
3225   {
3226     \tl_gset_eq:Nc
3227     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3228   }
3229   {
3230     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3231   }
3232   \tl_if_novalue:nTF {#2}
3233   {
3234     \dim_set_eq:cc
3235     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3236     { \l__enumext_labelsep_ \__enumext_level: _dim }
3237   }

```

```

3238     {
3239         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3240     }
3241     \bool_if:NT \__enumext_check_answers_bool
3242     {
3243         \int_gincr:N \g__enumext_item_number_int
3244         \bool_set_true:N \__enumext_item_number_bool
3245     }
3246     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3247     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3248 }

```

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

```

3249 \cs_new_protected:Nn \__enumext_item_star_exec:
3250 {
3251     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3252     {
3253         \mode_leave_vertical:
3254         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3255         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3256         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3257     }
3258 }

```

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

`__enumext_redefine_item:`

The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function is passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```

3259 \cs_new_protected:Nn \__enumext_redefine_item:
3260 {
3261     \RenewDocumentCommand \item { s o o }
3262     {
3263         \bool_if:nTF {##1}
3264         {
3265             \__enumext_starred_item:nn {##2} {##3}
3266         }
3267         { \__enumext_default_item:n {##2} }
3268     }
3269 }

```

(End of definition for `__enumext_redefine_item:`.)

`__enumext_make_label:`
`__enumext_make_label_std:`
`__enumext_make_label_box:`

The function `__enumext_make_label:` redefine `\make_label` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function is passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```

3270 \cs_new_protected:Nn \__enumext_make_label:
3271 {
3272     \IfDocumentMetadataTF
3273     {
3274         \__enumext_make_label_box:
3275     }
3276     {
3277         \bool_if:NTF \l__enumext_mode_box_bool
3278         {
3279             \__enumext_make_label_box:
3280         }
3281         {
3282             \__enumext_make_label_std:
3283         }
3284     }
3285 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3286 \cs_new_protected:Nn \__enumext_make_label_std:
3287 {
3288     \RenewDocumentCommand \make_label { m }
3289     {
3290         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3291         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }

```

```

3292     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3293     {
3294         \__enumext_item_star_exec:
3295         \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3296     }
3297     { ##1 }
3298     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3299     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3300 }
3301 }

```

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

```

3302 \cs_new_protected:Nn \__enumext_make_label_box:
3303 {
3304     \RenewDocumentCommand \make_label { m }
3305     {
3306         \makebox
3307         [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3308         [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3309         {
3310             \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3311             \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3312             {
3313                 \__enumext_item_star_exec:
3314                 \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3315             }
3316             { ##1 }
3317             \tl_gclear:N \g__enumext_item_symbol_aux_tl
3318         }
3319     }
3320 }

```

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

13.35 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` for the `enumext` and `enumext*` environments it is best to define a couple of keys that allow us to control and set by default the `<symbol>` and its `<offset>`.

`item-sym*` Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

3321 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3322 {
3323     \keys_define:nn { enumext / #1 }
3324     {
3325         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3326         item-sym* .value_required:n = true,
3327         item-sym* .initial:n = {\textasteriskcentered},
3328         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3329         item-pos* .value_required:n = true,
3330     }
3331 }
3332 \clist_map_inline:nn
3333 {
3334     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3335 }
3336 { \__enumext_tmp:nn #1 }

```

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

13.36 Handling unknown keys

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

13.36.1 Handling unknown keys for `keyans`, `keyans*` and `keyanspic`

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

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:n
3337 \cs_set_protected:Npn \__enumext_tmp:n #1
3338 {
3339     \keys_define:nn { enumext / #1 }
3340     {
3341         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3342     }

```

```

3343     }
3344     \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3345 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3346 {
3347     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3348 }
3349 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3350 {
3351     \tl_if_blank:nTF {#2}
3352     {
3353         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3354     }
3355     {
3356         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3357     }
3358 }

```

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

13.36.2 Handling unknown keys for `enumext*`

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

```

unknown
\__enumext_starred_unknown_keys:n 3359 \keys_define:nn { enumext / enumext* }
\__enumext_starred_unknown_keys:nn 3360 {
3361     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3362 }

```

Internal functions for handling `unknown` key.

```

3363 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3364 {
3365     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3366 }
3367 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3368 {
3369     \tl_if_blank:nTF {#2}
3370     {
3371         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3372     }
3373     {
3374         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3375     }
3376 }

```

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

13.36.3 Handling unknown keys for `enumext`

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

```

unknown
\__enumext_standar_unknown_keys:n 3377 \cs_set_protected:Npn \__enumext_tmp:n #1
\__enumext_standar_unknown_keys:nn 3378 {
3379     \keys_define:nn { enumext / #1 }
3380     {
3381         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3382     }
3383 }
3384 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3385 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3386 {
3387     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3388 }
3389 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3390 {
3391     \tl_if_blank:nTF {#2}
3392     {
3393         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3394     }
3395     {
3396         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3397     }
3398 }

```

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

13.37 Redefining \item and \makeLabel in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current `⟨label⟩` next to the `⟨content⟩` if it is present in the *sequence* and *prop list* defined by `save-ans` key.

`__enumext_keyans_default_item:n`

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

```

3399 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3400 {
3401   \tl_if_novalue:nTF { #1 }
3402   {
3403     \bool_set_true:N \__enumext_wrap_label_v_bool
3404     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3405   }
3406   {
3407     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3408     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3409   }
3410 }

```

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

`__enumext_keyans_starred_item:n`

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

```

3411 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3412 {
3413   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3414   \__enumext_keyans_show_left:n { #1 }
3415   \bool_set_true:N \__enumext_wrap_label_v_bool
3416   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3417   \__enumext_keyans_show_item_opt:

```

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

```

3418   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3419   \__enumext_keyans_addto_prop:n { #1 }
3420   \__enumext_keyans_store_ref:
3421   \__enumext_keyans_addto_seq:n { #1 }
3422   \int_gincr:N \g__enumext_check_starred_cmd_int
3423 }

```

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

`\item*`

`__enumext_keyans_redefine_item:`

The function `__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we need to use `\peek_remove_spaces:n` to prevent an unwanted space when using `\item*` in conjunction with the `itemindent` key. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38.2).

```

3424 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3425 {
3426   \RenewDocumentCommand \item { s o }
3427   {
3428     \bool_if:nTF {##1}
3429     {
3430       \peek_remove_spaces:n
3431       {
3432         \__enumext_keyans_starred_item:n {##2}
3433       }
3434     }
3435     {
3436       \__enumext_keyans_default_item:n {##2}
3437     }
3438   }
3439 }

```

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

```

\__enumext_keyans_make_label:
\__enumext_keyans_make_label_std:
\__enumext_keyans_make_label_box:

```

The function `__enumext_keyans_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `keyans` environment. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38.2).

```

3440 \cs_new_protected:Nn \__enumext_keyans_make_label:
3441 {
3442   \IfDocumentMetadataTF
3443   {
3444     \__enumext_keyans_make_label_box:
3445   }
3446   {
3447     \bool_if:NTF \l__enumext_mode_box_bool
3448     {
3449       \__enumext_keyans_make_label_box:
3450     }
3451     {
3452       \__enumext_keyans_make_label_std:
3453     }
3454   }
3455 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3456 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3457 {
3458   \RenewDocumentCommand \makelabel { m }
3459   {
3460     \tl_use:N \l__enumext_label_fill_left_v_tl
3461     \tl_use:N \l__enumext_label_font_style_v_tl
3462     \bool_if:NTF \l__enumext_wrap_label_v_bool
3463     {
3464       \__enumext_wrapper_label_v:n { ##1 }
3465     }
3466     { ##1 }
3467     \tl_use:N \l__enumext_label_fill_right_v_tl
3468   }
3469 }

```

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

```

3470 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3471 {
3472   \RenewDocumentCommand \makelabel { m }
3473   {
3474     \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3475     {
3476       \tl_use:N \l__enumext_label_font_style_v_tl
3477       \bool_if:NTF \l__enumext_wrap_label_v_bool
3478       {
3479         \__enumext_wrapper_label_v:n { ##1 }
3480       }
3481       { ##1 }
3482     }
3483   }
3484 }

```

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

13.38 Second argument of the lists

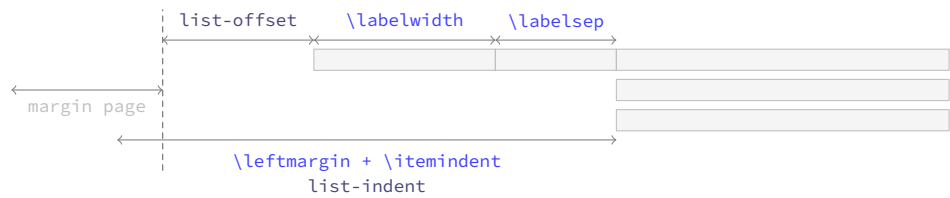
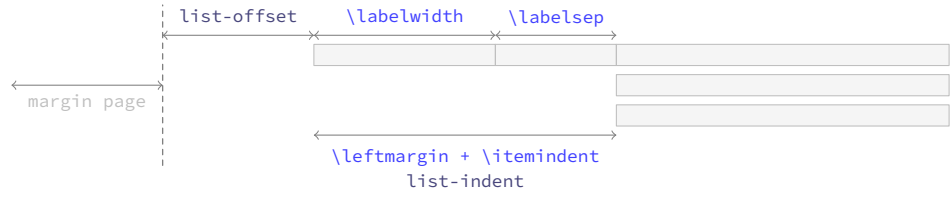
At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:n` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and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

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

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

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The key relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the *label box* is at `\leftmargin+\itemindent` minus `\labelwidth+\labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

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

Figure 9: Representation of standard horizontal lengths in `list` environment.Figure 10: Representation of horizontal lengths concept in list in `enumext`.

```

\__enumext_calc_hspace:NNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

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

```

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

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` and `keyans` environments (§13.38.2).

```

3485 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3486 {
3487   \dim_compare:nNt { #1 } < { \c_zero_dim }
3488   {
3489     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3490     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3491   }
3492   \dim_compare:nNt { #2 } < { \c_zero_dim }
3493   {
3494     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3495     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3496   }

```

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

```

3497   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

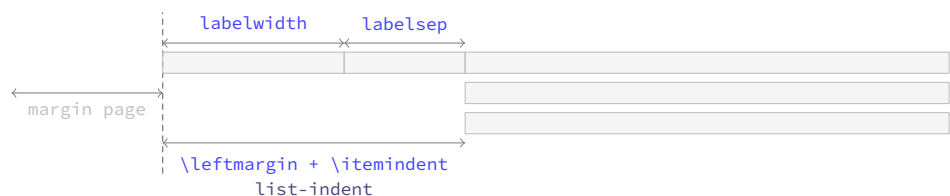
```

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

```

3498   \dim_compare:nNtTF { #4 } < { \c_zero_dim }
3499   {
3500     \dim_set:Nn #6 { #1 + #2 - #4 }
3501     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3502   }
3503   {
3504     \dim_compare:nNt { #4 } = { #1 + #2 }
3505     { \dim_set:Nn #6 { \c_zero_dim } }
3506     \dim_compare:nNt { #4 } < { #1 + #2 }
3507     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3508     \dim_compare:nNt { #4 } > { #1 + #2 }
3509     {

```

Figure 11: Default horizontal lengths in `enumext`.

```

3510         \dim_set:Nn #6 { -#1 - #2 + #4 }
3511         \dim_set:Nn #6 { #6*-1 }
3512     }
3513     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3514 }
3515 }
3516 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(End of definition for __enumext_calc_hspace:NNNNNNN.)

13.38.2 Setting second argument of the lists

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

3517 \cs_set_protected:Npn \__enumext_tmp:n #1
3518 {
3519     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3520     {
3521         \__enumext_calc_hspace:ccccc
3522         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3523         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3524         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3525         { l__enumext_leftmargin_tmp_#1_bool }
3526     \clist_map_inline:nn
3527     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3528     { \dim_set_eq:cc {###1} { l__enumext_###1_#1_dim } }
3529     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3530     { \skip_set_eq:cc {###1} { l__enumext_###1_#1_skip } }
3531     \usecounter { enumX#1 }
3532     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3533     \str_if_eq:nnTF {#1} { v }
3534     {
3535         \__enumext_keyans_redefine_item:
3536         \__enumext_keyans_make_label:
3537         \__enumext_keyans_ref:
3538         \__enumext_keyans_fake_item_indent:
3539         \bool_if:cT { l__enumext_show_length_#1_bool }
3540         {
3541             \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3542         }
3543     }
3544     {
3545         \__enumext_redefine_item:
3546         \__enumext_make_label:
3547         \__enumext_standar_ref:
3548         \__enumext_fake_item_indent:
3549         \bool_if:cT { l__enumext_show_length_#1_bool }
3550         {
3551             \msg_term:nnne { enumext } { list-lengths } {#1}
3552             { \int_use:N \l__enumext_level_int }
3553         }
3554     }
3555 }
3556 }
3557 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

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

```

3558 \cs_set_protected:Npn \__enumext_tmp:n #1
3559 {
3560     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3561     {
3562         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3563         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3564         \__enumext_calc_hspace:ccccc
3565         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3566         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3567         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }

```

```

3568         { \__enumext_leftmargin_tmp_#1_bool }
3569 \clist_map_inline:nn
3570   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3571   { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3572 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3573   { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3574 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3575 \skip_zero:N \partopsep
3576 \usecounter { enumX#1 }
3577 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3578 \__enumext_starred_ref:
3579 \str_if_eq:nnTF {#1} { vii }
3580   {
3581     \__enumext_fake_item_indent_vii:
3582     \bool_if:cT { \__enumext_show_length_vii_bool }
3583       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3584   }
3585   {
3586     \__enumext_fake_item_indent_viii:
3587     \bool_if:cT { \__enumext_show_length_#1_bool }
3588       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3589   }
3590 }
3591 }
3592 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

13.39 The environment enumext

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to “true” if we are NOT nested within `enumext*`, then call the function __enumext_internal_mini_page: to create the environment `__enumext_mini_page`, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to “true” and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to “true” only if the environment is NOT nested and we are at the “first level”.

```

3593 \cs_new_protected:Nn \__enumext_safe_exec:
3594   {
3595     \__enumext_is_not_nested:
3596     \__enumext_internal_mini_page:
3597     \int_incr:N \l__enumext_level_int
3598     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3599       { \msg_fatal:nn { enumext } { list-too-deep } }
3600     \bool_set_true:N \l__enumext_standar_bool
3601     \bool_set_false:N \l__enumext_starred_bool
3602     \__enumext_is_on_first_level:
3603   }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_str used by the key `series` and then we check if we are at the “first level”, if so we process the `<keys>` and then execute the function __enumext_parse_series:n used by the key `series` and call the function __enumext_nested_base_line_fix: used by the key `base-fix`, otherwise we will pass the `<keys>` to the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the `<keys>` to pass them to the `sequence` if the key `save-key` is not active.

```

3604 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3605   {
3606     \tl_if_novalue:nF {#1}
3607     {
3608       \str_clear:N \l__enumext_series_str
3609       \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3610       {
3611         \keys_set:nn { enumext / level-1 } {#1}
3612         \__enumext_parse_series:n {#1}
3613         \__enumext_nested_base_line_fix:
3614       }
3615       {
3616         \exp_args:Ne \keys_set:nn
3617           { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3618       }

```

```

3619         \__enumext_store_active_keys:n {#1}
3620     }
3621 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level: The __enumext_start_store_level: function activate the “*storing structure*” mechanism in the *sequence* for the command \anskey and the environment anskey*.

```

3622 \cs_new_protected:Nn \__enumext_start_store_level:
3623 {
3624     \bool_lazy_all:nT
3625     {
3626         { \bool_if_p:N \__enumext_store_active_bool }
3627         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3628         { \bool_if_p:N \g__enumext_standar_bool }
3629     }
3630     {
3631         \int_compare:nNnT { \__enumext_level_int } > { 1 }
3632         {
3633             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3634             \__enumext_store_level_open:
3635         }
3636     }

```

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

```

3637     \bool_lazy_all:nT
3638     {
3639         { \bool_if_p:N \__enumext_store_active_bool }
3640         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3641         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3642     }
3643     {
3644         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3645         {
3646             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3647             \__enumext_store_level_open:
3648         }
3649     }
3650 }

```

(End of definition for __enumext_start_store_level:.)

__enumext_stop_store_level: The __enumext_stop_store_level: function stop the “*storing structure*” mechanism in the *sequence* for the command \anskey and the environment anskey*.

```

3651 \cs_new_protected:Nn \__enumext_stop_store_level:
3652 {
3653     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3654     {
3655         \__enumext_store_level_close:
3656     }
3657 }

```

(End of definition for __enumext_stop_store_level:.)

__enumext_multicols_start: The function __enumext_multicols_start: will start the multicols environment according to the value passed by the columns key, then set the default value for \columnsep when columns-sep=opt and set the value of \multicolsep equal to zero and leave \columnseprule equal to zero for inner levels.

```

3658 \cs_new_protected:Nn \__enumext_multicols_start:
3659 {
3660     \int_compare:nNnT
3661     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3662     {
3663         \dim_compare:nNnT
3664         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3665         {
3666             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3667             {
3668                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3669                 + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3670                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3671                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }

```

```

3672     }
3673   }
3674   \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3675   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3676   {
3677     \dim_zero:N \columnseprule
3678   }

```

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.

```

3679   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3680   {
3681     \skip_zero:N \multicolsep
3682     \__enumext_multi_addvspace:
3683   }
3684   \raggedcolumns
3685   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3686 }
3687 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment and apply our “*vertical adjust*” spacing. For compatibility with *tagged* PDF, the closing of the `list` environment is executed here along with `__enumext_stop_store_level:`.

```

3688 \cs_new_protected:Nn \__enumext_multicols_stop:
3689 {
3690   \int_compare:nNnTF
3691   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3692   {
3693     \__enumext_stop_list:
3694     \__enumext_stop_store_level:
3695     \end{multicols}
3696     \__enumext_unskip_unkern:
3697     \__enumext_unskip_unkern:
3698     \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3699   }
3700   {
3701     \__enumext_stop_list:
3702     \__enumext_stop_store_level:
3703   }
3704 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_before_list:` The function `__enumext_before_list:` first calls the function `__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `__enumext_before_args_exec:` used by the key `before*` and finally execute the function `__enumext_check_ans_active:` for the check answer mechanism.

```

3705 \cs_new_protected:Nn \__enumext_before_list:
3706 {
3707   \__enumext_vspace_above:
3708   \__enumext_before_args_exec:
3709   \__enumext_check_ans_active:

```

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

```

3710   \dim_compare:nNnT
3711   { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3712   {
3713     \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3714     {
3715       \linewidth
3716       - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3717       - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3718     }

```

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

```

3719         \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3720         \int_gincr:N \g__enumext_minipage_stat_int
3721         \__enumext_minipage_add_space:
3722         \noindent
3723         \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3724     }
3725     \__enumext_multicols_start:
3726 }

```

(End of definition for `__enumext_before_list:`)

`__enumext_second_part:` The function `__enumext_second_part:` first check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_page` environment has not been closed), then close `__enumext_mini_page` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3727 \cs_new_protected:Nn \__enumext_second_part:
3728 {
3729     \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3730     {
3731         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3732         {
3733             \msg_warning:nn { enumext } { missing-miniright }
3734             \miniright
3735         }
3736         \int_gzero:N \g__enumext_minipage_stat_int
3737         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3738         \end__enumext_mini_page
3739     }
3740     {
3741         \__enumext_multicols_stop:
3742     }

```

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.

```

3743     \__enumext_after_stop_list:
3744     \__enumext_check_ans_key_hook:
3745     \__enumext_vspace_below:
3746     \bool_set_false:N \l__enumext_standar_bool
3747     \__enumext_resume_save_counter:
3748 }

```

(End of definition for `__enumext_second_part:`)

`__enumext_set_item_width:` The function `__enumext_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key for each level of the environment.

```

3749 \cs_new_protected:Nn \__enumext_set_item_width:
3750 {
3751     \dim_set:Nn \itemwidth { \linewidth }
3752     \dim_compare:nT
3753     {
3754         \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3755     }
3756     {
3757         \dim_sub:Nn \itemwidth
3758         {
3759             \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3760         }
3761     }
3762 }

```

(End of definition for `__enumext_set_item_width:`)

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

```

3763 \NewDocumentEnvironment{enumext}{0}{ }
3764 {
3765   \__enumext_safe_exec:
3766   \__enumext_parse_keys:n {#1}
3767   \__enumext_before_list:
3768   \__enumext_start_store_level:
3769   \__enumext_start_list:nn
3770   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3771   {
3772     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3773     \__enumext_before_keys_exec:
3774   }
3775   \__enumext_set_item_width:
3776   \__enumext_after_args_exec:
3777 }
3778 {
3779   \__enumext_second_part:
3780 }

```

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

As we don't want our check to be executed **check-ans** by levels but on the complete list, we will take it out of the **enumext** environment using the “hook” function `__enumext_after_env:nn`.

```

3781 \__enumext_after_env:nn {enumext}
3782 {
3783   \__enumext_execute_after_env:
3784 }

```

13.40 The environment keyans

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

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

```

3785 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3786 {
3787   \bool_if:NF \__enumext_store_active_bool
3788   {
3789     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3790   }
3791   \int_incr:N \__enumext_keyans_level_int
3792   \bool_set_true:N \__enumext_keyans_env_bool
3793   \__enumext_keyans_name_and_start:
3794   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3795   \bool_set_false:N \__enumext_store_active_bool
3796   \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
3797   {
3798     \msg_error:nn { enumext } { keyans-nested }
3799   }
3800   \int_compare:nNnT { \__enumext_level_int } > { 1 }
3801   {
3802     \msg_error:nn { enumext } { keyans-wrong-level }
3803   }
3804 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

3805 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3806 {
3807   \keys_set:nn { enumext / keyans } {#1}
3808 }

```

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

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

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_second_part_v:
3809 \cs_new_protected:Nn \__enumext_before_list_v:
3810 {
3811   \__enumext_vspace_above_v:
3812   \__enumext_before_args_exec_v:
3813   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3814   {
3815     \dim_set:Nn \l__enumext_minipage_left_v_dim
3816     {
3817       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3818     }
3819     \bool_set_true:N \l__enumext_minipage_active_v_bool
3820     \int_gincr:N \g__enumext_minipage_stat_int
3821     \__enumext_keyans_minipage_add_space:
3822     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3823   }
3824   \__enumext_keyans_multicols_start:
3825 }
3826 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3827 {
3828   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3829   {
3830     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3831     {
3832       \dim_set:Nn \l__enumext_columns_sep_v_dim
3833       {
3834         (
3835           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3836         ) / \l__enumext_columns_v_int
3837         - \l__enumext_listoffset_v_dim
3838       }
3839     }
3840     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3841     \dim_zero:N \columnseprule % no rule here
3842     \bool_if:NF \l__enumext_minipage_active_v_bool
3843     {
3844       \skip_zero:N \multicolsep
3845       \__enumext_keyans_multi_addvspace:
3846     }
3847     \raggedcolumns
3848     \begin{multicols}{\l__enumext_columns_v_int}
3849   }
3850 }
3851 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3852 {
3853   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3854   {
3855     \__enumext_stop_list:
3856     \end{multicols}
3857     \__enumext_unskip_unkern:
3858     \__enumext_unskip_unkern:
3859     \par\addvspace{ \l__enumext_multicols_below_v_skip }
3860   }
3861   {
3862     \__enumext_stop_list:
3863   }
3864 }
3865 \cs_new_protected:Nn \__enumext_second_part_v:
3866 {
3867   \bool_if:NTF \l__enumext_minipage_active_v_bool
3868   {
3869     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3870     {
3871       \msg_warning:nn { enumext } { missing-miniright }
3872       \miniright
3873     }
3874     \int_gzero:N \g__enumext_minipage_stat_int
3875     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3876     \end__enumext_mini_page
3877     \par\addvspace{ \l__enumext_minipage_after_skip }
3878   }

```

```

3879     {
3880         \__enumext_keyans_multicols_stop:
3881     }
3882     \bool_set_false:N \__enumext_keyans_env_bool
3883     \__enumext_after_stop_list_v:
3884     \__enumext_vspace_below_v:
3885 }

```

(End of definition for __enumext_before_list_v: and others.)

__enumext_keyans_set_item_width:

The function __enumext_keyans_set_item_width: will set the value of \itemwidth taking into account the value established by the list-offset key.

