

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-09*

©2024 by Pablo González†

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

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

Abstract

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

Contents

1	Introduction	1	5	The storage system	10
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for <code>label</code> and <code>ref</code>	11
1.3	User interface	3	5.1.2	Keys for <code>wrap</code> and <code>display</code>	11
1.3.1	Internal counters	3	5.1.3	Keys for <code>debug</code> and <code>checking</code>	11
1.3.2	Support for <code>multicol</code>	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>minipage</code>	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	13
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
2	The environments provided	4	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
3	The command <code>\setenumext</code>	6	6	Full examples	17
4	The <code>keyval</code> system	6	7	The way of non-enumerated lists	19
4.1	Keys for <code>label</code> and <code>ref</code>	6	8	References	21
4.2	Keys for spaces	7	9	Change history	22
4.2.1	Vertical spaces	7	10	Index of Documentation	23
4.2.2	Horizontal spaces	8	11	Implementation	25
4.3	Keys for <code>add code</code>	8	12	Index of Implementation	123
4.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	9			
4.5	Keys for <code>multicols</code>	9			
4.6	Keys for <code>minipage</code>	9			
4.6.1	The command <code>\miniright</code>	10			
4.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

*This file describes a documentation for v1.0, last revised 2024-06-09.

†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 (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.txt>). The software has the status “maintained”.
The enumext package loads and requires multicol[3] and scontents[4] packages, need to have a modern T_EX distribution such as T_EX Live or MiK_TE_X. It has been tested with the standard classes provided by L^AT_EX: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the L^AT_EX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[1] and packages like xsim[2] do the job perfectly, but they don’t always fit the basic day to day needs.
In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

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

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

3. True False

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

* $(x - 1)^2$

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

* $3(x + y + z)$

3. True False

(a) $\alpha > \delta$

* False

(b) L^AT_EXze is cool?

* Very True!

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

* Yes

(b) Usually uses the package manager?

* Yes, dnf

(c) Rate the following package and class

i. xsim-exam

* doesn't exist for now :(

ii. xsim

* very good

iii. exsheets

* obsolete

Or we are interested in referring to a specific question and its “answer”, for example:
The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.
Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

ii. very good

iii. obsolete

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

E) value

4. Question with image and label below:

A

B

A

A) B) C)

A

duck

D) E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

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

2 / 136

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 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.

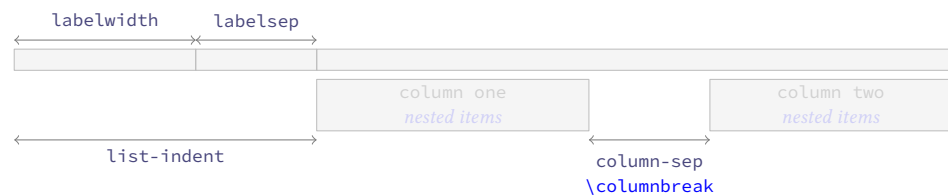


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §4.5).

- 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.

1.3.3 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 §4.6).

1.3.4 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 (§4.1), 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` (§5.1.1) when the key `save-ans` (§5.1) is active.

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

1.3.5 Support for \footnote

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

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

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

2 The environments provided

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

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

2.1 The environment `enumext`

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

Example with `columns=2`

1. This text is in the first level.

(a) This text is in the second level.

i. This text is in the third level.
- A. This text is in the fourth level.

X This text is in the first 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, but it can be nested within `enumext` and can contain it nested within it.
- Each “item” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that includes `labelwidth`, `labelsep` plus the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.

Example with `columns=2`

1. This text is in the first level.

X This text is in the first level.
2. This text is in the first level.

★ 3. This text is in the first level.