```

3886 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3887 {
3888     \dim_set:Nn \itemwidth { \linewidth }
3889     \dim_compare:nT
3890     {
3891         \__enumext_listoffset_v_dim != \c_zero_dim
3892     }
3893     {
3894         \dim_sub:Nn \itemwidth { \__enumext_listoffset_v_dim }
3895     }
3896 }

```

(End of definition for __enumext_keyans_set_item_width:.)

keyans

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

```

3897 \NewDocumentEnvironment{keyans}{0}{}
3898 {
3899     \__enumext_keyans_safe_exec:
3900     \__enumext_keyans_parse_keys:n {#1}
3901     \__enumext_before_list_v:
3902     \__enumext_start_list:nn
3903     { \tl_use:N \__enumext_label_v_tl }
3904     {
3905         \__enumext_list_arg_two_v:
3906         \__enumext_before_keys_exec_v:
3907     }
3908     \__enumext_keyans_set_item_width:
3909     \__enumext_after_args_exec_v:
3910 }
3911 {
3912     \__enumext_check_starred_cmd:n { item }
3913     \__enumext_second_part_v:
3914 }

```

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

13.41 Tagging PDF support for non-standart list environments

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

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

13.41.1 Socket for tagging support in enumext* and keyans*

start-list-tags

stop-start-tags

stop-list-tags

__enumext_start_list_tag:n

__enumext_stop_start_list_tag:

__enumext_stop_list_tag:n

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```

3915 \socket_new:nn {taggsupport/enumext/starred}{1}
3916 \socket_new_plug:nnn {taggsupport/enumext/starred} {start-list-tags}
3917 {
3918     \tag_resume:n {#1}
3919     \tag_struct_begin:n {tag=LI}
3920     \tag_struct_begin:n {tag=Lbl}
3921     \tag_mc_begin:n {tag=Lbl}
3922 }
3923 \socket_new_plug:nnn {taggsupport/enumext/starred} {stop-start-tags}
3924 {

```

```

3925     \tag_mc_end:
3926     \tag_struct_end:n {tag=Lbl}
3927     \tag_struct_begin:n {tag=LBody}
3928     \tag_struct_begin:n {tag=text-unit}
3929     \tag_struct_begin:n {tag=text}
3930   }
3931   \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3932   {
3933     \tag_struct_end:n {tag=text}
3934     \tag_struct_end:n {tag=text-unit}
3935     \tag_struct_end:n {tag=LBody}
3936     \tag_struct_end:n {tag=LI}
3937     \tag_suspend:n {#1}
3938   }

```

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

```

3939   \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3940   {
3941     \IfDocumentMetadataTF
3942     {
3943       \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3944       \socket_use:n {tagsupport/enumext/starred} {#1}
3945     } {}
3946   }
3947   \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3948   {
3949     \IfDocumentMetadataTF
3950     {
3951       \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3952       \socket_use:nn {tagsupport/enumext/starred} { }
3953     } {}
3954   }
3955   \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3956   {
3957     \IfDocumentMetadataTF
3958     {
3959       \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3960       \socket_use:nn {tagsupport/enumext/starred} {#1}
3961     } {}
3962   }

```

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

13.41.2 Socket for tagging support in keyanspic

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

```

start-list-tags
stop-start-tags
stop-list-tags
\__enumext_anspic_start_list_tag:
\__enumext_anspic_stop_start_list_tag:
\__enumext_anspic_stop_list_tag:
3963   \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3964   \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3965   {
3966     \tag_resume:n {keyanspic}
3967     \tag_struct_begin:n {tag=LI}
3968     \tag_struct_begin:n {tag=Lbl}
3969     \tag_mc_begin:n {tag=Lbl}
3970   }
3971   \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3972   {
3973     \tag_mc_end:
3974     \tag_struct_end:n {tag=Lbl}
3975     \tag_struct_begin:n {tag=LBody}
3976     \tag_struct_begin:n {tag=text-unit}
3977     \tag_struct_begin:n {tag=text}
3978     \tag_mc_begin:n {tag=text}
3979   }
3980   \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3981   {
3982     \tag_mc_end:
3983     \tag_struct_end:n {tag=text-unit}
3984     \tag_struct_end:n {tag=text}
3985     \tag_struct_end:n {tag=LBody}
3986     \tag_struct_end:n {tag=LI}
3987     \tag_suspend:n {keyanspic}
3988   }

```

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

```

3989 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3990 {
3991   \IfDocumentMetadataTF
3992   {
3993     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3994     \socket_use:n {tagsupport/enumext/keyanspic}
3995   } {}
3996 }
3997 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3998 {
3999   \IfDocumentMetadataTF
4000   {
4001     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
4002     \socket_use:nn {tagsupport/enumext/keyanspic}
4003   } {}
4004 }
4005 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4006 {
4007   \IfDocumentMetadataTF
4008   {
4009     \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
4010     \socket_use:nn {tagsupport/enumext/keyanspic}
4011   } {}
4012 }

```

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

13.42 The environment `keyanspic` and `\anspic`

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

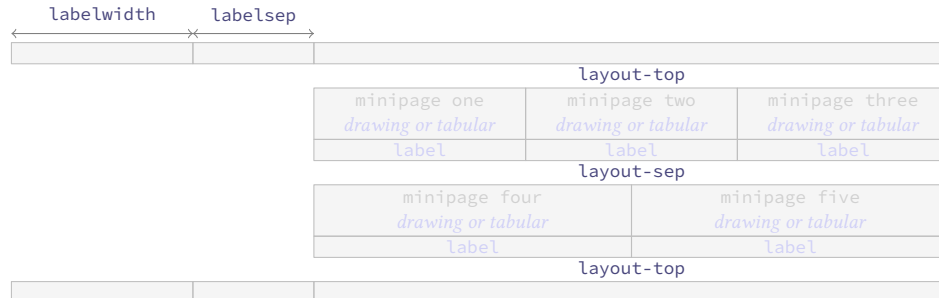


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

13.42.1 The environment `keyanspic`

First we define the key that allows us to process the position of the `\label` centered “*above*” or “*below*” which will be `label-pos`, the vertical separation of these from *drawing or tabular* will be handled with the key `label-sep`. The “*layout style*” will be handled with the key `layout-sty` will take two values separated by comma `{n° upper, n° lower}` and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “*upper*” and “*lower*” within the environments separated by the value of the key `layout-sep`. The vertical space “*top*” and “*bottom*” of the environment will be handled with the key `layout-top`.

```

4013 \keys_define:nn { enumext / keyanspic }
4014 {
4015   label-pos .choice:,
4016   label-pos / above .code:n =
4017     \bool_set_true:N \l__enumext_anspic_label_above_bool
4018     \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4019   label-pos / below .code:n =
4020     \bool_set_false:N \l__enumext_anspic_label_above_bool
4021     \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4022   label-pos / unknown .code:n =

```

```

4023             \msg_error:nneee { enumext } { unknown-choice }
4024             { label-pos } { above,~ below } { \exp_not:n {#1} },
4025     label-pos .initial:n      = below,
4026     label-pos .value_required:n = true,
4027     label-sep .skip_set:N     = \l__enumext_anspic_label_sep_skip,
4028     label-sep .value_required:n = true,
4029     layout-sty .tl_set:N      = \l__enumext_anspic_layout_style_tl,
4030     layout-sty .value_required:n = true,
4031     layout-sep .code:n        = \keys_set:nn { enumext / keyans }
4032                               { parsep = #1 },
4033     layout-sep .value_required:n = true,
4034     layout-top .code:n        = \keys_set:nn { enumext / keyans }
4035                               { topsep = #1 },
4036     layout-top .value_required:n = true,
4037     unknown .code:n           = { \l__enumext_keyans_unknown_keys:n {#1} }
4038 }

```

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

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

```

\__enumext_keyans_pic_parse_keys:n
\__enumext_keyans_pic_skip_abs:N
\__enumext_keyans_pic_arg_two:
4039 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4040 {
4041   \int_incr:N \l__enumext_keyans_pic_level_int
4042   \int_compare:nNtT { \l__enumext_keyans_pic_level_int } > { 1 }
4043   {
4044     \msg_error:nn { enumext } { keyanspic-nested }
4045   }
4046   \__enumext_keyans_name_and_start:
4047 }

```

Parse [`<key = val>`] for `keyanspic` environment.

```

4048 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4049 {
4050   \tl_if_novalue:nF {#1}
4051   {
4052     \keys_set:nn { enumext / keyanspic } {#1}
4053   }
4054 }

```

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

```

4055 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4056 {
4057   \dim_compare:nNtT { #1 } < { \c_zero_dim }
4058   {
4059     \skip_set:Nn #1 { -#1 }
4060   }
4061 }

```

The `__enumext_keyans_pic_arg_two:` function will be used in the *second argument* of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the “spaces” and the keys `label`, `wrap-label`, `parsep` and `topsep` from the `keyans` environment. The first thing we need to do is set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to “false”, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```

4062 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4063 {
4064   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4065   \__enumext_list_arg_two_v:
4066   \__enumext_keyans_pic_skip_abs:N \parsep

```

Now we increment the counter `enumXv` of the `keyans` environment and save the *total height* of the `<label>` in `\l__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the key `label-pos` is set to *below*.

```

4067   \bool_if:nF \l__enumext_anspic_label_above_bool
4068   {
4069     \stepcounter { enumXv }
4070     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4071     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4072     {
4073       \box_ht_plus_dp:N \l__enumext_anspic_label_box

```

```

4074     }
4075     \skip_add:Nn \parsep
4076     {
4077         \l__enumext_anspic_label_htdp_dim
4078         + \box_dp:N \strutbox
4079         + \l__enumext_anspic_label_sep_skip
4080     }
4081 }

```

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

```

4082     \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
4083     \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4084     \dim_zero:N \listparindent
4085     \skip_zero:N \partopsep
4086     \skip_zero:N \itemsep
4087 }

```

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

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

```

4088 \NewDocumentEnvironment{keyanspic}{o}
4089 {
4090     \__enumext_keyans_pic_safe_exec:
4091     \__enumext_keyans_pic_parse_keys:n {#1}
4092     \begin{list} {} { \__enumext_keyans_pic_arg_two: }
4093     \IfDocumentMetadataTF
4094     {
4095         \tag_suspend:n {list}
4096     }{}
4097     \item[] \scan_stop:
4098     % paranoia
4099     \RenewDocumentCommand \item {}
4100     {
4101         \msg_error:nn { enumext } { keyanspic-item-cmd }
4102     }
4103     \IfDocumentMetadataTF
4104     {
4105         \tag_resume:n {keyanspic}
4106         \tag_tool:n {para/tagging=false}
4107         \tag_suspend:n {keyanspic}
4108     } { }
4109 }
4110 {
4111     \IfDocumentMetadataTF
4112     {
4113         \tag_resume:n {keyanspic}
4114         \tag_struct_begin:n {tag=L,attribute=enumerate}
4115     } { }

```

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

```

4116     \__enumext_anspic_exec:
4117     \IfDocumentMetadataTF
4118     {
4119         \tag_suspend:n {keyanspic}
4120     } { }
4121     \end{list}
4122     \IfDocumentMetadataTF
4123     {
4124         \tag_struct_end:
4125         \tag_struct_end:
4126     } { }

```

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

```

4127     \__enumext_check_starred_cmd:n { anspic }
4128     \setcounter { enumXvi } { 0 }
4129     \bool_if:NTF \l__enumext_anspic_label_above_bool
4130     {
4131         \par\addvspace{ 0.5\box_dp:N \strutbox }

```

```

4132     }
4133     {
4134         \par
4135         \addvspace
4136         {
4137             \dim_eval:n
4138             {
4139                 \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4140                 + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4141             }
4142         }
4143     }
4144 }

```

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

13.42.2 The command `\anspic`

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

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

`\anspic`

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

```

4145 \NewDocumentCommand \anspic { s o +m }
4146 {
4147     \bool_if:NF \l__enumext_store_active_bool
4148     {
4149         \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4150     }
4151     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4152     {
4153         \msg_error:nn { enumext } { keyanspic-wrong-level }
4154     }
4155     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4156     {
4157         \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4158     }
4159     \seq_put_right:Nn \l__enumext_anspic_args_seq
4160     {
4161         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4162     }
4163 }

```

The `__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4164 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4165 {
4166     \bool_if:NF \l__enumext_anspic_label_above_bool
4167     {
4168         \IfDocumentMetadataTF
4169         {
4170             \tag_suspend:n {keyanspic}
4171         } { }
4172         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4173         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4174         {
4175             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4176         }
4177         \IfDocumentMetadataTF
4178         {
4179             \tag_resume:n {keyanspic}
4180         } { }
4181     }
4182 }

```


The `__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* ‘*’ and *optional argument* passed to the command. Here we will store the `\label` and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label` and `wrap-opt` keys.

```

4183 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4184 {
4185   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4186   {
4187     \bool_if:nT { #1 }
4188     {
4189       \__enumext_keyans_addto_prop:n { #2 }
4190       \__enumext_keyans_store_ref:
4191       \__enumext_keyans_addto_seq:n { #2 }
4192       \int_gincr:N \g__enumext_check_starred_cmd_int
4193       \bool_lazy_or:nnT
4194       { \bool_if_p:N \l__enumext_show_answer_bool }
4195       { \bool_if_p:N \l__enumext_show_position_bool }
4196       {
4197         \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4198         \__enumext_keyans_show_left:n { #2 }
4199         \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4200       }
4201     }
4202     \tl_use:N \l__enumext_label_font_style_v_tl
4203     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4204     \__enumext_keyans_show_item_opt:
4205   }
4206 }

```

The function `__enumext_anspic_label_pos:nnn` will be in charge of handling the “*counter*” and the position of the `\label`, set by `label-pos` key which will have the same configuration as the `keyanspic` environment.

```

4207 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4208 {
4209   \stepcounter { enumXvi }
4210   \__enumext_anspic_body_dim:n { #3 }
4211   \bool_if:NTF \l__enumext_anspic_label_above_bool
4212   {
4213     \__enumext_anspic_label:nn { #1 } { #2 }
4214   }
4215   {
4216     \raisebox
4217     {
4218       -\dim_eval:n
4219       {
4220         \l__enumext_anspic_label_htdp_dim
4221         + \l__enumext_anspic_body_htdp_dim
4222         + \box_dp:N \strutbox
4223         + \l__enumext_anspic_label_sep_skip
4224       }
4225     }
4226     [ opt ] [ opt ]
4227     {
4228       \__enumext_anspic_label:nn { #1 } { #2 }
4229     }
4230   }
4231 }
4232 %

```

The `__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```

4233 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4234 {
4235   \__enumext_anspic_start_list_tag:
4236   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4237   \__enumext_anspic_stop_start_list_tag:
4238   \bool_if:NTF \l__enumext_anspic_label_above_bool
4239   {
4240     \[\[ \l__enumext_anspic_label_sep_skip] #3
4241   }
4242   {
4243     \[ #3

```

```

4244     }
4245     \__enumext_anspic_stop_list_tag:
4246 }

```

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

```

4247 \cs_new_protected:Nn \__enumext_anspic_print:n
4248 {
4249     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4250 }
4251 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }

```

The function `__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic` saved in the `\l__enumext_anspic_args_seq` sequence inside them.

```

4252 \cs_new_protected:Nn \__enumext_anspic_row:n
4253 {
4254     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4255     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4256     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4257     \int_step_inline:nnn
4258     { \l__enumext_anspic_above_int + 1 }
4259     { \l__enumext_anspic_below_int }
4260     {
4261         \IfDocumentMetadataTF
4262         {
4263             \tag_suspend:n {minipage}
4264         } { }
4265         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4266             \centering
4267             \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4268             \end{minipage}
4269         \IfDocumentMetadataTF
4270         {
4271             \tag_resume:n {minipage}
4272         } { }
4273     }
4274     \par
4275 }

```

The `__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the `keyanspic` environment definition. If the key `layout-sty` is not set, everything will be printed on a *single line*.

```

4276 \cs_new_protected:Nn \__enumext_anspic_exec:
4277 {
4278     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4279     {
4280         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4281     }
4282     {
4283         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4284     }
4285 }

```

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

13.43 The horizontal environments

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

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

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

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

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

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

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

13.43.1 Functions for item box width

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:
```

```
4286 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4287 {
4288   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4289   {
4290     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4291     {
4292       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4293       / \l__enumext_columns_vii_int
4294     }
4295   }
4296   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4297   \dim_set:Nn \l__enumext_item_width_vii_dim
4298   {
4299     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4300     / \l__enumext_columns_vii_int
4301     - \l__enumext_labelwidth_vii_dim
4302     - \l__enumext_labelsep_vii_dim
4303   }
```

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

```
4304   \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4305   {
4306     \dim_sub:Nn \l__enumext_item_width_vii_dim
4307     {
4308       ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4309       / \l__enumext_columns_vii_int
4310     }
4311     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4312     {
4313       \l__enumext_rightmargin_vii_dim
4314     }
4315   }
4316 }
```

Same implementation for the `keyans*` environment.

```
4317 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4318 {
4319   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4320   {
4321     \dim_set:Nn \l__enumext_columns_sep_viii_dim
4322     {
4323       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4324       / \l__enumext_columns_viii_int
4325     }
4326   }
4327   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4328   \dim_set:Nn \l__enumext_item_width_viii_dim
4329   {
4330     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4331     / \l__enumext_columns_viii_int
4332     - \l__enumext_labelwidth_viii_dim
4333     - \l__enumext_labelsep_viii_dim
4334   }
4335   \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4336   {
4337     \dim_sub:Nn \l__enumext_item_width_viii_dim
4338     {
4339       ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
4340       / \l__enumext_columns_viii_int
4341     }
4342     \dim_add:Nn \l__enumext_columns_sep_viii_dim
4343     {
4344       \l__enumext_rightmargin_viii_dim
4345     }
4346   }
```

```

4346     }
4347 }

```

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

13.43.2 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.

```

4348 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4349 {
4350   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4351   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4352   {
4353     \msg_warning:nnee { enumext } { item-joined }
4354     { \int_use:N \l__enumext_joined_item_vii_int }
4355     { \int_use:N \l__enumext_columns_vii_int }
4356     \int_set:Nn \l__enumext_joined_item_vii_int
4357     {
4358       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4359     }
4360   }
4361   \int_compare:nNnT
4362   { \l__enumext_joined_item_vii_int }
4363   >
4364   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4365   {
4366     \msg_warning:nnee { enumext } { item-joined-columns }
4367     { \int_use:N \l__enumext_joined_item_vii_int }
4368     {
4369       \int_eval:n
4370       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4371     }
4372     \int_set:Nn \l__enumext_joined_item_vii_int
4373     {
4374       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4375     }
4376   }
4377   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4378   {
4379     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4380     \int_decr:N \l__enumext_joined_item_aux_vii_int
4381     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4382     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4383     \dim_set:Nn \l__enumext_joined_width_vii_dim
4384     {
4385       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4386       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4387         + \l__enumext_columns_sep_vii_dim
4388       ) * \l__enumext_joined_item_aux_vii_int
4389     }
4390     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4391   }
4392   {
4393     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4394     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4395   }
4396 }

```

Same implementation for the `keyans*` environment.

```

4397 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4398 {
4399   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4400   \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4401   {
4402     \msg_warning:nnee { enumext } { item-joined }
4403     { \int_use:N \l__enumext_joined_item_viii_int }
4404     { \int_use:N \l__enumext_columns_viii_int }
4405     \int_set:Nn \l__enumext_joined_item_viii_int
4406     {
4407       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1

```

```

4408     }
4409   }
4410   \int_compare:nNt
4411   { \l__enumext_joined_item_viii_int }
4412   >
4413   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4414   {
4415     \msg_warning:nnee { enumext } { item-joined-columns }
4416     { \int_use:N \l__enumext_joined_item_viii_int }
4417     {
4418       \int_eval:n
4419       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4420     }
4421     \int_set:Nn \l__enumext_joined_item_viii_int
4422     {
4423       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4424     }
4425   }
4426   \int_compare:nNtF { \l__enumext_joined_item_viii_int } > { 1 }
4427   {
4428     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4429     \int_decr:N \l__enumext_joined_item_aux_viii_int
4430     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4431     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4432     \dim_set:Nn \l__enumext_joined_width_viii_dim
4433     {
4434       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4435       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4436         + \l__enumext_columns_sep_viii_dim
4437       ) * \l__enumext_joined_item_aux_viii_int
4438     }
4439     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4440   }
4441   {
4442     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4443     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4444   }
4445 }

```

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

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

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

```

4446 \cs_new_protected:Nn \__enumext_start_mini_vii:
4447 {
4448   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4449   {
4450     \dim_set:Nn \l__enumext_minipage_left_vii_dim
4451     {
4452       \linewidth
4453       - \l__enumext_minipage_right_vii_dim
4454       - \l__enumext_minipage_hsep_vii_dim
4455     }
4456     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4457     \dim_gset_eq:NN
4458     \g__enumext_minipage_right_vii_dim
4459     \l__enumext_minipage_right_vii_dim
4460     \__enumext_mini_addvspace_vii:
4461     \nointerlineskip\noindent
4462     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4463   }
4464 }

```

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

command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```

4465 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4466 {
4467   \bool_if:NTF \l__enumext_minipage_active_vii_bool
4468   {
4469     \__enumext_stop_list:
4470     \__enumext_stop_store_level_vii:
4471     \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4472     \end__enumext_mini_page
4473     \hfill
4474     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4475   }
4476   {
4477     \__enumext_stop_list:
4478     \__enumext_stop_store_level_vii:
4479   }
4480 }

```

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

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `minipage` environment on the “*right side*”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

4481 \__enumext_after_env:nn {enumext*}
4482 {
4483   \bool_if:NT \g__enumext_minipage_active_vii_bool
4484   {
4485     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4486     \legacy_if_gset_false:n { @minipage }
4487     \skip_vertical:N \c_zero_skip
4488     \par\addvspace { \g__enumext_minipage_right_skip }
4489     \bool_if:NF \g__enumext_minipage_center_vii_bool
4490     {
4491       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4492       {
4493         \centering
4494       }
4495     }
4496     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4497     {
4498       \tl_use:N \g__enumext_miniright_code_vii_tl
4499     }
4500     \box_use_drop:N \l__enumext_miniright_code_vii_box
4501     \skip_vertical:N \c_zero_skip
4502     \__enumext_endminipage:
4503     \par\addvspace{ \g__enumext_minipage_after_skip }
4504   }
4505   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4506   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4507   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4508   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4509   \bool_gset_false:N \g__enumext_starred_bool
4510 }

```

`__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.

```

4511 \cs_new_protected:Nn \__enumext_start_mini_viii:
4512 {
4513   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4514   {
4515     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4516     {
4517       \linewidth
4518       - \l__enumext_minipage_right_viii_dim
4519       - \l__enumext_minipage_hsep_viii_dim
4520     }
4521     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4522     \dim_gset_eq:NN
4523     \g__enumext_minipage_right_viii_dim

```

```

4524         \l__enumext_minipage_right_viii_dim
4525         \__enumext_mini_addvspace_viii:
4526         \nointerlineskip\noindent
4527         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4528     }
4529 }
4530 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4531 {
4532     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4533     {
4534         \__enumext_stop_list:
4535         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4536         \end__enumext_mini_page
4537         \hfill
4538         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4539     }
4540     {
4541         \__enumext_stop_list:
4542     }
4543 }
4544 \__enumext_after_env:nn {keyans*}
4545 {
4546     \bool_if:NT \g__enumext_minipage_active_viii_bool
4547     {
4548         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4549         \par\addvspace { \g__enumext_minipage_right_skip }
4550         \bool_if:NF \g__enumext_minipage_center_viii_bool
4551         {
4552             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4553             {
4554                 \centering
4555             }
4556         }
4557         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4558         {
4559             \tl_use:N \g__enumext_miniright_code_viii_tl
4560         }
4561         \box_use_drop:N \l__enumext_miniright_code_viii_box
4562         \end__enumext_mini_page
4563         \par\addvspace{ \g__enumext_minipage_after_skip }
4564     }
4565     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4566     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4567     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4568     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4569 }

```

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

13.44 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the **shortlst** package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```

4570 \NewDocumentEnvironment{enumext*}{o }
4571 {
4572     \__enumext_safe_exec_vii:
4573     \__enumext_parse_keys_vii:n {#1}
4574     \__enumext_before_list_vii:
4575     \__enumext_start_store_level_vii:
4576     \__enumext_start_list:nn { }
4577     {
4578         \__enumext_list_arg_two_vii:
4579         \__enumext_before_keys_exec_vii:
4580     }
4581     \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4582     \__enumext_starred_columns_set_vii:
4583     \item[] \scan_stop:

```



```

4584     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4585     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4586     \ignorespaces
4587   }
4588   {
4589     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4590     \__enumext_stop_item_tmp_vii:
4591     \__enumext_remove_extra_parsep_vii:
4592     \__enumext_after_list_vii:
4593   }

```

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

`__enumext_safe_exec_vii:` We will first call the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are NOT nested within `enumext`, then call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

4594 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4595   {
4596     \__enumext_is_not_nested:
4597     \__enumext_internal_mini_page:
4598     \int_incr:N \l__enumext_level_h_int
4599     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4600       {
4601         \msg_error:nn { enumext } { nested }
4602       }
4603     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4604       {
4605         \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4606       }
4607     \bool_set_true:N \l__enumext_starred_bool
4608     \bool_set_false:N \l__enumext_standar_bool
4609     \__enumext_is_on_first_level:
4610   }

```

(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.

```

4611 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4612   {
4613     \tl_if_novalue:nF {#1}
4614     {
4615       \str_clear:N \l__enumext_series_str
4616       \keys_set:nn { enumext / enumext* } {#1}
4617       \__enumext_parse_series:n {#1}
4618       \__enumext_store_active_keys_vii:n {#1}
4619     }
4620   }

```

(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*`.

```

4621 \cs_new_protected:Nn \__enumext_before_list_vii:
4622   {
4623     \__enumext_vspace_above_vii:
4624     \__enumext_check_ans_active:
4625     \__enumext_before_args_exec_vii:
4626     \__enumext_start_mini_vii:
4627   }

```

(End of definition for `__enumext_before_list_vii:.`)

`__enumext_after_list_vii:` The function `__enumext_after_list_vii:` first calls the function `__enumext_stop_mini_vii:` which internally calls `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` (§13.43.3) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions `__enumext_after_stop_list_vii:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below_vii:` used by the keys `below` and `below*`. Finally set `\l__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

4628 \cs_new_protected:Nn \__enumext_after_list_vii:
4629 {
4630   \__enumext_stop_mini_vii:
4631   \__enumext_after_stop_list_vii:
4632   \__enumext_check_ans_key_hook:
4633   \__enumext_vspace_below_vii:
4634   \bool_set_false:N \l__enumext_starred_bool
4635   \__enumext_resume_save_counter:
4636 }

```

(End of definition for `__enumext_after_list_vii:`.)

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

`__enumext_stop_store_level_vii:`

```

4637 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4638 {
4639   \bool_if:NT \l__enumext_store_active_bool
4640   {
4641     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4642     {
4643       \__enumext_store_level_open_vii:
4644     }
4645   }
4646 }
4647 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4648 {
4649   \bool_if:NT \l__enumext_store_active_bool
4650   {
4651     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4652     {
4653       \__enumext_store_level_close_vii:
4654     }
4655   }
4656 }

```

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

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

`__enumext_first_item_tmp_vii:`

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

```

4657 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4658 {
4659   \skip_horizontal:n
4660   {
4661     -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
4662   }
4663   \ignorespaces
4664 }

```

(End of definition for `__enumext_first_item_tmp_vii:`.)

`__enumext_start_item_tmp_vii:`

`__enumext_item_peek_args_vii:`

`__enumext_joined_item_vii:w`

`__enumext_standar_item_vii:w`

`__enumext_starred_item_vii:w`

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`.

```

4665 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4666 {
4667   \__enumext_stop_item_tmp_vii:
4668   \int_incr:N \l__enumext_item_column_pos_vii_int
4669   \int_gincr:N \g__enumext_item_count_all_vii_int

```

```

4670     \__enumext_item_peek_args_vii:
4671 }

```

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).

```

4672 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4673 {
4674     \peek_meaning:NTF (
4675     { \__enumext_joined_item_vii:w }
4676     { \__enumext_joined_item_vii:w (1) }
4677 }

```

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`.

```

4678 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4679 {
4680     \__enumext_starred_joined_item_vii:n {#1}
4681     \peek_meaning_remove:NTF *
4682     { \__enumext_starred_item_vii:w }
4683     { \__enumext_standar_item_vii:w }
4684 }

```

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]`.

```

4685 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4686 {
4687     \bool_set_false:N \l__enumext_item_starred_vii_bool
4688     \peek_meaning:NTF [
4689     {
4690         \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4691         \__enumext_start_item_vii:w
4692     }
4693     {
4694         \bool_set_true:N \l__enumext_wrap_label_vii_bool
4695         \legacy_if_set_true:n { @noitemarg }
4696         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4697     }
4698 }

```

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>]`.

```

4699 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4700 {
4701     \bool_set_true:N \l__enumext_item_starred_vii_bool
4702     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4703     \peek_meaning:NTF [
4704     { \__enumext_starred_item_vii_aux_i:w }
4705     { \__enumext_starred_item_vii_aux_ii:w }
4706 }
4707 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4708 {
4709     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4710     \__enumext_starred_item_vii_aux_ii:w
4711 }
4712 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4713 {
4714     \peek_meaning:NTF [
4715     { \__enumext_starred_item_vii_aux_iii:w }
4716     {
4717         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4718         \legacy_if_set_true:n { @noitemarg }
4719         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4720     }
4721 }

```

```

4722 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4723 {
4724   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4725   \legacy_if_set_true:n { @noitemarg }
4726   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4727 }

```

(End of definition for __enumext_start_item_tmp_vii: and others.)

__enumext_fake_make_label_vii:n

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

- For compatibility with *tagged* PDF and *hyperref* when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lua2latex`

```

4728 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4729 {
4730   \legacy_if:nT { @noitemarg }
4731   {
4732     \legacy_if_set_false:n { @noitemarg }
4733     \legacy_if:nT { @nmbrrlist }
4734     {
4735       \IfDocumentMetadataTF
4736       {
4737         \bool_if:NT \l__enumext_hyperref_bool
4738         {
4739           \legacy_if_set_true:n { @hyper@item }
4740         }
4741       } { }
4742       \refstepcounter{enumXvii}
4743       \bool_if:NT \l__enumext_check_answers_bool
4744       {
4745         \int_gincr:N \g__enumext_item_number_int
4746         \bool_set_true:N \l__enumext_item_number_bool
4747       }
4748     }
4749   }
4750   \bool_if:NT \l__enumext_item_starred_vii_bool
4751   {
4752     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4753     {
4754       \tl_gset_eq:NN
4755       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4756     }
4757     \mode_leave_vertical:
4758     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4759     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4760     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4761     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4762   }
4763   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4764   {
4765     \tl_use:N \l__enumext_label_font_style_vii_tl
4766     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4767     {
4768       \__enumext_wrapper_label_vii:n {#1}
4769     }
4770     { #1 }
4771   }
4772   \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4773 }

```

(End of definition for __enumext_fake_make_label_vii:n.)

13.44.2 Real definition of \item in enumext*

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

```
\__enumext_start_item_vii:w
  \__enumext_stop_item_vii:
```

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

```
4774 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4775   {
4776     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4777     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4778     {
4779       \l__enumext_joined_width_vii_dim
4780       + \l__enumext_labelwidth_vii_dim
4781       + \l__enumext_labelsep_vii_dim
4782     }
4783   }
```

Redefine the `\footnote` command.

```
4783   \__enumext_renew_footnote_starred:
```

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

```
4784   \__enumext_start_list_tag:n {enumext*}
4785   \__enumext_fake_make_label_vii:n {#1}
4786   \__enumext_stop_start_list_tag:
```

Finally we open the `minipage` environment, capture the “*item content*”, make `\parindent` take the value of the key `listparindent` and `\parskip` take the value of the key `parsep`, then execute the keys `itemindent` and `first`.

Here the use of `\unskip` and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[⟨opt⟩]` and the value passed to the key `itemindent` is incremented.

```
4787   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4788   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4789   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4790   \__enumext_unskip_unkern:
4791   \__enumext_unskip_unkern:
4792   \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4793   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4794   \tl_use:N \l__enumext_after_list_args_vii_tl
4795 }
```

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

```
4796 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4797   {
4798     \__enumext_endminipage:
4799     \__enumext_stop_list_tag:n {enumext*}
4800     \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print `\item` and “*item content*” from the *horizontal box* and *footnotes*.

```
4801   \int_set:Nn \hbadness { 10000 }
4802   \box_use_drop:N \l__enumext_item_text_vii_box
4803   \__enumext_print_footnote_starred:
```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```
4804   \int_compare:nNnTF
4805   { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4806   {
4807     \par\noindent
4808     \int_zero:N \l__enumext_item_column_pos_vii_int
4809   }
4810   {
4811     \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4812   }
4813 }
```

(End of definition for `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:`)

`__enumext_remove_extra_parsep_vii:`

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

```
4814 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4815 {
4816   \int_compare:nNnT
4817   {
4818     \int_mod:nn
4819     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4820   }
4821   =
4822   { 0 }
4823   {
4824     \para_end:
4825     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4826     \skip_vertical:N \c_zero_skip
4827     \int_gzero:N \g__enumext_item_count_all_vii_int
4828   }
4829 }
```

(End of definition for `__enumext_remove_extra_parsep_vii:`.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “hook” function `__enumext_after_env:nn`.

```
4830 \__enumext_after_env:nn {enumext*}
4831 {
4832   \__enumext_execute_after_env:
4833 }
```

13.45 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```
4834 \NewDocumentEnvironment{keyans*}{ o }
4835 {
4836   \__enumext_safe_exec_viii:
4837   \__enumext_parse_keys_viii:n {#1}
4838   \__enumext_before_list_viii:
4839   \__enumext_start_list:nn { }
4840   {
4841     \__enumext_list_arg_two_viii:
4842     \__enumext_before_keys_exec_viii:
4843   }
4844   \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { } { }
4845   \__enumext_starred_columns_set_viii:
4846   \item[] \scan_stop:
4847   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4848   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4849   \ignorespaces
4850 }
4851 {
4852   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { } { }
4853   \__enumext_stop_item_tmp_viii:
4854   \__enumext_remove_extra_parsep_viii:
4855   \__enumext_check_starred_cmd:n { item }
4856   \__enumext_after_list_viii:
4857 }
```

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

`__enumext_safe_exec_viii:`

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

```
4858 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4859 {
4860   \bool_if:NF \l__enumext_store_active_bool
```

```

4861     {
4862         \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4863     }
4864     \int_incr:N \l__enumext_keyans_level_h_int
4865     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4866     {
4867         \msg_error:nn { enumext } { nested }
4868     }
4869     \__enumext_keyans_name_and_start:
4870     \bool_if:NT \l__enumext_starred_bool
4871     {
4872         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4873     }
4874     \bool_set_true:N \l__enumext_starred_bool
4875     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4876     \bool_set_false:N \l__enumext_store_active_bool
4877     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4878     {
4879         \msg_error:nn { enumext } { keyans-wrong-level }
4880     }
4881 }

```

(End of definition for `__enumext_safe_exec_viii:`)

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

```

4882 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4883 {
4884     \tl_if_novalue:nF {#1}
4885     {
4886         \keys_set:nn { enumext / keyans* } {#1}
4887     }
4888 }

```

(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`.

```

4889 \cs_new_protected:Nn \__enumext_before_list_viii:
4890 {
4891     \__enumext_vspace_above_viii:
4892     \__enumext_before_args_exec_viii:
4893     \__enumext_start_mini_viii:
4894 }

```

(End of definition for `__enumext_before_list_viii:`)

`__enumext_after_list_viii:` The function `__enumext_after_list_viii:` first call the function `__enumext_stop_mini_viii:`, then apply the `{⟨code⟩}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

4895 \cs_new_protected:Nn \__enumext_after_list_viii:
4896 {
4897     \__enumext_stop_mini_viii:
4898     \__enumext_after_stop_list_viii:
4899     \__enumext_vspace_below_viii:
4900 }

```

(End of definition for `__enumext_after_list_viii:`)

13.45.1 The command `\item` in `keyans*`

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

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

```

4901 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4902 {
4903   \skip_horizontal:n
4904   {
4905     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
4906   }
4907   \ignorespaces
4908 }

```

(End of definition for `__enumext_first_item_tmp_viii:`.)

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. `__enumext_item_peek_args_viii:` After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

`__enumext_item_peek_args_viii:`
`__enumext_joined_item_viii:w`
`__enumext_standar_item_viii:w`

```

4909 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4910 {
4911   \__enumext_stop_item_tmp_viii:
4912   \int_incr:N \l__enumext_item_column_pos_viii_int
4913   \int_gincr:N \g__enumext_item_count_all_viii_int
4914   \__enumext_item_peek_args_viii:
4915 }

```

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

```

4916 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4917 {
4918   \peek_meaning:NTF (
4919     { \__enumext_joined_item_viii:w }
4920     { \__enumext_joined_item_viii:w (1) }
4921   )

```

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`.

```

4922 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4923 {
4924   \__enumext_starred_joined_item_viii:n {#1}
4925   \peek_meaning_remove:NTF *
4926   { \__enumext_starred_item_viii:w }
4927   { \__enumext_standar_item_viii:w }
4928 }

```

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

```

4929 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4930 {
4931   \bool_set_false:N \l__enumext_item_starred_viii_bool
4932   \peek_meaning:NTF [
4933     {
4934       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4935       \__enumext_start_item_viii:w
4936     }
4937     {
4938       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4939       \legacy_if_set_true:n { @noitemarg }
4940       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4941     }
4942   }

```

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

```
\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_starred_item_exec:
```

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]`.

```
4943 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4944 {
4945   \bool_set_true:N \l__enumext_item_starred_viii_bool
4946   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4947   \peek_meaning:NTF [
4948     { \__enumext_starred_item_viii_aux_i:w }
4949     { \__enumext_starred_item_viii_aux_ii:w }
4950   }
```

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`.

```
4951 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4952 {
4953   \tl_clear:N \l__enumext_store_current_label_tl
4954   \tl_if_no_value:nF { #1 }
4955   {
4956     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4957     {
4958       \tl_put_right:Ne \l__enumext_store_current_label_tl
4959       {
4960         \l__enumext_store_keyans_item_opt_sep_tl
4961       }
4962       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4963     }
4964     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4965   }
4966   \__enumext_starred_item_viii_aux_ii:w
4967 }
4968 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4969 {
4970   \legacy_if_set_true:n { @noitemarg }
4971   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4972 }
```

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.

```
4973 \cs_new_protected:Nn \__enumext_starred_item_exec:
4974 {
4975   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4976   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4977   \__enumext_keyans_store_ref:
4978   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4979   \__enumext_keyans_addto_seq_link:
4980   \int_gincr:N \g__enumext_check_starred_cmd_int
4981   \bool_if:NT \l__enumext_show_answer_bool
4982   {
4983     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4984   }
4985   \bool_if:NT \l__enumext_show_position_bool
4986   {
4987     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4988     {
4989       \group_begin:
4990       \exp_not:N \normalfont
4991       \exp_not:N \footnotesize [ \int_eval:n
4992         {
4993           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4994         }
4995       ]
4996       \group_end:
4997     }
4998     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4999   }
5000 }
```

(End of definition for `__enumext_starred_item_viii:w` and others.)

`__enumext_fake_make_label_viii:n`

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

```

5001 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5002 {
5003   \legacy_if:nT { @noitemarg }
5004   {
5005     \legacy_if_set_false:n { @noitemarg }
5006     \legacy_if:nT { @nmbrrlist }
5007     {
5008       \refstepcounter{enumXviii}
5009     }
5010   }
5011   \bool_if:NT \l__enumext_item_starred_viii_bool
5012   {
5013     \__enumext_starred_item_exec:
5014   }
5015   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5016   {
5017     \tl_use:N \l__enumext_label_font_style_viii_tl
5018     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5019     {
5020       \__enumext_wrapper_label_viii:n {#1}
5021     }
5022     { #1 }
5023   }
5024   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5025 }

```

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

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

`__enumext_start_item_viii:w`

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

`__enumext_stop_item_viii:`

```

5026 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5027 {
5028   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5029   \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5030   {
5031     \l__enumext_joined_width_viii_dim
5032     + \l__enumext_labelwidth_viii_dim
5033     + \l__enumext_labelsep_viii_dim
5034   }
5035   \__enumext_renew_footnote_standar:
5036   \__enumext_start_list_tag:n {keyans*}
5037   \__enumext_fake_make_label_viii:n {#1}
5038   \__enumext_stop_start_list_tag:
5039   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5040   \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5041   \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5042   \__enumext_unskip_unkern:
5043   \__enumext_unskip_unkern:
5044   \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5045   \tl_use:N \l__enumext_fake_item_indent_viii_tl
5046   \bool_if:NT \l__enumext_item_starred_viii_bool
5047   {
5048     \__enumext_keyans_show_item_opt:
5049   }
5050   \tl_use:N \l__enumext_after_list_args_viii_tl
5051 }
5052 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5053 {
5054   \__enumext_endminipage:
5055   \__enumext_stop_list_tag:n {keyans*}
5056   \hbox_set_end:
5057   \int_set:Nn \hbadness { 10000 }
5058   \box_use_drop:N \l__enumext_item_text_viii_box
5059   \__enumext_print_footnote_standar:
5060   \int_compare:nNnTF
5061   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5062   {
5063     \par\noindent

```

```

5064     \int_zero:N \l__enumext_item_column_pos_viii_int
5065   }
5066   {
5067     \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5068   }
5069 }

```

(End of definition for __enumext_start_item_viii:w and __enumext_stop_item_viii:.)

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

```

5070 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5071 {
5072   \int_compare:nNnT
5073   {
5074     \int_mod:nn
5075     { \g__enumext_item_count_all_viii_int }
5076     { \l__enumext_columns_viii_int }
5077   }
5078   =
5079   { 0 }
5080   {
5081     \para_end:
5082     \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5083     \skip_vertical:N \c_zero_skip
5084     \int_gzero:N \g__enumext_item_count_all_viii_int
5085   }
5086 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

13.46 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* and anskey* from *prop list* defined by save-ans key.

```

5087 \NewDocumentCommand \getkeyans { m }
5088 {
5089   \exp_args:Ne \__enumext_getkeyans_aux:n
5090   { \tl_to_str:e { \text_expand:n {#1} } }
5091 }

```

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

```

5092 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5093 {
5094   \str_if_in:nnTF {#1} { : }
5095   {
5096     \use:e
5097     {
5098       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5099       { {##1} {##2} }
5100     }
5101     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5102   }
5103   { \msg_error:nnn { enumext } { missing-colon } {#1} }
5104 }

```

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*.

```

5105 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5106 {
5107   \prop_if_exist:cTF { g__enumext_#1_prop }
5108   {
5109     \prop_item:cn { g__enumext_#1_prop }{#2}
5110   }
5111   {
5112     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5113   }
5114 }

```

(End of definition for \getkeyans, __enumext_getkeyans_aux:n, and __enumext_getkeyans:nn. This function is documented on page 17.)

13.47 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans` key.

The first thing we will do is define a set of *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

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

```

5115 \keys_define:nn { enumext / print }
5116 {
5117   print* .code:n = \keys_precompile:neN { enumext / enumext* }
5118               { \__enumext_filter_save_key:n {#1} }
5119               \l__enumext_print_keyans_starred_tl, % starred cmd
5120   print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5121   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5122               { \__enumext_filter_save_key:n {#1} }
5123               \l__enumext_print_keyans_i_tl,
5124   print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5125   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5126               { \__enumext_filter_save_key:n {#1} }
5127               \l__enumext_print_keyans_ii_tl,
5128   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5129   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5130               { \__enumext_filter_save_key:n {#1} }
5131               \l__enumext_print_keyans_iii_tl,
5132   print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5133   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
5134               { \__enumext_filter_save_key:n {#1} }
5135               \l__enumext_print_keyans_iv_tl,
5136   print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5137   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
5138               { \__enumext_filter_save_key:n {#1} }
5139               \l__enumext_print_keyans_vii_tl, % starred nested
5140   print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5141 }

```

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

`\printkeyans`

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

`__enumext_printkeyans:nnn`

```

5142 \NewDocumentCommand \printkeyans { s O{ } m }
5143 {
5144   \group_begin:
5145     \tl_use:N \l__enumext_print_keyans_i_tl
5146     \tl_use:N \l__enumext_print_keyans_ii_tl
5147     \tl_use:N \l__enumext_print_keyans_iii_tl
5148     \tl_use:N \l__enumext_print_keyans_iv_tl
5149     \tl_use:N \l__enumext_print_keyans_vii_tl
5150     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5151   \group_end:
5152 }

```

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.

```

5153 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5154 {
5155   \seq_if_exist:cTF { g__enumext_#3_seq }
5156   {
5157     \seq_if_empty:cF { g__enumext_#3_seq }
5158     {

```

If the *starred argument* `*` is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default *keys* for the environment `enumext*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument*

and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```

5159         \bool_if:nTF {#1}
5160         {
5161             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5162             {
5163                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5164             }
5165         {
5166             \tl_use:N \l__enumext_print_keyans_starred_tl
5167             \bool_set_true:N \l__enumext_base_line_fix_bool
5168             \bool_set_true:N \l__enumext_print_keyans_star_bool
5169             \begin{enumext*}[#2]
5170                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5171             \end{enumext*}
5172             \bool_set_false:N \l__enumext_base_line_fix_bool
5173             \bool_set_false:N \l__enumext_print_keyans_star_bool
5174         }
5175     }

```

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

```

5176         {
5177             \begin{enumext}[#2]
5178                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5179             \end{enumext}
5180         }
5181     }
5182 }
5183 {
5184     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5185 }
5186 }

```

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

13.48 The command `\setenumext`

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

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

```

5187 \cs_new:Npn \__enumext_filter_first_level:n #1
5188 {
5189     \use:e
5190     {
5191         \keyval_parse:NNn
5192         \__enumext_filter_first_level_key:n
5193         \__enumext_filter_first_level_pair:nn {#1}
5194     }
5195 }

```

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*`.

```

5196 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5197 {
5198     \str_case:nnF {#1}
5199     {
5200         { resume } {}
5201         { resume* } {}
5202     }
5203     { , { \exp_not:n {#1} } }
5204 }

```

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.

```

5205 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5206 {
5207     \str_case:nnF {#1}
5208     {

```

```

5209     { series } {}
5210     { resume } {}
5211     { save-ans } {}
5212   }
5213   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5214 }

```

(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`.