2.3 The command `\item*`

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

```
\begin{enumext*}[widest=10,columns=4]
  \item The first
  \item* The second
  \item The third
  \item The fourth
  \item(3)* The fifth item is way too long for this and needs three columns
  \item The sixth
  \item the seventh
  \item(2)[X] The eighth item is way too long for this and needs two columns
  \item[Z] The ninth
  \item The tenth
\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 | The eighth item is way too long for this and needs Z | The ninth |
| | two columns | | |
| 8. The tenth | | | |

3 The command `\setenumext`

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

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

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

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

4 The 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.

4.1 Keys for label and ref

`label = {⟨\alph* | \Alph* | \arabic* | \roman* | \Roman*⟩}` default: *by levels*

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

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

`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`.

`labelsep = {⟨rigid length⟩}` default: *0.3333em*

Sets the *horizontal space* between the box containing the current `⟨label⟩` 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 = {⟨rigid length⟩}` default: *by label*

Sets the *width* of the box containing the current `⟨label⟩` 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 = {⟨integer | string⟩}` default: *empty*

Sets the `labelwidth` key pass the `⟨integer⟩` or converting the `⟨string⟩` 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 = {⟨font commands⟩}` default: *empty*

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

`align = {⟨left | right | center⟩}` default: *left*

Sets the *aligned* of `⟨label⟩` defined by `label` key on the current level in the label box.

`wrap-label = {⟨code {#1} more code⟩}` default: *empty*

Wraps the *current* `⟨label⟩` defined by `label` key referenced by `{#1}`. The `⟨code⟩` 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 \itembx { 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={\itembx{#1}}` or `wrap-label={\itembx*{#1}}`.

`wrap-label* = {⟨code {#1} more code⟩}` default: *empty*

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

4.2 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`.

4.2.1 Vertical spaces

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

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

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

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

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

Set the *vertical space* added, beyond `topsep`, to the “top” and “bottom” of the entire environment if the environment instance is preceded by a “blank line” or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level in environment `enumext` are 2.0pt plus 1.0pt minus 1.0pt, for third and fourth level are 1.0pt minus 1.0pt. For `keyans`, `keyans*` and `enumext*` environments the default value is 2.0pt plus 1.0pt minus 1.0pt.

- 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 \langle vertical mode \rangle and apply this value to the “top” and “bottom” the environment or nested level.

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

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

`noitemsep` \langle value forbidden \rangle default: *not used*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2.2 Horizontal spaces

- `itemindent` = { $\langle rigid\ length \rangle$ } default: 0pt
 Extra *horizontal indentation*, beyond `labelsep`, of the “*first line*” off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.
- `rightmargin` = { $\langle rigid\ length \rangle$ } default: 0pt
 Set the *horizontal space* between the right margin of the environment and the right margin of the enclosing environment, the value it takes must be greater than or equal to 0pt. Internally sets the value of `\rightmargin` for the current level.
- `listparindent` = { $\langle rigid\ length \rangle$ } default: 0pt
 Sets the *horizontal space* indentation, beyond `list-indent`, for second and subsequent paragraphs within a list item. Internally sets the value of `\listparindent` for the current level.
- `list-offset` = { $\langle rigid\ length \rangle$ } default: 0pt
 Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.
- `list-indent` = { $\langle rigid\ length \rangle$ } default: labelwidth + labelsep
 Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.
- If `list-indent=0pt` is set in the environment `enumext` the $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

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

4.3 Keys for add code

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

- `before` = { $\langle code \rangle$ } default: not used