```

5215 \keys_define:nn { enumext / meta-families }
5216 {
5217   enumext-1 .code:n =
5218     {
5219       \keys_set:ne { enumext / level-1 }
5220       {
5221         \__enumext_filter_first_level:n {#1}
5222       }
5223     } ,
5224   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5225   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5226   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5227   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5228   enumext* .code:n =
5229     {
5230       \keys_set:ne { enumext / enumext* }
5231       {
5232         \__enumext_filter_first_level:n {#1}
5233       }
5234     } ,
5235   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5236   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5237   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5238   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5239   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5240   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5241   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5242   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5243 }

```

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

```

5244 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5245 {
5246   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5247   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5248 }

```

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

```

\__enumext_set_parse:n
\__enumext_set_error:nn
5249 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5250 {
5251   \seq_clear:N \__enumext_setkey_tmpa_seq
5252   \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5253   \int_set:Nn \__enumext_setkey_tmpa_int
5254   {
5255     \seq_count:N \__enumext_setkey_tmpb_seq
5256   }
5257   \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5258   {
5259     \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5260     \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5261     \seq_set_map_e:Nn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5262     {
5263       \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5264     }
5265   }
5266   {
5267     \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5268   }
5269   \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5270   { \seq_map_inline:Nn \c__enumext_all_families_seq }
5271   { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }

```



```

5272     {
5273       \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5274     }
5275   }

```

Internal functions used by the `\setenumext` command.

```

5276 \cs_new_protected:Npn \__enumext_set_parse:n #1
5277 {
5278   \tl_set:Nx \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5279   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5280   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5281   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5282   {
5283     \seq_put_right:Nx \l__enumext_setkey_tmpa_seq
5284     { \tl_trim_spaces:n {#1} }
5285   }
5286   { \__enumext_set_error:nn {#1} { } }
5287 }
5288 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5289 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\setenumext`, `__enumext_set_parse:n`, and `__enumext_set_error:nn`. This function is documented on page 6.)

13.49 The command `\setenumextmeta`

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Add .meta key to existing keys \(l3keys\)](#).

`\setenumextmeta`

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the *optional argument*.

```

\c__enumext_meta_paths_prop
\__enumext_add_meta_key:nnn
\__enumext_def_meta_key:nnn
\__enumext_def_meta_key:Vnn
5290 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5291 {
5292   {enumext,1} = level-1,
5293   {enumext,2} = level-2,
5294   {enumext,3} = level-3,
5295   {enumext,4} = level-4,
5296   {enumext*} = enumext*
5297 }

```

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

```

5298 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5299 {
5300   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5301   { \msg_error:nn { enumext } { prohibited-unknown } }
5302   {
5303     \bool_if:nTF {#1}
5304     {
5305       \int_step_inline:nn { 4 }
5306       { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5307       \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5308     }
5309     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5310   }
5311 }

```

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

```

5312 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5313 {
5314   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5315   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5316   \prop_get:NVNTF
5317   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5318   { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5319   {
5320     \msg_error:nnn { enumext } { unknown-set } {#1}
5321     \use_none:nn
5322   }
5323 }
5324 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5325 {
5326   \bool_lazy_or:nTF

```

```

5327 { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5328 { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5329 { \msg_error:nnn { enumext } { already-defined } {#2} }
5330 {
5331   \keys_define:nn { enumext / #1 }
5332   {
5333     #2 .meta:n = {#3},
5334     #2 .value_forbidden:n = true
5335   }
5336 }
5337 }
5338 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

13.50 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function](#).

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.

```

\foreachkeyans
\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
\__enumext_foreach_keyans:nn
\__enumext_foreach_add_body:n

5339 \keys_define:nn { enumext / foreach }
5340 {
5341   before .tl_set:N = \l__enumext_foreach_before_tl,
5342   before .value_required:n = true,
5343   after .tl_set:N = \l__enumext_foreach_after_tl,
5344   after .value_required:n = true,
5345   start .int_set:N = \l__enumext_foreach_start_int,
5346   start .value_required:n = true,
5347   stop .int_set:N = \l__enumext_foreach_stop_int,
5348   stop .value_required:n = true,
5349   step .int_set:N = \l__enumext_foreach_step_int,
5350   step .value_required:n = true,
5351   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5352   wrapper .value_required:n = true,
5353   sep .tl_set:N = \l__enumext_foreach_sep_tl,
5354   sep .value_required:n = true,
5355   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5356 }
5357 \keys_precompile:nnN { enumext / foreach }
5358 {
5359   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5360 }
5361 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown *⟨keys⟩*.

```

5362 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5363 {
5364   \tl_if_blank:nTF {#2}
5365   {
5366     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5367   }
5368   {
5369     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5370   }
5371 }
5372 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5373 {
5374   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5375 }

```

We create the command.

```

5376 \NewDocumentCommand \foreachkeyans { +0{ } m }
5377 {
5378   \__enumext_foreach_keyans:nn {#1} {#2}
5379 }

```

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

```

5380 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5381 {

```

```

5382 \tl_use:N \g__enumext_foreach_default_keys_tl
5383 \keys_set:nn { enumext / foreach } {#1}
5384 \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5385 \prop_if_exist:cF { g__enumext_#2_prop }
5386 {
5387   \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5388 }
5389 \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5390 {
5391   \int_set:Nn \l__enumext_foreach_stop_int
5392     { \prop_count:c { g__enumext_#2_prop } }
5393 }
5394 \seq_clear:N \l__enumext_foreach_print_seq
5395 \int_step_function:nnnN
5396 { \l__enumext_foreach_start_int }
5397 { \l__enumext_foreach_step_int }
5398 { \l__enumext_foreach_stop_int }
5399 \__enumext_foreach_add_body:n
5400 \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5401 }
5402 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5403 {
5404   \seq_put_right:Ne \l__enumext_foreach_print_seq
5405   {
5406     \exp_not:V \l__enumext_foreach_before_tl
5407     \__enumext_foreach_wrapper:n
5408     {
5409       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5410     }
5411     \exp_not:V \l__enumext_foreach_after_tl
5412   }
5413 }

```

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

13.51 Messages

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

```

5414 \msg_new:nnn { enumext } { package-load }
5415 {
5416   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5417 }
5418 \msg_new:nnn { enumext } { package-not-load }
5419 {
5420   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5421 }
5422 \msg_new:nnn { enumext } { package-load-foot }
5423 {
5424   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5425 }

```

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

```

5426 \msg_new:nnn { enumext } { counters }
5427 {
5428   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5429   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5430 }

```

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

```

5431 \msg_new:nnn { enumext } { unknown-choice }
5432 {
5433   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5434 }

```

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

```

5435 \msg_new:nnnn { enumext } { anskey-env-error }
5436 {
5437   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
5438   'enumext' ~ package, ~ It~ is~ already~ defined.
5439 }
5440 {
5441   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5442   for ~ the ~ 'save-ans' ~ key.\\
5443 }

```

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

```

5444 \msg_new:nnn { enumext } { store-prop }
5445 {
5446   * ~ Package ~ enumext: ~ Creating ~
5447   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5448 }
5449 \msg_new:nnn { enumext } { store-seq }
5450 {
5451   * ~ Package ~ enumext: ~ Creating ~
5452   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5453 }
5454 \msg_new:nnn { enumext } { store-int }
5455 {
5456   * ~ Package ~ enumext: ~ Creating ~
5457   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5458 }
5459 \msg_new:nnn { enumext } { prop-seq-int-hook }
5460 {
5461   * ~ Package ~ enumext: ~ Elements ~ in ~
5462   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5463   * ~ Package ~ enumext: ~ Elements ~ in ~
5464   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5465   * ~ Package ~ enumext: ~ Value ~ off ~
5466   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5467 }
5468 \msg_new:nnn { enumext } { item-answer-hook }
5469 {
5470   * ~ Package ~ enumext: ~ Value ~ off ~
5471   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5472   * ~ Package ~ enumext: ~ Value ~ off ~
5473   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5474   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5475 }

```

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

```

5476 \msg_new:nnn { enumext } { invalid-key }
5477 {
5478   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5479 }
5480 \msg_new:nnn { enumext } { unknown-key-family }
5481 {
5482   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5483 }

```

Messages used in length calculation.

```

5484 \msg_new:nnn { enumext } { width-negative }
5485 {
5486   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5487   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5488 }
5489 \msg_new:nnn { enumext } { width-zero }
5490 {
5491   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5492   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5493 }

```

Messages used by show-length key in enumext.

```

5494 \msg_new:nnn { enumext } { list-lengths }
5495 {
5496   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5497   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5498   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5499   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5500   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5501   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5502   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5503   \__enumext_show_length:nnn { skip } { topsep } {#1}
5504   \__enumext_show_length:nnn { skip } { parsep } {#1}
5505   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5506   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5507   ****
5508 }

```

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

```

5509 \msg_new:nnn { enumext } { list-lengths-not-nested }
5510 {
5511     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5512     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5513     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5514     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5515     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5516     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5517     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5518     \__enumext_show_length:nnn { skip } { topsep } {#1}
5519     \__enumext_show_length:nnn { skip } { parsep } {#1}
5520     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5521     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5522     *****
5523 }
```

Messages used by `ref` key.

```

5524 \msg_new:nnn { enumext } { key-ref-empty }
5525 {
5526     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5527 }
```

Messages used by `save-ans` key.

```

5528 \msg_new:nnn { enumext } { save-ans-empty }
5529 {
5530     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5531 }
5532 \msg_new:nnn { enumext } { save-ans-log }
5533 {
5534     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5535 }
5536 \msg_new:nnn { enumext } { save-ans-log-hook }
5537 {
5538     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5539 }
5540 \msg_new:nnn { enumext } { save-ans-hook }
5541 {
5542     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5543 }
```

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

```

5544 \msg_new:nnn { enumext } { need-save-ans }
5545 {
5546     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5547 }
5548 \msg_new:nnn { enumext } { items-same-answer }
5549 {
5550     *****\\
5551     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5552     for ~ \c_left_brace_str #2 \c_right_brace_str\\
5553     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5554     'OK', ~ all ~ items ~ with ~ answer.\\
5555     *****
5556 }
5557 \msg_new:nnn { enumext } { item-greater-answer }
5558 {
5559     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5560     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5561     Items ~ > ~ Answers.
5562 }
5563 \msg_new:nnn { enumext } { item-less-answer }
5564 {
5565     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5566     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5567     Items ~ < ~ Answers.
5568 }
```

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

```

5569 \msg_new:nnn { enumext } { missing-starred }
5570 {
5571     Missing ~ '\c_backslash_str #1*' ~ #2.
5572 }
```

```

5573 \msg_new:nnn { enumext } { many-starred }
5574 {
5575   Many ~ '\c_backslash_str #1*' ~ #2.
5576 }

```

Messages used by `\printkeyans*` command.

```

5577 \msg_new:nnn { enumext } { print-starred }
5578 {
5579   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5580   #2 ~ environment ~ \msg_line_context:.
5581 }

```

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

```

5582 \msg_new:nnn { enumext } { list-too-deep }
5583 {
5584   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5585   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5586 }

```

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

```

5587 \msg_new:nnn { enumext } { anskey-unnumber-item }
5588 {
5589   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5590 }
5591 \msg_new:nnn { enumext } { anskey-already-stored }
5592 {
5593   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5594 }
5595 \msg_new:nnn { enumext } { anskey-empty-arg }
5596 {
5597   Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5598 }
5599 \msg_new:nnn { enumext } { anskey-wrong-place }
5600 {
5601   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5602   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5603 }
5604 \msg_new:nnn { enumext } { anskey-nested }
5605 {
5606   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5607 }
5608 \msg_new:nnn { enumext } { anskey-math-mode }
5609 {
5610   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5611 }
5612 \msg_new:nnn { enumext } { anskey-env-wrong }
5613 {
5614   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5615 }
5616 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5617 {
5618   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5619   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5620 }
5621 \msg_new:nnn { enumext } { command-wrong-place }
5622 {
5623   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5624   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5625 }
5626 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5627 {
5628   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5629   'anskey*' ~ and ~ is ~ being ~ ignored.
5630 }
5631 {
5632   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5633   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5634 }
5635 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5636 {
5637   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5638   'anskey*' ~ and ~ is ~ being ~ ignored.

```

```

5639     }
5640     {
5641         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5642         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5643     }
5644 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5645 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5646 {
5647     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5648     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5649 }
5650 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5651 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5652 {
5653     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5654     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5655 }

```

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

```

5656 \msg_new:nnn { enumext } { keyans-nested }
5657 {
5658     The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5659 }
5660 \msg_new:nnn { enumext } { keyans-wrong-level }
5661 {
5662     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5663     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5664 }
5665 \msg_new:nnn { enumext } { wrong-place }
5666 {
5667     Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5668     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5669 }
5670 \msg_new:nnn { enumext } { keyanspic-nested }
5671 {
5672     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5673 }
5674 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5675 {
5676     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5677     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5678 }
5679 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5680 {
5681     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5682 }
5683 \msg_new:nnnn { enumext } { keyans-unknown-key }
5684 {
5685     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5686     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5687 }
5688 {
5689     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5690     ~ have ~ a ~ key ~ called ~'#1'.\\
5691     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5692 }
5693 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5694 {
5695     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5696     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5697 }
5698 {
5699     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5700     ~ have ~ a ~ key ~ called ~'#1'.\\
5701     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5702 }

```

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

```

5703 \msg_new:nnnn { enumext } { starred-unknown-key }
5704 {
5705     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5706     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.

```



```

5707 }
5708 {
5709     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5710     ~ have ~ a ~ key ~ called ~ '#1'.\\
5711     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5712 }
5713 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5714 {
5715     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5716     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5717 }
5718 {
5719     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5720     ~ have ~ a ~ key ~ called ~ '#1'.\\
5721     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5722 }

```

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

```

5723 \msg_new:nnnn { enumext } { standar-unknown-key }
5724 {
5725     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5726     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5727 }
5728 {
5729     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5730     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5731     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5732 }
5733 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5734 {
5735     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5736     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5737 }
5738 {
5739     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5740     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5741     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5742 }

```

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

```

5743 \msg_new:nnnn { enumext } { for-key-unknown }
5744 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5745 {
5746     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5747     Check~that~you~have~spelled~the~key~name~correctly.
5748 }
5749 \msg_new:nnnn { enumext } { for-key-value-unknown }
5750 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5751 {
5752     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5753     Check~that~you~have~spelled~the~key~name~correctly.
5754 }

```

Messages used by **\getkeyans** command.

```

5755 \msg_new:nnn { enumext } { undefined-storage-anskey }
5756 {
5757     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5758 }

```

Messages used by **\miniright** command.

```

5759 \msg_new:nnn { enumext } { missing-miniright }
5760 {
5761     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5762     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5763 }
5764 \msg_new:nnn { enumext } { wrong-miniright-place }
5765 {
5766     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5767     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5768 }
5769 \msg_new:nnn { enumext } { wrong-miniright-use }
5770 {
5771     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\

```

```

5772     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5773   }
5774   \msg_new:nnn { enumext } { wrong-miniright-starred }
5775   {
5776     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5777   }
5778   \msg_new:nnn { enumext } { many-miniright-used }
5779   {
5780     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5781   }

```

Messages used by `\setenumextmeta` command.

```

5782   \msg_new:nnn { enumext } { unknown-set }
5783   {
5784     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5785   }
5786   \msg_new:nnn { enumext } { already-defined }
5787   {
5788     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5789   }
5790   \msg_new:nnn { enumext } { prohibited-unknown }
5791   {
5792     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5793   }

```

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

```

5794   \msg_new:nnn { enumext } { nested }
5795   {
5796     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5797   }
5798   \msg_new:nnn { enumext } { nested-horizontal }
5799   {
5800     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \msg_line_context:.
5801   }
5802   \msg_new:nnn { enumext } { item-joined }
5803   {
5804     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5805   }
5806   \msg_new:nnn { enumext } { item-joined-columns }
5807   {
5808     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5809   }

```

13.52 Finish package

Finish package implementation.

```

5810   \file_input_stop:
5811   \</package>

```

14 Index of Implementation

The *italic* numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>*</code>	228
<code>\+</code>	220
<code>\-</code>	220
<code>\\</code>	236, 2873, 4240, 4243, 5428, 5437, 5442, 5462, 5464, 5471, 5473, 5486, 5491, 5496, 5511, 5550, 5552, 5554, 5559, 5560, 5565, 5566, 5584, 5601, 5618, 5623, 5632, 5641, 5647, 5653, 5662, 5667, 5676, 5690, 5700, 5710, 5720, 5730, 5740, 5746, 5752, 5761, 5766, 5771
A	
<code>above</code>	<u>1687</u>
<code>above*</code>	<u>1687</u>
<code>\addvspace</code>	1257, 1285, 1328, 1331, 1499, 1502, 1599, 1605, 1640, 1646, 1667, 1673, 3698, 3859, 3877, 4131, 4135, 4488, 4503, 4549, 4563
<code>after</code>	<u>1087</u>
<code>align</code>	<u>640</u>
<code>\Alph</code>	40, 45
<code>\Alph</code>	582, 710, 754, 820, 5136
<code>\alph</code>	40, 45
<code>\alph</code>	583, 708, 5128
<code>\anskey</code>	13, 78, 80, <u>2691</u>
<code>anskey*</code>	14, <u>2801</u>
<code>\anspic</code>	16, 107, 110, <u>4145</u>
<code>\anspic*</code>	72
<code>\arabic</code>	32, 40
<code>\arabic</code>	581, 707, 753, 5120, 5124, 5140
B	
<code>base-fix</code>	<u>949</u>
<code>\baselineskip</code>	54
<code>\baselineskip</code>	965, 972
<code>before</code>	<u>1087</u>
<code>before*</code>	<u>1087</u>
<code>below</code>	<u>1687</u>
<code>below*</code>	<u>1687</u>
bool commands:	
<code>\bool_gset_false:N</code>	357, 358, 359, 2977, 2979, 4505, 4509, 4565
<code>\bool_gset_true:N</code>	265, 275, 1190, 2180, 2186, 4474, 4506, 4538, 4566
<code>\bool_if:NTF</code>	408, 418, 435, 478, 485, 494, 501, 510, 517, 526, 533, 1621, 1709, 1723, 1736, 1747, 1758, 1769, 1780, 1791, 1840, 1857, 1862, 1870, 1897, 1935, 1940, 1947, 1951, 1973, 1978, 1986, 1993, 2024, 2032, 2125, 2323, 2333, 2412, 2436, 2443, 2467, 2565, 2587, 2627, 2641, 2645, 2695, 2714, 2738, 2792, 2803, 2892, 2929, 2993, 3028, 3043, 3118, 3129, 3133, 3152, 3165, 3207, 3241, 3277, 3292, 3311, 3447, 3462, 3477, 3539, 3549, 3582, 3587, 3653, 3679, 3729, 3787, 3842, 3867, 4067, 4129, 4147, 4166, 4211, 4238, 4467, 4483, 4489, 4532, 4546, 4550, 4639, 4649, 4737, 4743, 4750, 4766, 4860, 4870, 4981, 4985, 5011, 5018, 5046
<code>\bool_if:nTF</code>	1647, 1674, 3263, 3428, 3497, 4187, 5159, 5303
<code>\bool_if_p:N</code>	284, 299, 959, 960, 968, 969, 2004, 2005, 2013, 2014, 2138, 2164, 2177, 2178, 2183, 2184, 2500, 2510, 2522, 2537, 2538, 2572, 2613, 2614, 2915, 3105, 3106, 3143, 3144, 3626, 3628, 3639, 4194, 4195
<code>\bool_lazy_all:nTF</code>	282, 297, 957, 2136, 2162, 2498, 2507, 2520, 2535, 3624, 3637
<code>\bool_lazy_and:nnTF</code>	261, 271, 967, 1614, 2003, 2012, 2176, 2182, 2571, 2578, 2612, 2756, 2768, 2914, 2920, 3104
<code>\bool_lazy_or:nnTF</code>	2065, 2072, 3142, 4193, 5326
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 97, 102, 103, 108, 109, 112, 131, 138, 139, 146, 153, 154, 159, 161, 162, 176, 188, 190
<code>\bool_not_p:n</code>	262, 272, 961, 2509, 2573, 2579, 2916, 2921, 3627, 3640
<code>\bool_set_eq:NN</code>	3216, 3407, 4690, 4934
<code>\bool_set_false:N</code>	415, 979, 2110, 2111, 2143, 2148, 2152, 2156, 2169, 2856, 3601, 3746, 3795, 3882, 4020, 4064, 4608, 4634, 4687, 4876, 4931, 5172, 5173
<code>\bool_set_true:N</code>	289, 290, 304, 305, 400, 403, 633, 994, 1693, 1698, 1960, 2082, 2083, 2355, 2363, 2857, 3210, 3212, 3244, 3246, 3403, 3415, 3562, 3600, 3633, 3646, 3719, 3792, 3819, 4017, 4456, 4521, 4607, 4694, 4701, 4702, 4746, 4874, 4938, 4945, 4946, 5167, 5168
box commands:	
<code>\box_dp:N</code>	1545, 1546, 1549, 1556, 1569, 1577, 1583, 1591, 4078, 4083, 4131, 4222
<code>\box_ht:N</code>	1328, 1331, 1342, 1343, 1354, 1356, 1371, 1374, 1382, 1383, 1394, 1396, 1411, 1414, 1421, 1422, 1433, 1435, 1450, 1453, 1499, 1502, 1510, 1511, 1519, 1520, 1532, 1534
<code>\box_ht_plus_dp:N</code>	4073, 4139, 4175
<code>\box_new:N</code>	70, 149, 150, 183, 189
<code>\box_use_drop:N</code>	4500, 4561, 4802, 5058
<code>\box_wd:N</code>	589
C	
<code>\c</code>	228, 229, 856, 858, 870, 872
<code>\catcode</code>	2873
<code>\cB</code>	229
<code>\cE</code>	229
<code>\centering</code>	1649, 1676, 4266, 4493, 4554
<code>check-ans</code>	<u>2102</u>
Document class:	
<code>article</code>	46
clist commands:	
<code>\clist_const:Nn</code>	195
<code>\clist_map_function:nN</code>	4249
<code>\clist_map_inline:Nn</code>	639, 904, 1086, 1101, 1182, 1703
<code>\clist_map_inline:nn</code>	49, 60, 78, 86, 99, 111, 141, 170, 194, 617, 670, 690, 999, 1020, 1196, 1809, 2049, 2116, 2302, 2320, 2352, 2495, 3037, 3332, 3344, 3384, 3526, 3529, 3557, 3569, 3572, 3592, 5279
<code>\columnbreak</code>	79
<code>\columnbreak</code>	2575
<code>columns</code>	<u>1166</u>
<code>columns-sep</code>	<u>1166</u>
<code>\columnsep</code>	100
<code>\columnsep</code>	3674, 3840
<code>\columnseprule</code>	100
<code>\columnseprule</code>	3677, 3841
Commands provide by enumext :	
<code>\anskey</code>	30, 69, 70, 74–78, 80, 81, 87, 89, 100, 119, 128,