 Execute $\{\langle code \rangle\}$ “*before*” the environment starts. The $\{\langle code \rangle\}$ must be passed between braces, is executed “*after*” performing all calculations related to the *list parameters* in the environment and the parameters sets by $[\langle key = val \rangle]$ that is, in the second argument of the list after setting all the parameters `\list{\langle arg one \rangle}{\langle arg two \rangle}{\langle code \rangle}`.
- `before*` = { $\langle code \rangle$ } default: not used
 Execute $\{\langle code \rangle\}$ “*before*” the environment starts. The $\{\langle code \rangle\}$ must be passed between braces, is executed “*before*” performing all calculations related to the *list parameters* and $[\langle key = val \rangle]$ sets in the environment that is, before the arguments defining the environment are executed: $\{\langle code \rangle\}\list{\langle arg one \rangle}{\langle arg two \rangle}$.
- `first` = { $\langle code \rangle$ } default: not used
 Executes $\{\langle code \rangle\}$ when “*starting*” the environment. The $\{\langle code \rangle\}$ must be passed between braces, is executed right “*after*” all *list parameters* are done, after the second argument of list, just before the first occurrence of `\item: \list{\langle arg one \rangle}{\langle arg two \rangle}{\langle code \rangle}\item`.
- Keep in mind that the code set in this key will affect the entire “*body*” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.
- `after` = { $\langle code \rangle$ } default: not used
 Execute $\{\langle code \rangle\}$ “*after*” finishing the environment. The $\{\langle code \rangle\}$ must be passed between braces.

4.4 Keys for start, series and resume

- `start` = { $\langle integer \mid string \rangle$ } default: 1
 Sets the *start value* of the numbering on the current level. Internally $\langle string \rangle$ is passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.
- The following $\langle keys \rangle$ are “*only*” available for the “*first level*” of `enumext` and `enumext*` and are ignored if set when nested inside each other.
- `series` = { $\langle series\ name \rangle$ } default: not used
 Stores the *keys* of the optional argument of the “*first level*” of the environment in which it is executed in $\{\langle series\ name \rangle\}$ which is used as an argument in the key `resume`. The $\langle keys \rangle$ stored in $\{\langle series\ name \rangle\}$ are not cumulative and are overwritten if the same $\{\langle series\ name \rangle\}$ is used again.
- `resume` = { $\langle series\ name \rangle$ } default: not used
 Sets the *start value* and *options* for the “*first level*” continuing the numbering of the environment in which the `series=\{\langle series\ name \rangle\}` key was executed. If passed *without value* this will only set *start value* continue the numbering from the last environment in which `series=\{\langle series\ name \rangle\}` or `resume=\{\langle series\ name \rangle\}` 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 the `start` key.

`resume*` $\langle \text{value forbidden} \rangle$ 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 the `start` key.

- For security reasons the `series` key will never save in $\langle \text{series name} \rangle$ the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` and `start`. When using the key `resume={\series name}` it will have hierarchy in the $\langle \text{keys} \rangle$ that are saved in $\langle \text{series name} \rangle$, in order to establish the value of a $\langle \text{key} \rangle$ already saved in $\langle \text{series name} \rangle$ 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*.

4.5 Keys for multicol

`columns` = $\langle \text{integer} \rangle$ 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` = $\langle \text{rigid length} \rangle$ 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.

4.6 Keys for minipage

`mini-env` = $\langle \text{rigid length} \rangle$ 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` = $\langle \text{rigid length} \rangle$ 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`.

4.6.1 The command \miniright

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

The `\miniright` command close the `minipage` environment on the “left side” and opens the `minipage` environment on the “right side” by starting it with the `\centering` command. It must be placed “after” the last `\item` of the current environment and “before” starting the material to be placed on the “right side”. The *starred argument* “*” inhibits the use of `\centering` command i.e. the usual L^AT_EX justification is maintained in the `minipage` on the “right side”.

- The `\footnote{\text}` command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark[\number]` inside the environment and `\footnotetext[\number]{\text}` outside the environment or via the `after` key (see §1.3.5 for full support).

4.6.2 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` = $\langle \text{content} \rangle$ 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 $\langle \text{content} \rangle$ must be passed between braces.

`mini-right*` = $\langle \text{code for drawing or tabular} \rangle$ default: *not used*