129, 137	
\anspic*	30, 31, 72, 75, 88, 109, 110, 128, 129
\anspic	30, 76, 107, 109, 110, 137
\foreachkeyans	133, 139
\getkeyans	75, 128, 139
\item* 30, 31, 72, 75, 76, 88, 91, 95, 120, 121, 126, 128, 129	
\item	91, 95, 114, 119, 120, 122, 125
\miniright	29, 51, 59, 60, 102, 139
\printkeyans*	129
\printkeyans	30, 76, 129
\setenumextmeta	132, 140
\setenumext	30, 129, 131, 132, 135
Counters defined by enumext :	
enumXiii	28, 39
enumXii	28, 39
enumXiv	28, 39
enumXi	28, 39
enumXviii	28, 39
enumXvii	28, 39, 121
enumXvi	28, 39
enumXv	28, 39
cs commands:	
\cs_generate_variant:Nn	200, 201, 591, 607, 862, 878, 2404, 2409, 2485, 2809, 3516, 4251, 5338
\cs_if_exist:NTF	561
\cs_if_free:NTF	2760, 2772
\cs_new:Nn	214
\cs_new:Npn	232, 1810, 1819, 1827, 2367, 2376, 2384, 5187, 5196, 5205
\cs_new_eq:NN	384, 385, 390, 391, 420, 421, 424, 425
\cs_new_protected:Nn	224, 238, 254, 280, 313, 343, 349, 355, 361, 367, 375, 395, 443, 447, 466, 476, 492, 508, 524, 540, 730, 791, 842, 955, 1102, 1106, 1110, 1114, 1118, 1122, 1126, 1130, 1134, 1138, 1142, 1146, 1150, 1154, 1158, 1162, 1197, 1209, 1242, 1259, 1270, 1287, 1313, 1334, 1459, 1485, 1505, 1538, 1560, 1595, 1601, 1704, 1718, 1732, 1743, 1754, 1765, 1776, 1787, 1868, 1971, 1984, 2001, 2022, 2050, 2055, 2080, 2121, 2131, 2174, 2189, 2196, 2205, 2210, 2215, 2220, 2229, 2234, 2239, 2410, 2434, 2441, 2465, 2472, 2486, 2712, 2731, 2747, 2810, 2846, 2877, 2912, 2954, 2975, 2983, 3026, 3041, 3069, 3102, 3138, 3150, 3163, 3249, 3259, 3270, 3286, 3302, 3424, 3440, 3456, 3470, 3593, 3622, 3651, 3658, 3688, 3705, 3727, 3749, 3785, 3809, 3826, 3851, 3865, 3886, 4039, 4233, 4247, 4252, 4276, 4286, 4317, 4446, 4465, 4511, 4530, 4594, 4621, 4628, 4637, 4647, 4672, 4814, 4858, 4889, 4895, 4916, 4973, 5070
\cs_new_protected:Npn	202, 206, 210, 428, 559, 576, 586, 592, 711, 755, 825, 849, 863, 1631, 1660, 1836, 1855, 1925, 1958, 2060, 2244, 2321, 2331, 2353, 2361, 2396, 2405, 2561, 2624, 2639, 2677, 2681, 2801, 2832, 2836, 2867, 3003, 3079, 3123, 3203, 3222, 3345, 3349, 3363, 3367, 3385, 3389, 3399, 3411, 3485, 3519, 3560, 3604, 3805, 4048, 4055, 4062, 4164, 4183, 4207, 4348, 4397, 4611, 4678, 4685, 4699, 4707, 4712, 4722, 4882, 4922, 4929, 4943, 4951, 4968, 5092, 5105, 5153, 5276, 5288, 5312, 5324, 5362, 5372, 5380, 5402
\cs_new_protected_nopar:Nn	3947, 3989, 3997, 4005, 4657, 4665, 4796, 4901, 4909, 5052
\cs_new_protected_nopar:Npn	3939, 3955, 4728, 4774, 5001, 5026
\cs_set:Npn	2496, 2533, 5098
\cs_set_eq:NN	4584, 4585, 4776, 4847, 4848, 5028
\cs_set_protected:Nn	1025, 1041, 1054, 1066
\cs_set_protected:Npn	45, 54, 71, 79, 94, 100, 134, 166, 174, 608, 618, 640, 675, 691, 737, 879, 905, 981, 1004, 1078, 1087, 1166, 1183, 1687, 1798, 2041, 2102, 2261, 2303, 2339, 2488, 3030, 3321, 3337, 3377, 3517, 3558
\cs_to_str:N	578, 601
\cs_undefine:N	2749, 2750, 2751, 2752
D	
\d	220
\DeclareDocumentEnvironment	544
dim commands:	
\dim_abs:n	3490, 3495
\dim_add:Nn	4082, 4311, 4342
\dim_compare:nNnTF	1027, 1043, 1056, 1068, 1346, 1358, 1386, 1398, 1425, 1437, 1514, 1522, 1633, 1662, 3487, 3492, 3498, 3504, 3506, 3508, 3663, 3710, 3813, 3830, 4057, 4288, 4304, 4319, 4335, 4448, 4513
\dim_compare:nTF	2597, 2942, 3752, 3889
\dim_eval:n	965, 4137, 4218
\dim_gset_eq:NN	4457, 4522
\dim_gzero:N	2981, 4508, 4568
\dim_new:N	67, 74, 75, 76, 96, 143, 151, 152, 182, 184, 185, 191
\dim_set:Nn	589, 995, 3239, 3490, 3495, 3497, 3500, 3501, 3505, 3507, 3510, 3511, 3513, 3666, 3713, 3751, 3815, 3832, 3888, 4071, 4173, 4254, 4290, 4297, 4321, 4328, 4383, 4432, 4450, 4515, 4724
\dim_set_eq:NN	698, 744, 813, 817, 3154, 3155, 3167, 3168, 3234, 3528, 3571, 3674, 3840, 4390, 4393, 4394, 4439, 4442, 4443, 4717, 4788, 5040
\dim_sub:Nn	3757, 3894, 4306, 4337
\dim_use:N	1028, 1036, 1634, 1644, 2475, 2478, 2483, 3254, 3256, 3307, 3664, 3668, 3669, 3671, 3711, 3716, 3717, 3723, 3754, 3759
\dim_zero:N	3563, 3677, 3841, 4084
\dim_zero_new:N	558
\c_zero_dim	1030, 1044, 1057, 1069, 1634, 1662, 2599, 2944, 3487, 3492, 3498, 3505, 3664, 3711, 3754, 3813, 3830, 3891, 4057, 4288, 4304, 4319, 4335, 4448, 4513
\dimeval	2268
E	
\end	2438, 2469, 3695, 3856, 4121, 4268, 5161, 5171, 5179
end internal commands:	
\end__enumext_mini_page	1642, 1669, 3738, 3876, 4472, 4536, 4562
\endgroup	2873
\endlist	385
\endminipage	391
enumext	5, 3763
enumext internal commands:	
\l__enumext__ref_the_count_tl	42
\l__enumext__resume_name_tl	65
__enumext_add_meta_key:nnn	132, 5290 , 5306, 5307, 5309, 5312
__enumext_add_pre_parsep:	52, 1207, 1209 , 1209
__enumext_after_args_exec:	50, 1102 , 1114, 3776
__enumext_after_args_exec_v:	1118 , 1130, 3909
__enumext_after_args_exec_vii:	1134 , 1158
__enumext_after_args_exec_viii:	1162
__enumext_after_env:nn	84, 85, 87, 103, 115, 123, 206 , 206, 2887, 3781, 4481, 4544, 4830
__enumext_after_hyperref:	36, 393, 395 , 395
\l__enumext_after_list_args_v_tl	1132
\l__enumext_after_list_args_vii_tl	1160, 4794

`\l__enumext_after_list_args_viii_tl` .. 1164, 5050
`__enumext_after_list_vii:` 116, 119, 4592, 4628, 4628
`__enumext_after_list_viii:` .. 124, 4856, 4895, 4895
`__enumext_after_stop_list:` 50, 102, 1102, 1110, 3743
`__enumext_after_stop_list_v:` 1118, 1126, 3883
`\l__enumext_after_stop_list_v_tl` 1128
`__enumext_after_stop_list_vii:` .. 119, 1134, 1150, 4631
`\l__enumext_after_stop_list_vii_tl` ... 1152
`__enumext_after_stop_list_viii:` . 1154, 4898
`\l__enumext_after_stop_list_viii_tl` ... 1156
`\l__enumext_align_label_pos_v_str` 3474
`\l__enumext_align_label_pos_X_str` 79
`\l__enumext_align_label_vii_str` 4763
`\l__enumext_align_label_viii_str` 5015
`\l__enumext_align_label_X_str` 174
`\c__enumext_all_envs_clist` . 195, 639, 904, 1086, 1101, 1182, 1703
`\c__enumext_all_families_seq` .. 131, 5244, 5270
`\l__enumext_anskey_env_bool` 33, 83, 34, 290, 305, 2803
`__enumext_anskey_env_clean_vars:` . 86, 2908, 2912, 2975
`__enumext_anskey_env_define_keys:` 83, 2801, 2810, 2881
`__enumext_anskey_env_exec:` 85, 2806, 2877, 2877
`__enumext_anskey_env_make:n` 69, 83, 2085, 2801, 2801, 2809
`__enumext_anskey_env_reset_keys:` 84, 85, 2846, 2909
`__enumext_anskey_env_reset_keys:__-enumext_rescan_anskey_env:n` 2801
`__enumext_anskey_env_save_keys:` .. 85, 2889, 2912, 2912
`__enumext_anskey_env_store:` .. 86, 2905, 2912, 2954
`__enumext_anskey_env_unknown:n` 84, 2829, 2832
`__enumext_anskey_env_unknown:nn` . 2834, 2836
`\l__enumext_anskey_level_int` .. 28, 2733, 2734
`__enumext_anskey_safe_inner:`.. 82, 2706, 2712, 2731
`__enumext_anskey_safe_inner:n` 81
`__enumext_anskey_safe_outer:` . 81, 2693, 2712, 2712
`__enumext_anskey_show_wrap_arg:n` . 80, 2624, 2624, 2643, 2658
`__enumext_anskey_show_wrap_left:n` 80, 2569, 2639, 2639
`__enumext_anskey_unknown:n` 81, 2661, 2675, 2677
`__enumext_anskey_unknown:nn` . 2661, 2679, 2681
`__enumext_anskey_wrapper:n` 2265, 2637
`\l__enumext_anspic_above_int` . 142, 4255, 4256, 4258
`__enumext_anspic_args:nnn` 110, 111, 4145, 4161, 4233
`\l__enumext_anspic_args_seq` 109–112, 142, 4159, 4267, 4280
`\l__enumext_anspic_below_int` . 142, 4255, 4256, 4259
`\l__enumext_anspic_body_box` ... 142, 4172, 4175
`__enumext_anspic_body_dim:n` .. 110, 4145, 4164, 4210
`\l__enumext_anspic_body_htdp_dim` .. 110, 142, 4173, 4221
`__enumext_anspic_exec:` 4145
`__enumext_anspic_exec:` 112, 4116, 4276
`__enumext_anspic_label:nn` 111, 4145, 4183, 4213, 4228
`\l__enumext_anspic_label_above_bool` ... 142, 4017, 4020, 4067, 4129, 4166, 4211, 4238
`\l__enumext_anspic_label_box` .. 142, 4070, 4073
`\l__enumext_anspic_label_htdp_dim` . 108, 142, 4071, 4077, 4139, 4220
`__enumext_anspic_label_pos:nnn` .. 111, 4145, 4207, 4236
`\l__enumext_anspic_label_sep_skip` 4027, 4079, 4140, 4223, 4240
`\l__enumext_anspic_layout_style_tl` 4029, 4278, 4283
`\l__enumext_anspic_mini_pos_str` .. 142, 4018, 4021, 4265
`\l__enumext_anspic_mini_width_dim` 142, 4185, 4254, 4265
`__enumext_anspic_print:n` 111, 112, 4145, 4247, 4251, 4280, 4283
`__enumext_anspic_row:n` .. 112, 4145, 4249, 4252
`__enumext_anspic_start_list_tag:` 3963, 3989, 4235
`__enumext_anspic_stop_list_tag:` . 3963, 4005, 4245
`__enumext_anspic_stop_start_list_tag:` 3963, 3997, 4237
`__enumext_at_begin_document:n` .. 35, 202, 202, 382, 388
`\l__enumext_base_line_fix_bool` 47, 129, 130, 951, 960, 979, 5167, 5172
`__enumext_before_args_exec:` 50, 101, 118, 1102, 1102, 3708
`__enumext_before_args_exec_v:` 1118, 1118, 3812
`__enumext_before_args_exec_vii:` . 1134, 1134, 4625
`__enumext_before_args_exec_viii:` 1138, 4892
`__enumext_before_env:nn` 83, 206, 210, 2754, 2766, 2778, 2879
`__enumext_before_keys_exec:` .. 50, 1102, 1106, 3773
`__enumext_before_keys_exec_v:` 1118, 1122, 3906
`__enumext_before_keys_exec_vii` 1134
`__enumext_before_keys_exec_vii:` . 1142, 4579
`__enumext_before_keys_exec_viii:` 1146, 4842
`__enumext_before_list:` .. 101, 3705, 3705, 3767
`__enumext_before_list_v:` ... 3809, 3809, 3901
`__enumext_before_list_vii:` ... 118, 4574, 4621, 4621
`__enumext_before_list_viii:` . 124, 4838, 4889, 4889
`\l__enumext_before_no_starred_key_v_tl` 1124
`\l__enumext_before_no_starred_key_vii_-tl` 1144
`\l__enumext_before_no_starred_key_viii_-tl` 1148
`\l__enumext_before_starred_key_v_tl` ... 1120
`\l__enumext_before_starred_key_vii_tl` . 1136
`\l__enumext_before_starred_key_viii_tl` 1140

```

\__enumext_calc_hspace:NNNNNNN 97, 3485, 3485,
3516, 3521, 3564
\__enumext_check_ans_active: 70, 101, 118, 2121,
2121, 3709, 4624
\g__enumext_check_ans_item_tl . . . . . 89
\g__enumext_check_ans_key_bool 71, 72, 153, 357,
2180, 2186, 2993
\l__enumext_check_ans_key_bool 71, 2106, 2111,
2177, 2183
\__enumext_check_ans_key_hook: . . 71, 102, 119,
2174, 2174, 3744, 4632
\__enumext_check_ans_level: 70, 2121, 2127, 2131
\__enumext_check_ans_log: 71, 72, 87, 2220, 2220,
2997
\__enumext_check_ans_log_msg_greater: 2220,
2226, 2239
\__enumext_check_ans_log_msg_less: 2220, 2224,
2229
\__enumext_check_ans_log_msg_same_ok: 2220,
2225, 2234
\__enumext_check_ans_msg_greater: 2196, 2202,
2215
\__enumext_check_ans_msg_less: 2196, 2200, 2205
\__enumext_check_ans_msg_same_ok: 2196, 2201,
2210
\__enumext_check_ans_show: . . 71, 86, 2196, 2196,
2995
\l__enumext_check_answers_bool . 69, 70, 81, 91,
153, 2083, 2110, 2125, 2412, 2436, 2443, 2467, 2695,
2892, 3118, 3207, 3241, 4743
\__enumext_check_starred_cmd:n 34, 72, 89, 123,
2244, 2244, 3912, 4127, 4855
\g__enumext_check_starred_cmd_int . . 95, 153,
2247, 2253, 2258, 3422, 4192, 4980
\l__enumext_check_start_line_env_tl . 34, 153,
320, 328, 336, 2250, 2256, 2259
\l__enumext_columns_sep_v_dim 3830, 3832, 3840
\l__enumext_columns_sep_vii_dim . . 4288, 4290,
4299, 4311, 4387, 4811
\l__enumext_columns_sep_viii_dim . 4319, 4321,
4330, 4342, 4436, 5067
\l__enumext_columns_v_int 1479, 1497, 1665, 3828,
3836, 3848, 3853
\l__enumext_columns_vii_int . . 4293, 4296, 4300,
4309, 4351, 4355, 4358, 4364, 4370, 4374, 4805, 4819
\l__enumext_columns_viii_int . 4324, 4327, 4331,
4340, 4400, 4404, 4407, 4413, 4419, 4423, 5061, 5076
\l__enumext_counter_i_tl . . . . . 45, 568
\l__enumext_counter_ii_tl . . . . . 45, 569
\l__enumext_counter_iii_tl . . . . . 45, 570
\l__enumext_counter_iv_tl . . . . . 45, 571
\c__enumext_counter_style_tl . . . . . 32, 50, 226
\g__enumext_counter_styles_tl . 28, 40, 67, 579,
597
\l__enumext_counter_v_tl . . . . . 45, 572, 833
\l__enumext_counter_vi_tl . . . . . 45, 573
\l__enumext_counter_vii_tl . . . . . 45, 574, 765
\l__enumext_counter_viii_tl . . . . . 45, 575, 781
\l__enumext_current_widest_dim 28, 67, 603, 699,
745, 814, 818
\__enumext_def_meta_key:nnn . . . 132, 5290, 5318,
5324, 5338
\__enumext_default_item:n . . . 3203, 3203, 3267
\__enumext_define_counters:Nn 28, 559, 559, 568,
569, 570, 571, 572, 573, 574, 575
\__enumext_endminipage: . 36, 382, 391, 553, 4502,
4798, 5054
\g__enumext_envir_name_tl 33, 34, 291, 306, 365,
2053, 2058, 2068, 2208, 2213, 2218, 2232, 2237, 2242
\l__enumext_envir_name_tl . 33, 34, 34, 260, 270,
319, 327, 335, 5686, 5689, 5696, 5699, 5706, 5709,
5716, 5719, 5725, 5729, 5735, 5739, 5796, 5800
\__enumext_execute_after_env: 35, 68, 71, 72, 82,
86, 2983, 2983, 3783, 4832
\__enumext_fake_item_indent: . 1025, 1025, 3548
\l__enumext_fake_item_indent_v_dim 1044, 1049
\l__enumext_fake_item_indent_v_tl 1046, 3404,
3408, 3416
\__enumext_fake_item_indent_vii: . 1025, 1054,
3581
\l__enumext_fake_item_indent_vii_dim . 1057,
1061
\l__enumext_fake_item_indent_vii_tl . . 1059,
4793
\__enumext_fake_item_indent_viii: 1025, 1066,
3586
\l__enumext_fake_item_indent_viii_dim 1069,
1073
\l__enumext_fake_item_indent_viii_tl . 1071,
5045
\l__enumext_fake_item_indent_X_tl . . . . 100
\__enumext_fake_make_label_vii:n . 121, 4728,
4728, 4785
\__enumext_fake_make_label_viii:n 5001, 5001,
5037
\__enumext_filter_first_level:n . . 130, 5187,
5187, 5221, 5232
\__enumext_filter_first_level_key:n 130, 5187,
5192, 5196
\__enumext_filter_first_level_pair:nn . 130,
5187, 5193, 5205
\__enumext_filter_save_key:n . . 75, 2328, 2336,
2359, 2365, 2367, 5118, 5122, 5126, 5130, 5134,
5138
\__enumext_filter_save_key_key:n . . 75, 2367,
2372, 2376
\__enumext_filter_save_key_pair:nn 75, 2367,
2373, 2384
\__enumext_filter_series:n 63, 1810, 1810, 1848,
1860, 1865
\__enumext_filter_series_key:n 64, 1810, 1815,
1819
\__enumext_filter_series_pair:nn . . 64, 1810,
1816, 1827
\__enumext_first_item_tmp_vii: 117, 119, 4584,
4657, 4657
\__enumext_first_item_tmp_viii: . . 125, 4847,
4901, 4901
\g__enumext_footnote_arg_seq 171, 449, 462, 472
\g__enumext_footnote_int 171, 456, 459, 461, 463
\g__enumext_footnote_int_seq 171, 450, 463, 468,
471
\__enumext_footnotes_key_bool . . . . . 36
\l__enumext_footnotes_key_bool 31, 36, 161, 403,
408, 415, 485, 501, 517, 533
\__enumext_footnotetext:nn . . . . . 443, 443, 473
\__enumext_foreach_add_body:n . 133, 5339, 5399,
5402
\l__enumext_foreach_after_tl . . . . . 5343, 5411

```


`\l__enumext_foreach_before_tl` 5341, 5406
`\g__enumext_foreach_default_keys_tl` 133, 126, 5361, 5382
`__enumext_foreach_keyans:nn` . . 133, 5339, 5378, 5380
`\l__enumext_foreach_name_prop_tl` . 126, 5384, 5409
`\l__enumext_foreach_print_seq` 126, 5394, 5400, 5404
`\l__enumext_foreach_sep_tl` 5353, 5400
`\l__enumext_foreach_start_int` 5345, 5396
`\l__enumext_foreach_step_int` 5349, 5397
`\l__enumext_foreach_stop_int` . 5347, 5389, 5391, 5398
`__enumext_foreach_wrapper:n` 5351, 5407
`__enumext_getkeyans:nn` . . 128, 5087, 5101, 5105
`__enumext_getkeyans_aux:n` 128, 5087, 5089, 5092
`\l__enumext_hyperref_bool` 31, 36, 161, 400, 418, 435, 2614, 3106, 4737
`__enumext_hypertarget:nn` 36, 395, 420, 424, 440
`__enumext_if_is_int:n` 218
`__enumext_if_is_int:nTF` 218, 851, 865
`__enumext_internal_mini_page:` 38, 99, 118, 540, 540, 3596, 4597
`__enumext_is_not_nested:` 28, 33, 99, 118, 254, 254, 3595, 4596
`__enumext_is_on_first_level:` . 28, 33, 99, 118, 254, 280, 3602, 4609
`\g__enumext_item_anskey_int` 81, 89, 153, 352, 379, 380, 2193, 2563, 3120
`__enumext_item_answer_diff:` . . 71, 72, 86, 2189, 2189, 2990
`\g__enumext_item_answer_diff_int` . 71, 72, 153, 353, 2191, 2198, 2222
`\l__enumext_item_column_pos_vii_int` 119, 4358, 4364, 4370, 4374, 4381, 4668, 4805, 4808
`\l__enumext_item_column_pos_viii_int` . . 125, 4407, 4413, 4419, 4423, 4430, 4912, 5061, 5064
`\l__enumext_item_column_pos_X_int` 174
`\g__enumext_item_count_all_vii_int` 119, 4382, 4669, 4819, 4827
`\g__enumext_item_count_all_viii_int` 125, 4431, 4913, 5075, 5084
`\g__enumext_item_count_all_X_int` 174
`\g__enumext_item_number_bool` 153
`\l__enumext_item_number_bool` 70, 159, 2143, 2148, 2152, 2156, 2169, 2738, 2792, 3210, 3244, 4746
`\g__enumext_item_number_int` 70, 71, 153, 351, 378, 380, 2142, 2147, 2151, 2155, 2168, 2193, 3209, 3243, 4745
`__enumext_item_peek_args_vii:` 119, 120, 4665, 4670, 4672
`__enumext_item_peek_args_viii:` . . 125, 4909, 4914, 4916
`__enumext_item_star_exec:` 92, 3222, 3249, 3294, 3313
`\l__enumext_item_starred_vii_bool` 4687, 4701, 4750
`\l__enumext_item_starred_viii_bool` 4931, 4945, 5011, 5046
`\l__enumext_item_starred_X_bool` 174
`__enumext_item_std:w` . 36, 91, 95, 382, 386, 3213, 3219, 3247, 3404, 3408, 3416
`\g__enumext_item_symbol_aux_tl` . 91, 130, 3227, 3230, 3255, 3299, 3317
`\g__enumext_item_symbol_aux_vii_tl` 4709, 4752, 4755, 4759, 4761
`\g__enumext_item_symbol_aux_X_tl` 174
`\l__enumext_item_symbol_sep_vii_dim` . . 4717, 4724, 4758, 4760
`\l__enumext_item_symbol_vii_tl` 4755
`\l__enumext_item_text_vii_box` 4777, 4802
`\l__enumext_item_text_viii_box` 5029, 5058
`\l__enumext_item_text_X_box` 174
`\l__enumext_item_width_vii_dim` 4297, 4306, 4385, 4393, 4394
`\l__enumext_item_width_viii_dim` . . 4328, 4337, 4434, 4442, 4443
`\l__enumext_item_width_X_dim` 174
`\l__enumext_itemindent_X_dim` 71
`\l__enumext_itemsep_i_skip` 1340, 1347, 1350, 1352, 1359, 1363, 1366, 1368, 1508, 1515, 1517, 1518, 1523, 1527, 1529, 1530
`\l__enumext_itemsep_ii_skip` 1380, 1387, 1390, 1392, 1399, 1403, 1406, 1408
`\l__enumext_itemsep_iii_skip` 1419, 1426, 1429, 1431, 1438, 1442, 1445, 1447
`\l__enumext_itemsep_vii_skip` 4825
`\l__enumext_itemsep_viii_skip` 5082
`\l__enumext_joined_item_aux_vii_int` . . 4379, 4380, 4381, 4382, 4388
`\l__enumext_joined_item_aux_viii_int` . . 4428, 4429, 4430, 4431, 4437
`\l__enumext_joined_item_aux_X_int` 174
`__enumext_joined_item_vii:w` . . 120, 4665, 4675, 4676, 4678
`\l__enumext_joined_item_vii_int` . . 4350, 4351, 4354, 4356, 4362, 4367, 4372, 4377, 4379, 4385
`__enumext_joined_item_viii:w` . . 125, 4909, 4919, 4920, 4922
`\l__enumext_joined_item_viii_int` . . 4399, 4400, 4403, 4405, 4411, 4416, 4421, 4426, 4428, 4434
`\l__enumext_joined_item_X_int` 174
`\l__enumext_joined_width_vii_dim` . . 4383, 4390, 4393, 4779, 4787
`\l__enumext_joined_width_viii_dim` 4432, 4439, 4442, 5031, 5039
`\l__enumext_joined_width_X_dim` 174
`__enumext_keyans_addto_prop:n` 87, 3003, 3003, 3419, 4189
`__enumext_keyans_addto_seq:n` . . 88, 3079, 3079, 3421, 4191
`__enumext_keyans_addto_seq_link:` 3079, 3100, 3102, 4979
`__enumext_keyans_default_item:n` . . 95, 3399, 3399, 3436
`\l__enumext_keyans_env_bool` 34, 3627, 3640, 3792, 3882
`__enumext_keyans_fake_item_indent:` . . 1025, 1041, 3538
`\l__enumext_keyans_level_h_int` . . 123, 28, 774, 800, 2722, 2784, 3057, 4603, 4864, 4865
`\l__enumext_keyans_level_int` . . 28, 1625, 2718, 2780, 3052, 3791, 3796, 4155
`__enumext_keyans_make_label:` 41, 96, 3440, 3440, 3536
`__enumext_keyans_make_label_box:` 3440, 3444, 3449, 3470
`__enumext_keyans_make_label_std:` 3440, 3452, 3456


```

\__enumext_keyans_mini_right_cmd:n 60, 1627,
1660, 1660
\__enumext_keyans_mini_set_vskip: . . . . . 57
\__enumext_keyans_minipage_add_space: 1459,
1485, 3821
\__enumext_keyans_minipage_set_skip: . 1459,
1459, 1487
\__enumext_keyans_multi_addvspace: 1259, 1270,
3845
\__enumext_keyans_multi_set_vskip: 53, 1259,
1259, 1272
\__enumext_keyans_multicols_start: 3809, 3824,
3826
\__enumext_keyans_multicols_stop: 1664, 3809,
3851, 3880
\__enumext_keyans_name_and_start: 28, 34, 123,
313, 313, 3793, 4046, 4869
\__enumext_keyans_parse_keys:n 3805, 3805, 3900
\__enumext_keyans_pic_arg_two: 108, 4039, 4062,
4092
\l__enumext_keyans_pic_level_int .. 28, 1609,
2726, 2788, 3006, 3047, 3082, 3170, 4041, 4042
\__enumext_keyans_pic_parse_keys:n 4039, 4048,
4091
\g__enumext_keyans_pic_parsep_skip ... 142
\__enumext_keyans_pic_safe_exec: . 108, 4039,
4039, 4090
\__enumext_keyans_pic_skip_abs:N . 108, 4039,
4055, 4066
\__enumext_keyans_pre_itemsep_skip: .. 1459,
1478, 1505
\__enumext_keyans_redefine_item: .. 95, 3424,
3424, 3535
\__enumext_keyans_ref: . . . . . 45, 825, 842, 3537
\__enumext_keyans_ref:n . . . . . 44, 822, 825, 825
\__enumext_keyans_safe_exec: . 3785, 3785, 3899
\__enumext_keyans_set_item_width: 105, 3886,
3886, 3908
\__enumext_keyans_show_ans: .. 3123, 3131, 3150
\__enumext_keyans_show_item_opt: .. 95, 3123,
3138, 3417, 4204, 5048
\__enumext_keyans_show_left:n . 95, 3123, 3123,
3414, 4198
\__enumext_keyans_show_pos: .. 3123, 3135, 3163
\__enumext_keyans_starred_item:n .. 95, 3411,
3411, 3432
\__enumext_keyans_store_ref: .. 88, 3026, 3026,
3420, 4190, 4977
\__enumext_keyans_store_ref_aux_i: 88, 3026,
3038, 3041
\__enumext_keyans_store_ref_aux_ii: 88, 3026,
3067, 3069
\__enumext_keyans_unknown_keys:n . 3337, 3341,
3345, 4037
\__enumext_keyans_unknown_keys:nn 3337, 3347,
3349
\__enumext_keyans_wrapper_opt:n .. 2271, 3146
\l__enumext_label_copy_i_tl .. 2529, 3045, 3050,
3055, 3060
\l__enumext_label_copy_v_tl . . . . . 3055
\l__enumext_label_copy_vi_tl . . . . . 3050
\l__enumext_label_copy_vii_tl 2505, 2516, 2545,
3045
\l__enumext_label_copy_viii_tl . . . . . 3060
\l__enumext_label_copy_X_tl . . . . . 163
\l__enumext_label_fill_left_v_tl . . . . . 3460
\l__enumext_label_fill_left_X_tl . . . . . 100
\l__enumext_label_fill_right_v_tl . . . . 3467
\l__enumext_label_fill_right_X_tl . . . . 100
\l__enumext_label_font_style_v_tl 3461, 3476,
4202
\l__enumext_label_font_style_vii_tl ... 4765
\l__enumext_label_font_style_viii_tl .. 5017
\l__enumext_label_i_tl . . . . . 691
\l__enumext_label_ii_tl . . . . . 691
\l__enumext_label_iii_tl . . . . . 691
\l__enumext_label_iv_tl . . . . . 691
\__enumext_label_style:Nnn 28, 40, 592, 592, 607,
696, 742, 811, 815
\l__enumext_label_v_tl 88, 808, 3011, 3087, 3157,
3197, 3413, 3418, 3903, 4070, 4197, 4199
\l__enumext_label_vi_tl 88, 808, 3008, 3084, 4197,
4199, 4203
\l__enumext_label_vii_tl . 737, 4696, 4719, 4726
\l__enumext_label_viii_tl 737, 4940, 4971, 4975
\l__enumext_label_width_by_box .. 67, 588, 589
\__enumext_label_width_by_box:Nn 40, 586, 586,
591, 603, 875
\l__enumext_labelsep_i_dim ... 3155, 3160, 3168,
3200, 4983, 4998
\l__enumext_labelsep_v_dim . . . . . 3835
\l__enumext_labelsep_vii_dim . 2630, 3155, 3168,
4292, 4302, 4386, 4661, 4717, 4772, 4781
\l__enumext_labelsep_viii_dim 4323, 4333, 4435,
4905, 5024, 5033
\l__enumext_labelwidth_i_dim . 3154, 3160, 3167,
3200, 4983, 4998
\l__enumext_labelwidth_v_dim . . . . . 3474, 3835
\l__enumext_labelwidth_vii_dim ... 2630, 3154,
3167, 4292, 4301, 4386, 4661, 4763, 4780
\l__enumext_labelwidth_viii_dim .. 4323, 4332,
4435, 4905, 5015, 5032
\l__enumext_leftmargin_tmp_v_bool . 108, 4064
\l__enumext_leftmargin_tmp_X_bool . . . . . 71
\l__enumext_leftmargin_tmp_X_dim . . . . . 71
\l__enumext_leftmargin_X_dim . . . . . 71
\__enumext_level: 214, 214, 720, 723, 724, 732, 734,
1028, 1032, 1036, 1104, 1108, 1112, 1116, 1199, 1201,
1203, 1205, 1247, 1249, 1251, 1253, 1257, 1291, 1297,
1302, 1304, 1307, 1310, 1323, 1326, 1634, 1638, 1644,
1707, 1709, 1711, 1714, 1721, 1723, 1725, 1728, 2323,
2325, 2327, 2355, 2356, 2358, 2414, 2422, 2426, 2430,
2634, 2635, 3212, 3213, 3217, 3218, 3219, 3227, 3235,
3236, 3239, 3246, 3247, 3251, 3254, 3256, 3290, 3291,
3292, 3295, 3298, 3307, 3308, 3310, 3311, 3314, 3633,
3646, 3653, 3661, 3664, 3666, 3668, 3669, 3670, 3671,
3674, 3679, 3685, 3691, 3698, 3711, 3713, 3716, 3717,
3719, 3723, 3729, 3754, 3759, 3770, 3772
\l__enumext_level_h_int 118, 28, 263, 286, 300, 758,
793, 1616, 2139, 2159, 2524, 2758, 2770, 3641, 4598,
4599
\l__enumext_level_int . 99, 28, 216, 273, 285, 301,
542, 1211, 1336, 1615, 2133, 2165, 2501, 2511, 2517,
2523, 2530, 2539, 2544, 2757, 2769, 2985, 3552, 3597,
3598, 3609, 3617, 3631, 3644, 3675, 3800, 4151, 4641,
4651, 4877, 5726, 5730, 5736, 5740
\__enumext_list_arg_two_i: . . . . . 3517
\__enumext_list_arg_two_ii: . . . . . 3517
\__enumext_list_arg_two_iii: . . . . . 3517

```

```

\__enumext_list_arg_two_iv: . . . . . 3517
\__enumext_list_arg_two_v: . 95, 3517, 3905, 4065
\__enumext_list_arg_two_vii: . . . . . 3558, 4578
\__enumext_list_arg_two_viii: . . . . . 3558, 4841
\l__enumext_listoffset_v_dim . 3837, 3891, 3894
\l__enumext_listparindent_vii_dim 4788, 4792
\l__enumext_listparindent_viii_dim 5040, 5044
\__enumext_log_answer_vars: . 35, 367, 375, 2992
\__enumext_log_global_vars: . 35, 367, 367, 2991
\__enumext_make_label: . 41, 92, 3270, 3270, 3546
\__enumext_make_label_box: . . . 3270, 3274, 3279,
3302
\__enumext_make_label_std: . . . 3270, 3282, 3286
\l__enumext_mark_answer_sym_tl 77, 2277, 2480,
2647, 3172, 3185, 4987
\l__enumext_mark_position_str 130, 2281, 2282,
2308, 2309, 2478
\l__enumext_mark_ref_sym_tl . 2294, 2619, 3114
\l__enumext_meta_path_tl . 126, 5314, 5315, 5317,
5318
\c__enumext_meta_paths_prop . . . . . 132, 5290
\__enumext_mini_addvspace_vii: 59, 1595, 1595,
4460
\__enumext_mini_addvspace_viii: 59, 1595, 1601,
4525
__enumext_mini_env* . . . . . 540
\__enumext_mini_page 1644, 1671, 3723, 3822, 4462,
4527, 4548
\__enumext_mini_right_cmd:n 60, 1629, 1631, 1631
\__enumext_mini_set_vskip_vii: 58, 1538, 1538,
1597
\__enumext_mini_set_vskip_viii: 58, 1538, 1560,
1603
\__enumext_minipage:w 36, 382, 390, 547, 4485, 4787,
5039
\l__enumext_minipage_active_v_bool 3819, 3842,
3867
\g__enumext_minipage_active_vii_bool . 115,
4474, 4483, 4505
\l__enumext_minipage_active_vii_bool . 4456,
4467
\g__enumext_minipage_active_viii_bool 4538,
4546, 4565
\l__enumext_minipage_active_viii_bool 4521,
4532
\g__enumext_minipage_active_X_bool . . . 174
\l__enumext_minipage_active_X_bool . . . . 87
\__enumext_minipage_add_space: . 55, 102, 1287,
1313, 3721
\g__enumext_minipage_after_skip 87, 1542, 1554,
4503, 4563
\l__enumext_minipage_after_skip . . 54, 102, 87,
1300, 1340, 1342, 1347, 1350, 1354, 1359, 1363, 1366,
1370, 1382, 1387, 1390, 1394, 1399, 1403, 1406, 1410,
1421, 1426, 1429, 1433, 1438, 1442, 1445, 1449, 1461,
1475, 1508, 1510, 1515, 1517, 1519, 1523, 1527, 1529,
1531, 1562, 1575, 1589, 1640, 1667, 3877
\g__enumext_minipage_center_vii_bool . 4489,
4506
\g__enumext_minipage_center_viii_bool 4550,
4566
\g__enumext_minipage_center_X_bool . . . 174
\l__enumext_minipage_hsep_v_dim . . . . . 3817
\l__enumext_minipage_hsep_vii_dim . . . . 4454
\l__enumext_minipage_hsep_viii_dim . . . 4519
\l__enumext_minipage_left_skip 87, 1462, 1540,
1545, 1549, 1563, 1567, 1581, 1599, 1605
\l__enumext_minipage_left_v_dim . . 3815, 3822
\l__enumext_minipage_left_vii_dim 4450, 4462
\l__enumext_minipage_left_viii_dim 4515, 4527
\l__enumext_minipage_left_X_dim . . . . . 87
\g__enumext_minipage_right_skip 87, 1541, 1546,
1550, 4488, 4549
\l__enumext_minipage_right_skip . 54, 87, 1289,
1295, 1300, 1302, 1304, 1463, 1464, 1470, 1475, 1476,
1477, 1482, 1564, 1571, 1585, 1646, 1673
\l__enumext_minipage_right_v_dim . 1662, 1671,
3813, 3817
\g__enumext_minipage_right_vii_dim 115, 4458,
4485, 4508
\l__enumext_minipage_right_vii_dim 115, 4448,
4453, 4459
\g__enumext_minipage_right_viii_dim . . 4523,
4548, 4568
\l__enumext_minipage_right_viii_dim . . 4513,
4518, 4524
\g__enumext_minipage_right_X_dim . . . . . 174
\g__enumext_minipage_right_X_skip . . . . 174
\__enumext_minipage_set_skip: . 54, 1287, 1287,
1315
\g__enumext_minipage_stat_int . . 102, 87, 1651,
1678, 3720, 3731, 3736, 3820, 3869, 3874
\l__enumext_minipage_temp_skip 87, 1361, 1371,
1374, 1401, 1411, 1414, 1440, 1450, 1453, 1525, 1532,
1534
\l__enumext_miniright_code_vii_box 4496, 4500
\g__enumext_miniright_code_vii_tl 116, 4491,
4498, 4507
\l__enumext_miniright_code_viii_box . . 4557,
4561
\g__enumext_miniright_code_viii_tl 4552, 4559,
4567
\l__enumext_miniright_code_X_box . . . . . 174
\l__enumext_mode_box_bool . . . . 612, 3277, 3447
\__enumext_multi_addvspace: 53, 101, 1242, 1242,
3682
\__enumext_multi_set_vskip: 52, 1197, 1197, 1244
\l__enumext_multicols_above_ii_skip . . 1216
\l__enumext_multicols_above_iii_skip . . 1225
\l__enumext_multicols_above_iv_skip . . 1234
\l__enumext_multicols_above_v_skip 1261, 1275,
1285, 1476
\l__enumext_multicols_above_X_skip . . . . 79
\l__enumext_multicols_below_ii_skip . . 1343,
1352, 1356, 1368, 1373
\l__enumext_multicols_below_iii_skip . 1383,
1392, 1396, 1408, 1413
\l__enumext_multicols_below_iv_skip . . 1422,
1431, 1435, 1447, 1452
\l__enumext_multicols_below_v_skip 1265, 1279,
1477, 1511, 1518, 1520, 1530, 1533, 3859
\l__enumext_multicols_below_X_skip . . . . 79
\g__enumext_multicols_right_X_skip . . . . 79
\__enumext_multicols_start: 100, 102, 3658, 3658,
3725
\__enumext_multicols_stop: 101, 1636, 3688, 3688,
3741
\__enumext_nested_base_line_fix: . 47, 99, 949,
955, 3613

```

`__enumext_newlabel:nn` 31, 37, 78, [428](#), 428, 2555, [3073](#)
`\l__enumext_newlabel_arg_one_tl` 31, 37, 78, 88, [163](#), [2548](#), [2556](#), [2618](#), [3062](#), [3074](#), [3112](#)
`\l__enumext_newlabel_arg_two_tl` 31, 37, 77, [163](#), [2504](#), [2514](#), [2527](#), [2542](#), [2557](#), [3049](#), [3054](#), [3059](#), [3075](#)
`__enumext_parse_foreach_keys:n` .. [5339](#), [5355](#), [5372](#)
`__enumext_parse_foreach_keys:nn` . [5339](#), [5362](#), [5374](#)
`__enumext_parse_keys:n` 47, [64](#), [3604](#), [3604](#), [3766](#)
`__enumext_parse_keys_vii:n` [64](#), [4573](#), [4611](#), [4611](#)
`__enumext_parse_keys_viii:n` . [4837](#), [4882](#), [4882](#)
`__enumext_parse_save_key:n` [74](#), [2348](#), [2353](#), [2353](#)
`__enumext_parse_save_key_vii:n` [74](#), [2343](#), [2353](#), [2361](#)
`__enumext_parse_series:n` [64](#), [99](#), [118](#), [1836](#), [1836](#), [3612](#), [4617](#)
`__enumext_parse_store_keys:n` [99](#)
`\l__enumext_parsep_i_skip` [1214](#), [1218](#)
`\l__enumext_parsep_ii_skip` [1223](#), [1227](#)
`\l__enumext_parsep_iii_skip` [1232](#), [1236](#)
`\l__enumext_parsep_vii_skip` [4789](#)
`\l__enumext_parsep_viii_skip` [5041](#)
`\l__enumext_partopsep_v_skip` . [1277](#), [1281](#), [1472](#), [1495](#)
`\l__enumext_partopsep_viii_skip` [1573](#)
`__enumext_phantomsection:` [36](#), [395](#), [421](#), [425](#), [441](#)
`__enumext_pre_itemsep_skip:` [54](#), [55](#), [1305](#), [1334](#), [1334](#)
`__enumext_print_footnote:` .. [443](#), [466](#), [498](#), [503](#), [530](#), [535](#)
`__enumext_print_footnote_standar:` [476](#), [492](#), [554](#), [5059](#)
`__enumext_print_footnote_starred:` [524](#), [4803](#)
`__enumext_print_keyans_box:NN` [77](#), [2472](#), [2472](#), [2485](#), [2629](#), [2633](#), [3159](#), [3199](#), [4983](#), [4998](#)
`\l__enumext_print_keyans_i_tl` [5123](#), [5145](#)
`\l__enumext_print_keyans_ii_tl` ... [5127](#), [5146](#)
`\l__enumext_print_keyans_iii_tl` .. [5131](#), [5147](#)
`\l__enumext_print_keyans_iv_tl` ... [5135](#), [5148](#)
`\l__enumext_print_keyans_star_bool` . [47](#), [129](#), [130](#), [130](#), [961](#), [969](#), [5168](#), [5173](#)
`\l__enumext_print_keyans_starred_tl` [129](#), [130](#), [5119](#), [5166](#)
`\l__enumext_print_keyans_vii_tl` [129](#), [5139](#), [5149](#)
`\l__enumext_print_keyans_X_tl` [130](#)
`__enumext_printkeyans:nnn` [129](#), [5142](#), [5150](#), [5153](#)
`__enumext_redefine_item:` . [92](#), [3259](#), [3259](#), [3545](#)
`\l__enumext_ref_key_arg_tl` [42](#), [50](#), [229](#), [713](#), [714](#), [726](#), [757](#), [760](#), [770](#), [776](#), [786](#), [827](#), [828](#), [838](#)
`\l__enumext_ref_the_count_tl` . [42](#), [50](#), [720](#), [723](#), [726](#), [765](#), [767](#), [770](#), [781](#), [783](#), [786](#), [833](#), [835](#), [838](#)
`__enumext_regex_counter_style:` .. [32](#), [42](#), [224](#), [224](#), [721](#), [766](#), [782](#), [834](#)
`__enumext_register_counter_style:Nn` .. [576](#), [576](#), [581](#), [582](#), [583](#), [584](#), [585](#)
`__enumext_remove_extra_parsep_vii:` .. [4591](#), [4814](#), [4814](#)
`__enumext_remove_extra_parsep_viii:` . [4854](#), [5070](#), [5070](#)
`__enumext_renew_footnote:` .. [443](#), [447](#), [482](#), [487](#), [514](#), [519](#)
`__enumext_renew_footnote_standar:` [476](#), [476](#), [546](#), [5035](#)
`__enumext_renew_footnote_starred:` [508](#), [4783](#)
`\l__enumext_renew_the_count_v_tl` [836](#), [844](#), [846](#)
`\l__enumext_renew_the_count_vii_tl` [768](#), [795](#), [797](#)
`\l__enumext_renew_the_count_viii_tl` [784](#), [802](#), [804](#)
`\l__enumext_renew_the_count_X_tl` [50](#)
`__enumext_rescan_anskey_env:n` .. [84](#), [86](#), [2867](#), [2962](#), [2970](#)
`__enumext_reset_global_bool:` .. [343](#), [346](#), [355](#)
`__enumext_reset_global_int:` ... [343](#), [345](#), [349](#)
`__enumext_reset_global_tl:` [343](#), [347](#), [361](#)
`__enumext_reset_global_vars:` . [35](#), [87](#), [343](#), [343](#), [3000](#)
`\l__enumext_resume_active_bool` [64](#), [66](#), [61](#), [1840](#), [1960](#)
`__enumext_resume_counter:` . [66](#), [1958](#), [1964](#), [1971](#)
`__enumext_resume_counter:n` . [64](#), [66](#), [1929](#), [1934](#), [1958](#), [1958](#), [2028](#), [2036](#)
`__enumext_resume_counter_save_ans:` .. [66](#), [67](#), [1958](#), [1969](#), [2001](#)
`__enumext_resume_counter_series:` [66](#), [67](#), [1958](#), [1967](#), [1984](#)
`\g__enumext_resume_int` ... [61](#), [1881](#), [1975](#), [1976](#)
`__enumext_resume_last:n` .. [64](#), [1836](#), [1842](#), [1855](#)
`\l__enumext_resume_name_tl` [61](#), [1877](#), [1885](#), [1888](#), [1904](#), [1912](#), [1915](#), [1961](#), [1962](#), [1990](#), [1997](#)
`__enumext_resume_save_counter:` . [65](#), [102](#), [119](#), [1868](#), [1868](#), [3747](#), [4635](#)
`__enumext_resume_series:n` . [66](#), [1804](#), [1925](#), [1925](#)
`__enumext_resume_starred:` . [67](#), [1805](#), [2022](#), [2022](#)
`\g__enumext_resume_vii_int` [61](#), [1908](#), [1980](#), [1981](#)
`\l__enumext_rightmargin_vii_dim` .. [4304](#), [4308](#), [4313](#)
`\l__enumext_rightmargin_viii_dim` . [4335](#), [4339](#), [4344](#)
`__enumext_safe_exec:` .. [39](#), [99](#), [3593](#), [3593](#), [3765](#)
`__enumext_safe_exec_vii:` . [39](#), [4572](#), [4594](#), [4594](#)
`__enumext_safe_exec_viii:` [123](#), [4836](#), [4858](#), [4858](#)
`__enumext_second_part:` .. [102](#), [3727](#), [3727](#), [3779](#)
`__enumext_second_part_v:` ... [3809](#), [3865](#), [3913](#)
`\l__enumext_series_name_tl` [66](#)
`\l__enumext_series_str` .. [65](#), [99](#), [118](#), [1802](#), [1838](#), [1846](#), [1847](#), [1849](#), [1851](#), [1872](#), [1875](#), [1879](#), [1899](#), [1902](#), [1906](#), [3608](#), [4615](#)
`__enumext_set_error:nn` [5249](#), [5286](#), [5288](#)
`__enumext_set_item_width:` [102](#), [3749](#), [3749](#), [3775](#)
`__enumext_set_parse:n` [5249](#), [5260](#), [5276](#)
`\l__enumext_setkey_tmpa_int` ... [121](#), [5253](#), [5257](#)
`\l__enumext_setkey_tmpa_seq` .. [121](#), [5251](#), [5261](#), [5267](#), [5269](#), [5271](#), [5283](#)
`\l__enumext_setkey_tmpa_tl` [121](#), [5259](#), [5263](#)
`\l__enumext_setkey_tmppb_seq` .. [121](#), [5252](#), [5255](#), [5259](#), [5260](#)
`\l__enumext_setkey_tmppb_tl` [121](#), [5278](#), [5280](#), [5281](#)
`\l__enumext_show_answer_bool` . [2288](#), [2312](#), [2641](#), [3129](#), [3143](#), [4194](#), [4981](#)
`__enumext_show_length:nnn` .. [49](#), [232](#), [232](#), [5497](#), [5498](#), [5499](#), [5500](#), [5501](#), [5502](#), [5503](#), [5504](#), [5505](#), [5506](#), [5512](#), [5513](#), [5514](#), [5515](#), [5516](#), [5517](#), [5518](#), [5519](#), [5520](#), [5521](#)
`\l__enumext_show_position_bool` ... [2291](#), [2315](#), [2645](#), [3133](#), [3144](#), [4195](#), [4985](#)