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

- The keys `mini-right` and `mini-right*` has a *slightly different* implementation, the argument $\langle \text{content} \rangle$ is saved in a box and then printed outside the environment using *hooks*.

5 The storage system

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

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

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

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

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

5.1 Keys for storage system

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

`save-ans = {\langle store name \rangle}` default: *not set*

Sets the *name* of the $\langle sequence \rangle$ and $\langle prop list \rangle$ in which the contents will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the $\langle sequence \rangle$ or $\langle prop list \rangle$ does not exist, it will be created globally and will not be overwritten if the key is used again.

`save-key = {\langle key list \rangle}` default: *not set*

This key *overrides* the default “*stored keys*” of the optional arguments of the inner levels or nested environment that will be passed to the $\langle sequence \rangle$. The $\langle key list \rangle$ passed to this key ignores any $\langle keys \rangle$ in the “*stored system*” and must be passed between braces. For example, if we execute at a second level:

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

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

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

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

5.1.1 Keys for label and ref

`save-ref = {\langle true | false \rangle}` default: *false*

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

`mark-ref = {\langle symbol \rangle}` default: `\textasteriskcentered`

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

5.1.2 Keys for wrap and display

- `wrap-ans` = {`{code {#1} more code}`} default: `\fbox{#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.

5.1.3 Keys for debug and checking

- `show-pos` = {`{true | false}`} default: `false`
 Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* `{store name}` set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.
- `check-ans` = {`{true | false}`} default: `false`
 Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “only one answer” or “only one execution” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.
- `no-store` *{value forbidden}* default: `not used`
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* `{store name}` set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey`, “without” use `anskey*`, “without” interfering with the `check-ans` key and “without” storing an unwanted structure in the *sequence* `{store name}`.

5.2 The command `\anskey`

`\anskey` `\anskey[keys]{content}`

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

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

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

5.2.1 Keys for `\anskey`

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

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

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

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

Example

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

- | | |
|--|---|
| <ul style="list-style-type: none"> ★ 1. Text containing our instructions or questions. ★ first answer 2. Text containing our instructions or questions. (a) Question. ★ second answer | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. ★ third answer 4. Text containing our instructions or questions. ★ fourth answer |
|--|---|

5.3 The environment `anskey*`

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

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

By design the environment cannot be nested but full supports “*verbatim material*” in the body and it is assumed that each numbered `\item` or `\item*` within the environment in which it is active it has a “*single execution*” unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the `scontents` package, for the correct operation `\begin{anskey*}` and `\end{anskey*}` must be in different lines, all $\langle keys \rangle$ must be passed separated by commas and “without separation” of the start of the environment. Comments “%” or “any character” after `\begin{anskey*}` or `[\langle key = val \rangle]` on the same line are NOT supported, the package `scontents` will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskey*}` on the same line the package `scontents` will return a “warning” message.

The `anskey*` environment uses the same $\langle keys \rangle$ as the `\anskey` command next to the keys `write-env`, `force-eol` and `overwrite` inherited from package `scontents`. The environment and is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

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

Example

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