`\g__enumext_standar_bool` 33, 99, 34, 262, 265, 284, 358, 478, 494, 1870, 1935, 1947, 1973, 1986, 2024, 2164, 2178, 2509, 2522, 2537, 3628
`\l__enumext_standar_bool` 99, 102, 34, 2510, 3600, 3746, 4608
`\l__enumext_standar_first_bool` 33, 99, 34, 289, 1857, 2004, 2066, 2073
`__enumext_standar_item_vii:w` . 120, 4665, 4683, 4685
`__enumext_standar_item_viii:w` 125, 4909, 4927, 4929
`__enumext_standar_ref:` 43, 711, 730, 3547
`__enumext_standar_ref:n` 42, 703, 711, 711
`\g__enumext_standar_series_tl` . 61, 1859, 1860, 2026, 2029
`__enumext_standar_unknown_keys:n` 3377, 3381, 3385
`__enumext_standar_unknown_keys:nn` 3377, 3387, 3389
`\g__enumext_starred_bool` 33, 118, 34, 272, 275, 299, 359, 510, 526, 1897, 1940, 1951, 1978, 1993, 2032, 2138, 2184, 2500, 3043, 4509
`\l__enumext_starred_bool` 118, 119, 123, 34, 1621, 2538, 2573, 2579, 2627, 2916, 2921, 3152, 3165, 3601, 4607, 4634, 4870, 4874
`__enumext_starred_columns_set_vii:` . 4286, 4286, 4582
`__enumext_starred_columns_set_viii:` . 4286, 4317, 4845
`\l__enumext_starred_first_bool` 33, 118, 34, 304, 959, 968, 1862, 2013, 2066, 2073
`__enumext_starred_item:nn` . . 3222, 3222, 3265
`__enumext_starred_item_exec:` . 126, 4943, 4973, 5013
`__enumext_starred_item_vii:w` . 120, 4665, 4682, 4699
`__enumext_starred_item_vii_aux_i:w` . 4665, 4704, 4707
`__enumext_starred_item_vii_aux_ii:w` . 4665, 4705, 4710, 4712
`__enumext_starred_item_vii_aux_iii:w` 4665, 4715, 4722
`__enumext_starred_item_viii:w` 125, 126, 4926, 4943, 4943
`__enumext_starred_item_viii_aux_i:w` . 126, 4943, 4948, 4951
`__enumext_starred_item_viii_aux_ii:w` . 126, 4943, 4949, 4966, 4968
`__enumext_starred_joined_item_vii:n` 114, 120, 4348, 4348, 4680
`__enumext_starred_joined_item_viii:n` . 114, 125, 4348, 4397, 4924
`__enumext_starred_ref:` 44, 755, 791, 3578
`__enumext_starred_ref:n` 43, 749, 755, 755
`\g__enumext_starred_series_tl` . 61, 1864, 1865, 2034, 2037
`__enumext_starred_unknown_keys:n` 3359, 3361, 3363
`__enumext_starred_unknown_keys:nn` 3359, 3365, 3367
`__enumext_start_from:NNn` 45, 849, 849, 862, 884, 890
`\l__enumext_start_i_int` 1976, 1988, 2007
`__enumext_start_item_tmp_vii:` 117, 4585, 4665, 4665
`__enumext_start_item_tmp_viii:` . . 4848, 4909, 4909
`__enumext_start_item_vii:w` 120, 122, 4691, 4696, 4719, 4726, 4774, 4774
`__enumext_start_item_viii:w` . . 125, 4935, 4940, 4971, 5026, 5026
`\g__enumext_start_line_tl` 33, 34, 292, 307, 364, 2208, 2213, 2218, 2232, 2237, 2242
`__enumext_start_list:nn` . 36, 96, 382, 384, 3769, 3902, 4576, 4839
`__enumext_start_list_tag:n` . . 3915, 3939, 4784, 5036
`__enumext_start_mini_vii:` 118, 4446, 4446, 4626
`__enumext_start_mini_viii:` . . 124, 4511, 4511, 4893
`__enumext_start_save_ans_msg:` 68, 2050, 2050, 2075
`__enumext_start_store_level:` . 100, 3622, 3622, 3768
`__enumext_start_store_level_vii:` 119, 4575, 4637, 4637
`\l__enumext_start_vii_int` . . . 1981, 1995, 2016
`\l__enumext_start_X_int` 100
`__enumext_stop_item_tmp_vii:` . . 117, 119, 122, 4584, 4590, 4667, 4776
`__enumext_stop_item_tmp_viii:` 125, 4847, 4853, 4911, 5028
`__enumext_stop_item_vii:` 122, 4774, 4776, 4796
`__enumext_stop_item_viii:` . . 5026, 5028, 5052
`__enumext_stop_list:` 36, 115, 119, 382, 385, 3693, 3701, 3855, 3862, 4469, 4477, 4534, 4541
`__enumext_stop_list_tag:n` . . 3915, 3955, 4799, 5055
`__enumext_stop_mini_vii:` 115, 119, 4446, 4465, 4630
`__enumext_stop_mini_viii:` 124, 4511, 4530, 4897
`__enumext_stop_save_ans_msg:` . 68, 2050, 2055, 2989
`__enumext_stop_start_list_tag:` . . 3915, 3947, 4786, 5038
`__enumext_stop_store_level:` . . 100, 101, 3651, 3651, 3694, 3702
`__enumext_stop_store_level_vii:` . . 115, 119, 4470, 4478, 4637, 4647
`\l__enumext_store_active_bool` 30, 69, 112, 2005, 2014, 2082, 2714, 3626, 3639, 3787, 3795, 4147, 4639, 4649, 4860, 4876
`__enumext_store_active_keys:n` . . 74, 99, 2321, 2321, 3619
`__enumext_store_active_keys_vii:n` . 74, 118, 2321, 2331, 4618
`__enumext_store_addto_prop:n` 75, 87, 2396, 2396, 2404, 2564, 3024, 4976
`__enumext_store_addto_seq:n` 76, 89, 2405, 2405, 2409, 2416, 2430, 2438, 2447, 2461, 2469, 2622, 3117
`\l__enumext_store_anskey_arg_tl` . . 30, 79, 112, 2570, 2575, 2577, 2582, 2589, 2592, 2602, 2607, 2610, 2616, 2622
`__enumext_store_anskey_code:n` 78, 81, 86, 2561, 2561, 2707, 2960, 2968
`\l__enumext_store_anskey_env_tl` . . 30, 85, 112, 2890, 2894, 2900, 2962, 2970
`\l__enumext_store_anskey_opt_tl` 30, 85, 86, 112, 2891, 2918, 2924, 2931, 2937, 2947, 2957, 2966


```

\__enumext_store_anskey_safe_outer: . . . . 81
\g__enumext_store_columns_break_bool . 2814,
    2915, 2977
\l__enumext_store_columns_break_bool . 2572,
    2663
\l__enumext_store_current_label_tl 30, 87, 89,
    126, 112, 3005, 3008, 3011, 3017, 3022, 3024, 3081,
    3084, 3087, 3093, 3098, 3108, 3117, 4953, 4958, 4962,
    4975, 4976, 4978
\l__enumext_store_current_label_tmp_tl . 30,
    112, 3413, 3418
\l__enumext_store_current_opt_arg_tl 30, 126,
    112, 3127, 3140, 3146, 4964
\__enumext_store_internal_ref: . . 77, 78, 2486,
    2486, 2567
\g__enumext_store_item_join_int . . 2817, 2922,
    2926, 2978
\l__enumext_store_item_join_int . . 2580, 2584,
    2666
\g__enumext_store_item_star_bool . 2819, 2929,
    2979
\l__enumext_store_item_star_bool . 2587, 2668
\g__enumext_store_item_symbol_sep_dim 2824,
    2944, 2949, 2981
\l__enumext_store_item_symbol_sep_dim 2599,
    2604, 2673
\g__enumext_store_item_symbol_tl . 2822, 2935,
    2939, 2980
\l__enumext_store_item_symbol_tl . 2590, 2594,
    2671
\l__enumext_store_keyans_item_opt_sep_
    tl . . . . 2274, 3015, 3019, 3091, 3095, 4956, 4960
\__enumext_store_level_close: . 76, 2410, 2434,
    3655
\__enumext_store_level_close_vii: . 76, 2441,
    2465, 4653
\__enumext_store_level_open: 76, 100, 2410, 2410,
    3634, 3647
\__enumext_store_level_open_vii: . . 76, 2441,
    2441, 4643
\g__enumext_store_name_tl 30, 69, 112, 363, 370,
    371, 372, 373, 2058, 2084, 2207, 2212, 2217, 2231,
    2236, 2241, 2987
\l__enumext_store_name_tl 30, 68, 70, 112, 1891,
    1894, 1918, 1921, 2009, 2018, 2053, 2062, 2063, 2084,
    2085, 2086, 2088, 2089, 2091, 2093, 2094, 2096, 2098,
    2099, 2123, 2398, 2400, 2407, 2550, 2551, 2653, 2896,
    3064, 3065, 3178, 3191, 4993
\l__enumext_store_ref_key_bool 78, 2297, 2565,
    2613, 3028, 3105
\l__enumext_store_save_key_vii_bool . . 2333,
    2363
\l__enumext_store_save_key_vii_tl 2335, 2336,
    2364, 2365, 2445, 2453, 2457, 2461
\l__enumext_store_save_key_X_bool . . 74, 130
\l__enumext_store_save_key_X_tl . . . . 74, 130
\l__enumext_store_upper_level_X_bool . . 130
\__enumext_storing_exec: . 68, 69, 83, 2060, 2076,
    2080
\__enumext_storing_set:n . . 68, 2045, 2060, 2060
\l__enumext_the_counter_v_tl . . . . . 835
\l__enumext_the_counter_vii_tl . . . . . 767
\l__enumext_the_counter_viii_tl . . . . . 783
\l__enumext_the_counter_X_tl . . . . . 50
\__enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 86, 94, 99,
    100, 111, 134, 141, 166, 170, 174, 194, 608, 617, 1798,
    1809, 2041, 2049, 2102, 2120, 2261, 2302, 2303, 2320,
    2339, 2352, 2488, 2495, 2496, 2517, 2530, 2533, 2544,
    3030, 3037, 3337, 3344, 3377, 3384, 3517, 3557, 3558,
    3592
\__enumext_tmp:nn 618, 639, 640, 674, 675, 690, 879,
    904, 981, 1003, 1004, 1024, 1078, 1086, 1087, 1101,
    1166, 1182, 1183, 1196, 1687, 1703, 3321, 3336
\__enumext_tmp:nnn 691, 707, 708, 709, 710, 737, 753,
    754
\__enumext_tmp:nnnnnn 905, 930, 933, 936, 938, 940,
    943, 946
\__enumext_tmp:w . . . . . 5098, 5101
\l__enumext_tmpa_vii_int 4296, 4299, 4308, 4339
\l__enumext_tmpa_viii_int . . . . . 4327, 4330
\l__enumext_tmpa_X_dim . . . . . 174
\l__enumext_tmpa_X_int . . . . . 174
\l__enumext_topsep_v_skip 1263, 1267, 1466, 4140
\l__enumext_topsep_vii_skip . . 1543, 1552, 1556
\l__enumext_topsep_viii_skip . 1565, 1587, 1591
\__enumext_undefine_anskey_env: . 82, 87, 2747,
    2747, 2998
\__enumext_unskip_unkern: . . 33, 238, 238, 1316,
    1488, 3696, 3697, 3737, 3857, 3858, 3875, 4790, 4791,
    5042, 5043
\l__enumext_vspace_a_star_v_bool . . . . 1736
\l__enumext_vspace_a_star_vii_bool . . 1758
\l__enumext_vspace_a_star_viii_bool . . 1769
\l__enumext_vspace_a_star_X_bool . . . . 100
\__enumext_vspace_above: 61, 101, 1704, 1704, 3707
\__enumext_vspace_above_v: . 62, 1732, 1732, 3811
\l__enumext_vspace_above_v_skip . . 1734, 1738,
    1740
\__enumext_vspace_above_vii: 62, 118, 1754, 1754,
    4623
\l__enumext_vspace_above_vii_skip 1756, 1760,
    1762
\__enumext_vspace_above_viii: . 62, 1754, 1765,
    4891
\l__enumext_vspace_above_viii_skip 1767, 1771,
    1773
\l__enumext_vspace_b_star_v_bool . . . . 1747
\l__enumext_vspace_b_star_vii_bool . . 1780
\l__enumext_vspace_b_star_viii_bool . . 1791
\l__enumext_vspace_b_star_X_bool . . . . 100
\__enumext_vspace_below: 61, 102, 1718, 1718, 3745
\__enumext_vspace_below_v: . 62, 1743, 1743, 3884
\l__enumext_vspace_below_v_skip . . 1745, 1749,
    1751
\__enumext_vspace_below_vii: 63, 119, 1776, 1776,
    4633
\l__enumext_vspace_below_vii_skip 1778, 1782,
    1784
\__enumext_vspace_below_viii: . 63, 1776, 1787,
    4899
\l__enumext_vspace_below_viii_skip 1789, 1793,
    1795
\__enumext_widest_from:nnn . . 45, 863, 863, 878,
    897
\g__enumext_widest_label_tl 28, 40, 67, 596, 600,
    604
\l__enumext_wrap_label_opt_v_bool . . . . 3407
\l__enumext_wrap_label_opt_vii_bool 120, 4690
\l__enumext_wrap_label_opt_viii_bool . . 125,
    4934

```

<code>\l__enumext_wrap_label_opt_X_bool</code>	100
<code>\l__enumext_wrap_label_v_bool</code>	3403, 3407, 3415, 3462, 3477
<code>\l__enumext_wrap_label_vii_bool</code>	120, 4690, 4694, 4702, 4766
<code>\l__enumext_wrap_label_viii_bool</code>	125, 4934, 4938, 4946, 5018
<code>\l__enumext_wrap_label_X_bool</code>	100
<code>__enumext_wrapper_label_v:n</code>	3464, 3479, 4203
<code>__enumext_wrapper_label_vii:n</code>	4768
<code>__enumext_wrapper_label_viii:n</code>	5020
<code>\l__enumext_write_aux_file_tl</code>	31, 78, 88, 163, 2553, 2559, 3071, 3077
<code>enumext*</code>	5, 4570
<code>enumXi</code>	568
<code>enumXii</code>	568
<code>enumXiii</code>	568
<code>enumXiv</code>	568
<code>enumXv</code>	568
<code>enumXvi</code>	568
<code>enumXvii</code>	568
<code>enumXviii</code>	568
Environments provide by <code>enumext</code> :	
<code>anskey*</code>	30, 69, 74, 77, 78, 80, 82, 83, 85, 87, 100, 119, 128, 129, 134, 137
<code>enumext*</code>	27, 28, 31–33, 37, 39, 40, 43–46, 48, 49, 51, 58, 59, 62–68, 70, 71, 73–82, 85, 87, 88, 93, 94, 98–100, 105, 113, 114, 116, 119, 121–124, 127–130, 132, 136, 138, 140
<code>enumext</code>	27, 28, 32, 33, 39–54, 57, 59–61, 63–68, 70, 71, 73–82, 85, 87, 88, 91–94, 96, 97, 100, 103, 104, 108, 112, 115, 118, 119, 121, 123, 129, 130, 132, 135, 137, 139
<code>keyans*</code>	27, 28, 30–34, 37, 39, 40, 43–46, 48, 49, 51, 58, 59, 62, 63, 69, 70, 72, 73, 75, 83, 87, 93, 98, 105, 113, 114, 123, 124, 136, 138, 140
<code>keyanspic</code>	27, 28, 30, 31, 34, 39, 41, 44, 69, 70, 72, 75, 76, 83, 87–89, 93, 105–111, 138
<code>keyans</code>	27, 28, 30, 31, 33, 34, 39–41, 44–46, 48, 49, 51, 53, 57, 59–62, 69, 70, 72, 73, 75, 76, 83, 87–89, 93, 95–97, 103, 105, 107, 108, 111, 115, 124, 136, 138
Environments:	
<code>center</code>	112
<code>description</code>	112
<code>enumerate</code>	112
<code>flushleft</code>	112
<code>flushright</code>	112
<code>itemize</code>	112
<code>list</code>	32, 35, 36, 80, 96, 97, 101, 103, 105, 107–109, 112, 115
<code>lrbox</code>	122
<code>minipage</code>	32, 35, 36, 38, 51, 54, 55, 107, 110, 112, 113, 115, 116, 122
<code>multicols</code>	52–55, 60, 100–102
<code>quotation</code>	112
<code>quote</code>	112
<code>scontents</code>	83, 85
<code>tabbing</code>	112
<code>trivlist</code>	112
<code>verbatim</code>	112
<code>verse</code>	112
exp commands:	
<code>\exp_after:wN</code>	5101
<code>\exp_args:Ne</code>	2959, 2967, 3616, 5089
<code>\exp_args:NV</code>	2679, 2834, 3347, 3365, 3387, 5374
<code>\exp_not:N</code>	58, 599, 726, 770, 786, 838, 1034, 1037, 1048, 1049, 1050, 1061, 1062, 1073, 1074, 2618, 2650, 2651, 3110, 3175, 3176, 3188, 3189, 4990, 4991, 5098

<code>\exp_not:n</code>	294, 309, 322, 330, 338, 665, 685, 726, 770, 786, 838, 1035, 1825, 1834, 2285, 2382, 2394, 2556, 2584, 2594, 2604, 2618, 2619, 2926, 2939, 2949, 3074, 3112, 3114, 4024, 5203, 5213, 5406, 5411
F	
<code>\fbox</code>	2268
<code>\fboxrule</code>	2268
<code>\fboxsep</code>	2268
file commands:	
<code>\file_input_stop:</code>	5810
<code>first</code>	1087
<code>font</code>	618
<code>\footnote</code>	37
<code>\footnote</code>	37, 451
<code>\footnotemark</code>	461
<code>\footnotesize</code>	2651, 3176, 3189, 4991
<code>\footnotetext</code>	445
<code>\foreachkeyans</code>	17, 133, 5339
G	
<code>\getkeyans</code>	17, 128, 5087
group commands:	
<code>\group_begin:</code>	2649, 2694, 2869, 2956, 3174, 3187, 4989, 5144
<code>\group_end:</code>	2656, 2710, 2973, 3181, 3194, 4996, 5151
H	
<code>\hbadness</code>	4801, 5057
hbox commands:	
<code>\hbox_overlap_left:n</code>	3255, 4759
<code>\hbox_set:Nn</code>	588, 4070
<code>\hbox_set_end:</code>	4800, 5056
<code>\hbox_set_to_wd:Nnw</code>	4777, 5029
<code>\hfill</code>	648, 653, 659, 660, 1643, 1670, 2618, 3110, 4473, 4537
hook commands:	
<code>\hook_gput_code:nnn</code>	9, 204, 208, 212, 393
<code>\hook_gremove_code:nn</code>	85, 2885
<code>\hook_gset_rule:nnnn</code>	394
<code>\hook_if_empty:nTF</code>	2883
<code>\hyperlink</code>	79, 89
<code>\hyperlink</code>	2618, 3110
<code>\hypertarget</code>	36
<code>\hypertarget</code>	420
I	
<code>\IfDocumentMetadataTF</code>	480, 496, 512, 528, 3272, 3442, 3941, 3949, 3957, 3991, 3999, 4007, 4093, 4103, 4111, 4117, 4122, 4168, 4177, 4261, 4269, 4471, 4535, 4581, 4589, 4735, 4844, 4852
<code>\IfHyperBoolean</code>	401
<code>\IfPackageLoadedTF</code>	11, 19, 397, 410
<code>\ignorespaces</code>	1037, 1050, 1062, 1074, 4586, 4663, 4696, 4719, 4726, 4772, 4792, 4849, 4907, 4940, 4971, 5024, 5044
<code>\inputlineno</code>	294, 309, 322, 330, 338
int commands:	
<code>\int_add:Nn</code>	4381, 4430
<code>\int_case:nn</code>	1211, 1336, 2133, 2159, 2198, 2222
<code>\int_case:nnTF</code>	240
<code>\int_compare:nNnTF</code>	542, 758, 774, 793, 800, 1306, 1325, 1479, 1497, 1609, 1625, 1637, 1665, 2246, 2252, 2718, 2722, 2726, 2734, 2780, 2784, 2788, 2985, 3006, 3047, 3052, 3057, 3082, 3170, 3598, 3609, 3631, 3644, 3660, 3675, 3690, 3731, 3796, 3800, 3828, 3853, 3869, 4042, 4151, 4155, 4351, 4361, 4377, 4400, 4410, 4426,

4599, 4603, 4641, 4651, 4804, 4816, 4865, 4877, 5060, 5072, 5257, 5389	
\int_compare_p:nNn	263, 273, 285, 286, 300, 301, 1615, 1616, 2139, 2165, 2501, 2511, 2523, 2524, 2539, 2580, 2757, 2758, 2769, 2770, 2922, 3641
\int_decr:N	4380, 4429
\int_eval:n	380, 892, 2400, 2551, 2651, 3065, 3176, 3189, 3532, 3577, 4369, 4418, 4991
\int_from_alph:n	857, 871
\int_from_roman:n	859, 873
\int_gadd:Nn	4382, 4431
\int_gdecr:N	2142, 2147, 2151, 2155, 2168
\int_gincr:N	1975, 1980, 2563, 3120, 3209, 3243, 3422, 3720, 3820, 4192, 4669, 4745, 4913, 4980
\int_gset:Nn	459, 2191
\int_gset_eq:NN	456, 1874, 1881, 1887, 1893, 1901, 1908, 1914, 1920
\int_gzero:N	351, 352, 353, 1651, 1678, 2258, 2978, 3736, 3874, 4827, 5084
\int_if_exist:NTF	1849, 1885, 1891, 1912, 1918, 2096
\int_incr:N	2733, 3597, 3791, 4041, 4598, 4668, 4864, 4912
\int_mod:nn	4818, 5074
\int_new:N	28, 29, 30, 31, 32, 33, 61, 62, 87, 104, 123, 144, 145, 156, 157, 158, 160, 171, 177, 178, 179, 180, 181, 1851, 2099
\int_set:Nn	853, 857, 859, 1988, 1995, 2007, 2016, 2870, 4255, 4256, 4296, 4327, 4350, 4356, 4372, 4399, 4405, 4421, 4801, 5057, 5253, 5391
\int_set_eq:NN	1976, 1981, 4379, 4428
\int_sign:n	2193
\int_step_function:nnN	2517, 2530, 2544
\int_step_function:nnnN	5395
\int_step_inline:nn	5305
\int_step_inline:nnn	4257
\int_to_roman:n	216, 2497, 2534
\int_use:N	373, 378, 379, 1307, 1326, 1638, 1990, 1997, 2009, 2018, 3532, 3552, 3577, 3617, 3661, 3670, 3685, 3691, 4354, 4355, 4367, 4403, 4404, 4416, 5726, 5730, 5736, 5740
\int_zero:N	4808, 5064
\item	91, 95, 119, 122, 124, 127, 386, 2418, 2424, 2449, 2455, 2577, 3084, 3087, 3261, 3426, 4097, 4099, 4583, 4585, 4846, 4848, 4978
\item*	5, 15, 72, 3424
item-pos*	3321
item-sym*	3321
\itemindent	97
\itemindent	96
itemindent	981
\itemsep	4086
\itemwidth	558, 2268, 3751, 3757, 3888, 3894, 4390, 4394, 4439, 4443

K

keyans	15, 3897
keyans*	15, 4834
keyanspic	16, 4088
Keys for \anskey provide by enumext:	
break-col	79, 80, 83–85
item-join	79, 80, 83–85
item-pos*	79, 80, 83, 84, 86
item-star	79, 80, 83, 84, 86
item-sym*	79, 80, 83, 84, 86

Keys for anskey* provide by enumext:	
break-col	79, 80, 83–85
item-join	79, 80, 83–85
item-pos*	79, 80, 83, 84, 86
item-star	79, 80, 83, 84, 86
item-sym*	79, 80, 83, 84, 86
Keys for environments provide by enumext:	
above*	29, 47, 61, 62, 101, 118
above	29, 47, 61, 62, 101, 118, 124
after	49, 50, 102, 119, 124
align	29, 41, 91, 92, 96, 121, 134
base-fix	47, 64, 75, 99
before*	49, 50, 101, 118, 124
before	49, 50
below*	29, 61–63, 102, 119
below	29, 61–63, 102, 119, 124
check-ans	31–33, 68–72, 75, 86, 89, 102, 103, 119, 123, 136
columns-sep	51, 100, 122
columns	29, 51, 61, 100
first	49, 50, 122
font	41, 92, 96, 111, 121
item-pos*	91, 93
item-sym*	30, 91, 93
itemindent	29, 48, 91, 95, 122
itemsep	46, 98, 122
label-pos	107, 108, 110, 111
label-sep	107
labelsep	41, 97, 121
labelwidth	40–45, 97, 121
label	28, 40, 42, 45, 108, 112
layout-sep	107
layout-sty	107, 109, 112
layout-top	107
lisparindent	98
list-indent	29, 48, 108
list-offset	48, 102, 105
listparindent	48, 122
mark-ans	73, 75, 80
mark-pos	73, 134
mark-ref	73, 75, 77, 79
mini-env	29, 38, 51, 60, 61, 75, 101, 113, 115, 116, 118, 119, 124
mini-right*	29, 32, 51, 75, 116, 118, 119
mini-right	29, 32, 51, 59, 75, 116, 118, 119
mini-sep	29, 51, 75, 101
mode-box	40, 91–93, 96
no-store	31, 68–70, 75, 81, 91
noitemsep	46
nosep	46
parindent	98
parsep	46, 98, 108, 122
partopsep	46
ref	28, 32, 42–44, 136
resume*	28, 63, 64, 67–69, 75, 102, 119, 130
resume	28, 35, 63–69, 75, 102, 119, 130
rightmargin	48, 113
save-ans	30, 35, 64–68, 70, 71, 74–76, 81–83, 86, 87, 89, 95, 103, 110, 121, 123, 124, 126, 128–130, 136
save-key	30, 64, 74, 99, 118
save-pos	75
save-ref	31, 37, 73, 75, 77–79, 88, 89, 95, 126
save-sep	73, 75, 87, 126
series	28, 63–67, 75, 99, 102, 118, 119, 130
show-ans	73, 75, 77, 78, 80, 95, 111, 126

show-length	33, 49, 135, 136
show-pos	30, 73, 77, 78, 80, 89, 95, 111, 126
start*	29, 45, 64
start	29, 32, 45, 64
store-key	74
topsep	46, 47, 108
widest	28, 32, 45
wrap-ans	39, 73, 75, 77, 80
wrap-label*	29, 41, 91, 92, 95, 96, 120, 121, 125
wrap-label	29, 41, 91, 92, 95, 96, 108, 111, 120, 121, 125
wrap-opt	73, 75, 95, 111
keys commands:	
\keys_define:nn	610, 620, 642, 677, 693, 739, 808, 881, 907, 949, 983, 1006, 1080, 1089, 1168, 1185, 1689, 1800, 2043, 2104, 2263, 2305, 2341, 2346, 2661, 2812, 2848, 3323, 3339, 3359, 3379, 4013, 5115, 5215, 5331, 5339
\keys_if_exist_p:nn	5327, 5328
\l_keys_key_str	81, 84, 2679, 2834, 3347, 3365, 3387, 5374, 5482
\keys_precompile:nnN	129, 200, 200, 5117, 5121, 5125, 5129, 5133, 5137, 5357
\keys_set:nn	634, 975, 1191, 1694, 1699, 1937, 1942, 2029, 2037, 2699, 3611, 3616, 3807, 4031, 4034, 4052, 4616, 4886, 5219, 5224, 5225, 5226, 5227, 5230, 5235, 5236, 5237, 5238, 5239, 5240, 5241, 5273, 5383
\keys_set_known:nn	2966
keyval commands:	
\keyval_parse:NNn	1814, 2371, 5191
L	
label	691, 737, 808
label-pos	4013
label-sep	4013
Labels provide by enumext:	
\Alph*	40
\Roman*	40
\alph*	40
\arabic*	32, 40
\roman*	40
\labelsep	4082
labelsep	618
\labelwidth	40
\labelwidth	4082
labelwidth	618
\lastkern	249
\lastnodetype	240
\lastskip	244
layout-sep	4013
layout-sty	4013
layout-top	4013
\leftmargin	97
\leftmargin	96, 4082
legacy commands:	
\legacy_if:nTF	4730, 4733, 5003, 5006
\legacy_if_gset_false:n	548, 4486
\legacy_if_set_false:n	4732, 5005
\legacy_if_set_true:n	4695, 4718, 4725, 4739, 4939, 4970
\linewidth	101
\linewidth	3715, 3751, 3817, 3888, 4254, 4299, 4330, 4452, 4517
\list	384
list-indent	981
list-offset	981

\listparindent	4084
listparindent	981
M	
\makebox	112
\makebox	2476, 2478, 3306, 3474, 4185, 4763, 5015
\makelabel	91, 92, 96, 112
\makelabel	91, 95, 3288, 3304, 3458, 3472
mark-ans	2261
mark-pos	2261, 2303
mark-ref	2261
mini-env	1166
mini-sep	1166
\minipage	390
\miniright	11, 59, 1607, 1655, 1682, 3734, 3872
mode commands:	
\mode_if_math:TF	2742, 2796
\mode_if_vertical:TF	1245, 1273, 1293, 1317, 1468, 1489
\mode_leave_vertical:	964, 971, 1034, 1048, 2474, 3253, 4757
mode-box	608
msg commands:	
\msg_error:nn	1657, 1684, 2703, 2736, 2740, 2794, 2902, 3798, 3802, 4044, 4101, 4153, 4601, 4867, 4879, 5242, 5301
\msg_error:nnn	716, 762, 778, 830, 1611, 1618, 1623, 1653, 1680, 1949, 1953, 2068, 2685, 2744, 2762, 2774, 2782, 2786, 2790, 2798, 2840, 3353, 3371, 3393, 4605, 4872, 5103, 5112, 5184, 5289, 5320, 5329, 5366, 5387
\msg_error:nnnn	2688, 2716, 2720, 2724, 2728, 2843, 3356, 3374, 3396, 3789, 4149, 4157, 4862, 5163, 5369
\msg_error:nnnnn	664, 684, 2284, 4023
\msg_fatal:nn	3599
\msg_fatal:nnn	562
\msg_info:nnn	13, 16, 21, 24, 399, 412
\msg_line_context:	5447, 5452, 5457, 5486, 5491, 5496, 5511, 5526, 5530, 5534, 5538, 5542, 5546, 5553, 5560, 5566, 5580, 5584, 5589, 5593, 5597, 5601, 5606, 5610, 5614, 5618, 5623, 5658, 5662, 5667, 5672, 5676, 5681, 5757, 5761, 5766, 5771, 5776, 5780, 5784, 5788, 5792, 5796, 5800, 5804, 5808
\msg_log:nnn	2088, 2093, 2098
\msg_log:nnnnn	377, 2231, 2236, 2241
\msg_log:nnnnnn	369
\msg_new:nnn	5414, 5418, 5422, 5426, 5431, 5444, 5449, 5454, 5459, 5468, 5476, 5480, 5484, 5489, 5494, 5509, 5524, 5528, 5532, 5536, 5540, 5544, 5548, 5557, 5563, 5569, 5573, 5577, 5582, 5587, 5591, 5595, 5599, 5604, 5608, 5612, 5616, 5621, 5656, 5660, 5665, 5670, 5674, 5679, 5755, 5759, 5764, 5769, 5774, 5778, 5782, 5786, 5790, 5794, 5798, 5802, 5806
\msg_new:nnnn	5435, 5626, 5635, 5644, 5650, 5683, 5693, 5703, 5713, 5723, 5733, 5743, 5749
\msg_term:nnnn	2052, 2057, 3541, 3551, 3583, 3588
\msg_term:nnnnn	2212
\msg_warning:nn	3733, 3871
\msg_warning:nnnn	2249, 2255, 3489, 3494, 4353, 4366, 4402, 4415
\msg_warning:nnnnn	2207, 2217
\multicolsep	100
\multicolsep	1310, 1482, 3681, 3844

N	
\NeedsTeXFormat	3
\NewCommandCopy	386

[\newcounter](#) 565
[\NewDocumentCommand](#) 1607, 2691, 4145, 5087, 5142, 5249, 5298, 5376
[\NewDocumentEnvironment](#) . 3763, 3897, 4088, 4570, 4834
[\newenvsc](#) 2805
[\newlabel](#) 37
[\newlabel](#) 432
[no-store](#) 2102
[\noindent](#) 3722, 4461, 4526, 4807, 5063
[\nointerlineskip](#) 1319, 1322, 1491, 1494, 1645, 1672, 4461, 4526
[noitemsep](#) 905
[\nopagebreak](#) 1256, 1284, 1319, 1322, 1491, 1494, 1598, 1604
[\normalfont](#) 2650, 3175, 3188, 4990
[nosep](#) 905

P

Packages:

[caption](#) 116
[enumext](#) 27, 39, 42, 68, 97, 107, 134, 135
[enumitem](#) 39, 40
[expl3](#) 112
[footnotehyper](#) 36
[hyperref](#) 31, 32, 36, 37, 79, 89, 121, 134
[ltxcmd](#) 35
[lua-visual-debug](#) 54
[multicol](#) 27, 134
[scontents](#) 27, 82, 83
[shortlst](#) 112, 117, 122
[\par](#) .. 1256, 1284, 1322, 1494, 1598, 1604, 1640, 1645, 1667, 1672, 2626, 3698, 3859, 3877, 4131, 4134, 4274, 4488, 4503, 4549, 4563, 4807, 5063

para commands:

[\para_end:](#) 4824, 5081
[\parbox](#) 2268
[\parindent](#) 4788, 5040
[\parsep](#) 52, 108
[\parsep](#) 965, 3574, 4066, 4075
[parsep](#) 905
[\parskip](#) 4789, 5041
[\partopsep](#) 3575, 3875, 4085
[partopsep](#) 905

peek commands:

[\peek_meaning:Ntf](#) 4674, 4688, 4703, 4714, 4918, 4932, 4947
[\peek_meaning_remove:Ntf](#) 4681, 4925
[\peek_remove_spaces:n](#) 3430
[\phantomsection](#) 36
[\phantomsection](#) 421

prg commands:

[\prg_do_nothing:](#) 425
[\prg_new_protected_conditional:Npnn](#) ... 218
[\prg_replicate:nn](#) 235
[\prg_return_false:](#) 222
[\prg_return_true:](#) 221

[\printkeyans](#) 18, 129, 5142

prop commands:

[\prop_const_from_keyval:Nn](#) 5290
[\prop_count:N](#) 371, 2400, 2551, 2653, 3065, 3178, 3191, 4993, 5392
[\prop_get:NnNtf](#) 5316
[\prop_gput_if_not_in:Nnn](#) 2398
[\prop_if_exist:Ntf](#) 2086, 5107, 5385
[\prop_item:Nn](#) 5109, 5409
[\prop_new:N](#) 2089

[\ProvidesExplPackage](#) 4

R

[\raggedcolumns](#) 3684, 3847
[\raisebox](#) 4216
[\ref](#) 77, 87
[ref](#) 691, 737, 808
[\refstepcounter](#) 4742, 5008
 regex commands:
 [\regex_match:nnTF](#) .. 220, 856, 858, 870, 872, 2898
 [\regex_replace_once:nnN](#) 228
[\renewcommand](#) 726, 770, 786, 838
[\RenewDocumentCommand](#) 451, 1655, 1682, 3261, 3288, 3304, 3426, 3458, 3472, 4099
[\RequirePackage](#) 17, 25
[resume](#) 1798
[resume*](#) 1798
[rightmargin](#) 981
[\Roman](#) 40, 45
[\Roman](#) 584
[\roman](#) 40, 45
[\roman](#) 585, 709, 5132

S

[\s](#) 2899
[save-ans](#) 2041
[save-key](#) 2339
[save-ref](#) 2261
[save-sep](#) 2261
 scan commands:
 [\scan_stop:](#) 4097, 4583, 4846, 5098, 5101
 scontents internal commands:
 [\l_scontents_fname_out_tl](#) 2858
 [__scontents_parse_environment_keys:n](#) . 2864
 [__scontents_rescan_tokens:n](#) 2871
 [\l_scontents_storing_bool](#) 2856
 [\l_scontents_writing_bool](#) 2857

seq commands:

[\seq_clear:N](#) 5251, 5394
[\seq_const_from_clist:Nn](#) 5244
[\seq_count:N](#) 372, 4280, 5255
[\seq_gclear:N](#) 449, 450
[\seq_gput_right:Nn](#) 462, 463, 2407
[\seq_if_empty:Ntf](#) 468, 5157, 5269
[\seq_if_exist:Ntf](#) 2091, 5155
[\seq_if_in:NnNtf](#) 5161
[\seq_item:Nn](#) 2896, 4267
[\seq_map_function:NN](#) 5260
[\seq_map_inline:Nn](#) 5170, 5178, 5270, 5271
[\seq_map_pairwise_function:NNN](#) 470
[\seq_new:N](#) 124, 125, 127, 142, 172, 173, 2094
[\seq_pop_left:NN](#) 5259
[\seq_put_right:Nn](#) 4159, 5267, 5283, 5404
[\seq_set_from_clist:Nn](#) 5252
[\seq_set_map_e:NNn](#) 5261
[\seq_use:Nn](#) 200, 201, 5400

[series](#) 1798

[\setcounter](#) 867, 871, 873, 3532, 3577, 4128
[\setenumext](#) 6, 130, 5249
[\setenumextmeta](#) 6, 132, 5290
[show-ans](#) 2261, 2303
[show-length](#) 1078
[show-pos](#) 2303

skip commands:

<code>\skip_add:Nn</code>	1216, 1225, 1234, 1247, 1251, 1275, 1279, 1295, 1353, 1355, 1369, 1372, 1393, 1395, 1409, 1412, 1432, 1434, 1448, 1451, 1470, 1519, 1520, 1531, 1533, 4075, 4083
<code>\skip_gset:Nn</code>	1546, 1550, 1554
<code>\skip_gzero_new:N</code>	1541, 1542
<code>\skip_horizontal:N</code>	1049, 1061, 1073, 4760, 4772, 4811, 5024, 5067
<code>\skip_horizontal:n</code>	1035, 2475, 2483, 3254, 3256, 4659, 4758, 4792, 4903, 5044
<code>\skip_if_eq:nnTF</code>	1214, 1223, 1232, 1339, 1379, 1419, 1507, 1543, 1565, 1706, 1720, 1734, 1745, 1756, 1767, 1778, 1789
<code>\skip_new:N</code>	81, 82, 83, 88, 89, 90, 91, 92, 93, 148, 192
<code>\skip_set:Nn</code>	1199, 1203, 1261, 1265, 1289, 1342, 1343, 1361, 1382, 1383, 1401, 1421, 1422, 1440, 1464, 1510, 1511, 1525, 1545, 1549, 1567, 1571, 1575, 1581, 1585, 1589, 4059
<code>\skip_set_eq:NN</code>	1300, 1301, 1303, 1310, 1475, 1476, 1477, 1482, 3530, 3573, 3574, 4789, 5041
<code>\skip_sub:Nn</code>	1349, 1351, 1365, 1367, 1389, 1391, 1405, 1407, 1428, 1430, 1444, 1446, 1517, 1518, 1529, 1530
<code>\skip_use:N</code>	1201, 1205, 1249, 1253, 1257, 1277, 1281, 1291, 1297, 1707, 1711, 1714, 1721, 1725, 1728, 3698
<code>\skip_vertical:N</code>	549, 552, 973, 4487, 4501, 4826, 5083
<code>\skip_vertical:n</code>	972, 4825, 5082
<code>\skip_zero:N</code>	1309, 1323, 1461, 1462, 1463, 1481, 1495, 3575, 3681, 3844, 4085, 4086
<code>\skip_zero_new:N</code>	1540, 1562, 1563, 1564
<code>\c_zero_skip</code>	549, 552, 973, 1214, 1223, 1232, 1380, 1419, 1543, 1565, 1707, 1721, 1734, 1745, 1756, 1767, 1778, 1789, 4487, 4501, 4826, 5083
<code>\small</code>	5120, 5124, 5128, 5132, 5136, 5140
socket commands:	
<code>\socket_assign_plug:nn</code>	3943, 3951, 3959, 3993, 4001, 4009
<code>\socket_new:nn</code>	3915, 3963
<code>\socket_new_plug:nnn</code>	3916, 3923, 3931, 3964, 3971, 3980
<code>\socket_use:n</code>	3944, 3994
<code>\socket_use:nn</code>	3952, 3960, 4002, 4010
<code>start</code>	879
<code>start*</code>	879
<code>start-list-tags</code>	3915, 3963
<code>\stepcounter</code>	455, 4069, 4209
<code>stop-list-tags</code>	3915, 3963
<code>stop-start-tags</code>	3915, 3963
str commands:	
<code>\c_backslash_str</code>	2744, 5447, 5452, 5457, 5462, 5464, 5466, 5471, 5473, 5571, 5575, 5579, 5589, 5593, 5601, 5602, 5606, 5618, 5619, 5623, 5624, 5645, 5647, 5651, 5653, 5681, 5744, 5746, 5750, 5752, 5761, 5762, 5766, 5771, 5772, 5776, 5780, 5784
<code>\c_colon_str</code>	2550, 3064, 5098
<code>\c_left_brace_str</code>	5552, 5559, 5565
<code>\c_right_brace_str</code>	5552, 5559, 5565
<code>\str_case:nn</code>	256, 315
<code>\str_case:nnTF</code>	1821, 1829, 2378, 2386, 5198, 5207
<code>\str_clear:N</code>	3608, 4615
<code>\str_count:n</code>	235
<code>\str_if_empty:NTF</code>	1838, 1879, 1906
<code>\str_if_eq:nnTF</code>	3533, 3579, 5300
<code>\str_if_in:nnTF</code>	5094

<code>\str_new:N</code>	84, 132, 147, 187
<code>\str_set:Nn</code>	649, 655, 661, 680, 681, 682, 2281, 2282, 2308, 2309, 4018, 4021
<code>\str_use:N</code>	3308
<code>\strutbox</code>	1328, 1331, 1342, 1343, 1354, 1356, 1371, 1374, 1382, 1383, 1394, 1396, 1411, 1414, 1421, 1422, 1433, 1435, 1450, 1453, 1499, 1502, 1510, 1511, 1519, 1520, 1532, 1534, 1545, 1546, 1549, 1556, 1569, 1577, 1583, 1591, 4078, 4083, 4131, 4139, 4222

T

tag commands:

<code>\tag_mc_begin:n</code>	3921, 3969, 3978
<code>\tag_mc_end:</code>	3925, 3973, 3982
<code>\tag_resume:n</code>	3918, 3966, 4105, 4113, 4179, 4271, 4471, 4535
<code>\tag_struct_begin:n</code>	3919, 3920, 3927, 3928, 3929, 3967, 3968, 3975, 3976, 3977, 4114
<code>\tag_struct_end:</code>	4124, 4125
<code>\tag_struct_end:n</code>	3926, 3933, 3934, 3935, 3936, 3974, 3983, 3984, 3985, 3986, 4589, 4852
<code>\tag_suspend:n</code>	3937, 3987, 4095, 4107, 4119, 4170, 4263, 4581, 4844
<code>\tag_tool:n</code>	4106

TeX and L^AT_EX 2_ε commands:

<code>\@auxout</code>	430
<code>\@currentenv</code>	256, 315
<code>\protected@write</code>	430

tex commands:

<code>\tex_newlinechar:D</code>	2870
---------------------------------	------

text commands:

<code>\text_expand:n</code>	5090
<code>\textasteriskcentered</code>	2278, 3327
<code>\textreferencemark</code>	2295
<code>\the</code>	244, 249
<code>\thepage</code>	436

tl commands:

<code>\c_space_tl</code>	3146, 5496, 5511, 5534, 5538, 5725, 5726, 5735, 5736, 5796, 5800
<code>\tl_clear:N</code>	647, 654, 2259, 2325, 2335, 2356, 2364, 2570, 2890, 2891, 3005, 3081, 4953
<code>\tl_clear_new:N</code>	594
<code>\tl_const:Nn</code>	50, 578
<code>\tl_gclear:N</code>	363, 364, 365, 1859, 1864, 2980, 3299, 3317, 4507, 4567, 4761
<code>\tl_gclear_new:N</code>	1846
<code>\tl_gput_right:Nn</code>	579
<code>\tl_greplace_all:Nnn</code>	600
<code>\tl_gset:Nn</code>	291, 292, 306, 307, 1847, 1860, 1865, 2084, 2894, 3230, 4709
<code>\tl_gset_eq:NN</code>	596, 3226, 4754
<code>\tl_if_blank:nTF</code>	2683, 2701, 2838, 3351, 3369, 3391, 4752, 5364
<code>\tl_if_empty:NTF</code>	714, 732, 760, 776, 795, 802, 828, 844, 1872, 1877, 1899, 1904, 1962, 2026, 2034, 2063, 2123, 2414, 2445, 2590, 2935, 2957, 2987, 3015, 3091, 3140, 3251, 4278, 4956, 5281
<code>\tl_if_empty:nTF</code>	1927
<code>\tl_if_exist:NTF</code>	1932
<code>\tl_if_novalue:nTF</code>	453, 2697, 3013, 3089, 3125, 3205, 3224, 3232, 3401, 3606, 4050, 4613, 4884, 4954
<code>\tl_map_inline:Nn</code>	226, 597
<code>\tl_new:N</code>	42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 105, 106, 107, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 126, 128, 129, 130, 133, 136, 137, 155, 163, 164, 165, 168, 186

156 / 156