```

\begin{anskey*}
  \langle fourth answer \rangle
\end{anskey*}
\end{enumext}

```

- | | |
|---|--|
| <p>★ 5. Text containing our instructions or questions.</p> <p>[5] First answer with verbatim</p> <p>6. Text containing our instructions or questions.</p> <p style="padding-left: 20px;">(a) Question.</p> <p style="padding-left: 20px;">[6] second answer</p> | <p>7. Text containing our instructions or questions.</p> <p>[7] third answer</p> <p>8. Text containing our instructions or questions.</p> <p>[8] fourth answer</p> |
|---|--|

5.4 The environments `keyans` and `keyans*`

```

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

```

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

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

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

5.4.1 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 `\langle content \rangle` (if it is present) in *sequence* and *prop list* `{\langle store name \rangle}` set by `save-ans` key in the “*first level*” of the `enumext` or `enumext*` environments.

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

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

Example

```

\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
  \begin{keyans*}[nosep,columns=2]
    \item Choice
    \item* Correct choice
    \item Choice
    \item Choice
    \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}

```



```
\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

5.5 The environment keyanspic

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

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

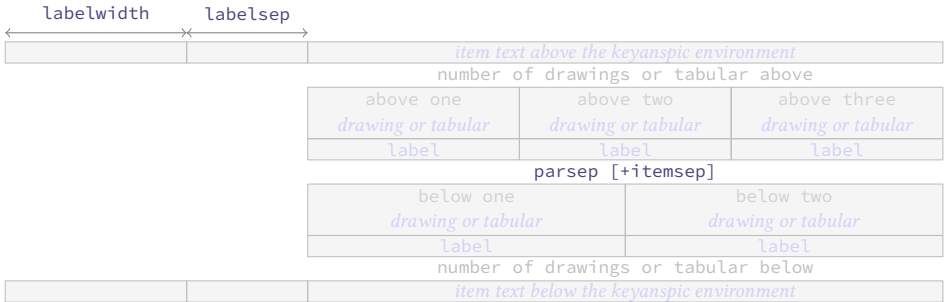


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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

5.5.1 The command \anspic

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

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

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

Example

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

1. Question with images.


A)


B)


C)


D)


* E)[note]

5.6 Printing stored content

5.6.1 The command `\getkeyans`

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

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

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

5.6.2 The command `\printkeyans`

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

The command `\printkeyans` prints “all stored content” in *sequence* `{<store name>}` defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred argument* ‘*’ is used. The “stored content” can only be accessed *after* it is stored in the *sequence*, if `{<store name>}` does not exist the command will return an error.

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

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

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

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

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[<print>,<1>]{<keys>}` and `\setenumext[<print*>]{<keys>}`. If we need to set the `<keys>` for the environment `enumext` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print>,<level>]{<keys>}` and if we need to set the `<keys>` for the environment `enumext*` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print>,<*>]{<keys>}`.

Example

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

  \begin{enumext}[nosep]
    \item \LaTeX2e\ 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}
```

```
\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
- *

*

*

*

*

6 Full examples

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

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

and then you can use the excellent arara¹ tool to compile them.

Example 1

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

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

36 km/h.

360 km/h.

27,8 km/h.

$3,60 \times 10^8$ 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?

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

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

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

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

36 km/h.

360 km/h.

27,8 km/h.

$3,60 \times 10^8$ 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?

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

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

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

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

2. A

3. B

4. A

Example 2

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

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

36 km/h.

360 km/h.

27,8 km/h.

$3,60 \times 10^8$ 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?

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

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

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

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

- D

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

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

✓

A

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

B

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

C

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

D

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

2. A

3. B

4. A
- *

*

*

*

Example 3

A “simple multiple choice” test .

1. First type of questions

A

value

B

correct

C

value

D

value
2. Second type of questions

I.

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

II.

$\alpha = \delta$

III.

$\angle EDF = 45^\circ$

A

I only

B

II only

C

I and II only

D

I and III only

E

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

(1)

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

(2)

$\angle EDF = 45^\circ$

A

value

B

value

C

value

D

value

E

value
4. Question with image and label below:

A

A

D

B

B

A

C



E

B

5. Question with image on left side:

A

value

B

value

C

value

D

correct

E

value

Test keys

1. B, $x = 5$

2. D

3. C, some note

*


4. E, A duck


*

5. D, other note


*

Example 4

A “simple worksheet” using ducks :) .



Factor $x^2 - 2x + 1$



Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

©2024 by Pablo González L

18 / 136



True False

- (a) $\alpha > \delta$
- (b) $\mathbb{E}\mathbb{T}\mathbb{E}\mathbb{X}$ ze is cool?



Related to Linux

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

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

- | | | | |
|-------------------|---|---------------------------------|---|
| 1. $(x - 1)^2$ | * | (b) Yes, dnf | * |
| 2. $3(x + y + z)$ | * | (c) i. doesn't exist for now :(| * |
| 3. (a) False | * | ii. very good | * |
| (b) Very True! | * | iii. obsolete | * |
| 4. (a) Yes | * | | |

Example 5

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

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

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

7 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 $\mathbb{E}\mathbb{T}\mathbb{E}\mathbb{X}$ for the four levels `\textbullet`, `\textendash`, `\textasteriskcentered` and `\textperiodcentered` together with the `nosep` key to reduce the vertical spaces in the left side example and set the `label` key in *mathematical mode* for the right side as `\ast`, `\diamond`, `\circ` and `\star` for the four levels together with the `nosep` key

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

Fake description environment

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

SomeThing A short one-line description.
This is an entry *without* a label.
Something A short *one-line* description text.
Something long A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

SomeThing A short one-line description.
This is an entry *without* a label.
Something A short *one-line* description text.
Something long A much *longer* description text may take more than one line or more than one paragraph.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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


```
\NewDocumentCommand \itembx { 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*={\itembx{#1}}`.

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short one-line description.

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

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

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

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

LoNg 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.

8 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2024.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2022.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ $\text{\LaTeX} 2_{\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.

- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The \LaTeX 2_ε sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.
- [13] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2024.
- [14] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [15] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [16] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.

9 Change history

v1.0 2024-06-09 – First public release.

10 Index of Documentation

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

C	
Document class:	
article	2
book	2
exam	2
letter	2
report	2
\columnbreak	4, 12
\columnsep	10
Commands provide by enumext:	
\anskey	11–13
\anspic	11, 12, 15
\getkeyans	12, 16
\item*	5–7, 11, 12, 14
\itemwidth	5
\item	5–7, 9, 10, 12, 14
\miniright	10
\printkeyans	6, 11, 16
\setenumext	5–7, 11, 12, 14, 16
Counters defined by enumext:	
enumXiii	4
enumXii	4
enumXiv	4
enumXi	4
enumXviii	4
enumXvii	4
enumXvi	4
enumXv	4
E	
Environments provide by enumext:	
anskey*	11–13
enumext*	4–14, 16
enumext	4–9, 11–14, 16, 19
keyans*	4–14
keyanspic	4, 6, 8, 11–13, 15, 19
keyans	4–9, 11–15, 19
Environments:	
enumerate	1, 3, 5, 21
figure	5
list	3, 9, 21
minipage	3–5, 10, 21
multicols	3, 4, 10
table	5
task	5
F	
\footnote	5
I	
\itemsep	8
K	
Keys for command provide by enumext:	
break-col	12
item-join	12
item-pos*	13
item-star	12, 13
item-sym*	13
Keys for environments provide by enumext:	
above*	8
above	8
after	9, 10
align	7, 20
base-fix	8
before*	9
before	9
below*	8
below	8
check-ans	12
columns-sep	4, 10
columns	4, 8, 10
first	9
font	7
item-pos*	5, 6
item-sym*	5, 6
itemindent	9
itemsep	8, 15
labelsep	3, 5–7, 9, 10, 12, 20
labelwidth	3, 6, 7, 9, 10, 12, 20
labelwith	5
label	7, 9, 14, 19, 20
list-indent	3, 9
list-offset	3, 9, 20
listparindent	9
mark-ans	12
mark-pos	12
mark-ref	11
mini-env	4, 8, 10
mini-right*	6, 10
mini-right	6, 10
mini-sep	4, 10
no-store	11–13
noitemsep	8
nosep	8, 19
parsep	8, 15
partopsep	8
ref	4, 7
resume*	6, 10, 11
resume	6, 9–11
rightmargin	9
save-ans	4, 6, 9–16
save-key	10, 11, 16
save-ref	4, 7, 11–13, 16
save-sep	11
series	6, 9–11
show-ans	11, 12
show-length	7
show-pos	11, 12, 16
start	9, 10
topsep	8
widest	7
wrap-ans	12
wrap-label*	7, 20
wrap-label	7
wrap-opt	12
L	
\label	4
Labels provide by enumext:	
\Alph*	7, 14
\Roman*	7
\alph*	7

\arabic*	7	l3seq	1, 21
\roman*	7	multicol	1, 2, 4, 21
\labelsep	3, 7	scontents	1, 2, 13
\labelwidth	3, 7	task	5, 6
\linewidth	10	xsim	2
\listparindent	9	\parsep	8
P		\partopsep	8
Packages:			
enumerate	21	R	
enumext	1-6, 15, 21	\raggedcolumns	4
enumitem	3-5, 9, 20, 21	\ref	4
footnotehyper	4, 5	\rightmargin	9
hyperref	4, 5, 11-13, 21	T	
l3keys	6	\topsep	8
l3prop	1, 21		

11 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 \TeX pert are superfluous, but, after so many years developing this project is the only way to remember what does what.

11.1 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.

11.2 Initial set up

Start the DocStrip guards.

```
1 < *package >
```

Identify the internal prefix (\LaTeX 3 DocStrip convention) for `l3doc` class.

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

11.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of \LaTeX to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-09}
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 }
```

11.4 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
```

Internal variables used by functions `__enumext_is_not_nested:`, `__enumext_is_on_first_level:` and `__enumext_keyans_start_line:` (§11.5.1).

```
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
```

(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` (§11.9) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§11.12).

```
44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \tl_new:c { l__enumext_counter_#1_tl }
47 }
48 \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 (§11.12).

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

Internal variables used by `resume`, `resume*` and `series` keys (§11.23).

```
60 \int_new:N \g__enumext_resume_int
61 \int_new:N \g__enumext_resume_vii_int
62 \tl_new:N \l__enumext_resume_name_tl
63 \bool_new:N \l__enumext_resume_active_bool
64 \tl_new:N \g__enumext_standar_series_tl
65 \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` (§11.13) and `label` (§11.11) keys.

```

66 \dim_new:N \l__enumext_current_widest_dim
67 \tl_new:N \g__enumext_counter_styles_tl
68 \tl_new:N \g__enumext_widest_label_tl
69 \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 (§11.16). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§11.34.1).

```

70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
73   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
74   \dim_new:c { \l__enumext_leftmargin_#1_dim }
75   \dim_new:c { \l__enumext_itemindent_#1_dim }
76 }
77 \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

```

Internal variables used by `columns` key §11.20).

```

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

```

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

```

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

```

Internal variables used by `\miniright` command (§11.21.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.19, §11.21).

```

84 \int_new:N \g__enumext_minipage_stat_int
85 \skip_new:N \l__enumext_minipage_left_skip
86 \skip_new:N \l__enumext_minipage_right_skip
87 \skip_new:N \l__enumext_minipage_after_skip
88 \skip_new:N \g__enumext_minipage_right_skip
89 \skip_new:N \g__enumext_minipage_after_skip
90 \cs_set_protected:Npn \__enumext_tmp:n #1
91 {
92   \dim_new:c { \l__enumext_minipage_left_#1_dim }
93   \bool_new:c { \l__enumext_minipage_active_#1_bool }
94 }
95 \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 integer variable `\l__enumext_start_X_int` are used by the `start` key (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§11.16.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.11). 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 (§11.18).

```

96 \cs_set_protected:Npn \__enumext_tmp:n #1
97 {
98   \bool_new:c { \l__enumext_wrap_label_#1_bool }
99   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
100   \int_new:c { \l__enumext_start_#1_int }
101   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
103   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
104   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
105   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
106 }
107 \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 (§11.24.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§11.27) and the variables `\l__enumext_store_anskey_env_tl`, `\l__enumext_store_anskey_opt_tl` save the `⟨body⟩` and `⟨keys⟩` of the environment `anskey*` (§11.28).

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*` (§11.32.2) and `\anspic*` (§11.37.1) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```
108 \bool_new:N \l__enumext_store_active_bool
109 \tl_new:N \l__enumext_store_name_tl
110 \tl_new:N \g__enumext_store_name_tl
111 \tl_new:N \l__enumext_store_anskey_arg_tl
112 \tl_new:N \l__enumext_store_anskey_env_tl
113 \tl_new:N \l__enumext_store_anskey_opt_tl
114 \tl_new:N \l__enumext_store_current_label_tl
115 \tl_new:N \l__enumext_store_current_opt_arg_tl
116 \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` (§11.43).

```
117 \tl_new:N \l__enumext_setkey_tmpa_tl
118 \tl_new:N \l__enumext_setkey_tmpb_tl
119 \int_new:N \l__enumext_setkey_tmpa_int
120 \seq_new:N \l__enumext_setkey_tmpa_seq
121 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

```
\l__enumext_print_keyans_starred_tl
\l__enumext_mark_position_str
\g__enumext_item_symbol_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` (§11.42), `show-pos` key (§11.26), `item-sym*` key (§11.30), `save-key` key (§11.26.2) and “*storage level system*”.

```
122 \tl_new:N \l__enumext_print_keyans_starred_tl
123 \str_new:N \l__enumext_mark_position_str
124 \tl_new:N \g__enumext_item_symbol_tl
125 \cs_set_protected:Npn \__enumext_tmp:n #1
126 {
127   \tl_new:c { \l__enumext_print_keyans_#1_tl }
128   \tl_new:c { \l__enumext_store_save_key_#1_tl }
129   \bool_new:c { \l__enumext_store_save_key_#1_bool }
130   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
131 }
132 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

```
\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip
```

Internal variables used by `keyanspic` environment (§11.37.2).

```
133 \seq_new:N \l__enumext_keyans_pic_body_seq
134 \dim_new:N \l__enumext_keyans_pic_width_dim
135 \int_new:N \l__enumext_keyans_pic_above_int
136 \int_new:N \l__enumext_keyans_pic_below_int
137 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

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

Internal variables used by “*internal check answer*” mechanism (§11.24.3) 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.

```
138 \bool_new:N \l__enumext_check_answers_bool
139 \bool_new:N \g__enumext_check_ans_key_bool
140 \tl_new:N \l__enumext_check_start_line_env_tl
141 \int_new:N \g__enumext_check_starred_cmd_int
142 \int_new:N \g__enumext_item_anskey_int
```

```

143 \int_new:N \g__enumext_item_number_int
144 \bool_new:N \l__enumext_item_number_bool
145 \int_new:N \g__enumext_item_answer_diff_int
146 \int_new:N \l__enumext_item_answer_diff_int

```

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

```

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

```

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§11.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

147 \bool_new:N \l__enumext_hyperref_bool
148 \bool_new:N \l__enumext_footnotes_key_bool

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

Internal variables used by `save-ref` key (§11.26). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the *labels* defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§11.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

149 \tl_new:N \l__enumext_newlabel_arg_one_tl
150 \tl_new:N \l__enumext_newlabel_arg_two_tl
151 \tl_new:N \l__enumext_write_aux_file_tl
152 \cs_set_protected:Npn \__enumext_tmp:n #1
153 {
154   \tl_new:c { \l__enumext_label_copy_#1_tl }
155 }
156 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

Internal variables used for redefinition of `\footnote` (§11.31).

```

157 \int_new:N \g__enumext_footnote_int
158 \seq_new:N \g__enumext_footnote_arg_seq
159 \seq_new:N \g__enumext_footnote_int_seq

```

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

```

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

```

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

```

160 \cs_set_protected:Npn \__enumext_tmp:n #1
161 {
162   \bool_new:c { \l__enumext_item_starred_#1_bool }
163   \int_new:c { \l__enumext_item_column_pos_#1_int }
164   \int_new:c { \g__enumext_item_count_all_#1_int }
165   \int_new:c { \l__enumext_joined_item_#1_int }
166   \int_new:c { \l__enumext_joined_item_aux_#1_int }
167   \int_new:c { \l__enumext_tmpa_#1_int }
168   \box_new:c { \l__enumext_item_text_#1_box }
169   \dim_new:c { \l__enumext_joined_width_#1_dim }
170   \dim_new:c { \l__enumext_item_width_#1_dim }
171   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
172   \str_new:c { \l__enumext_align_label_#1_str }
173   \bool_new:c { \g__enumext_minipage_active_#1_bool }
174   \box_new:c { \l__enumext_miniright_code_#1_box }
175   \bool_new:c { \g__enumext_minipage_center_#1_bool }
176   \dim_new:c { \g__enumext_minipage_right_#1_dim }
177   \skip_new:c { \g__enumext_minipage_right_#1_skip }
178 }
179 \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`.

```

180 \clist_const:Nn \c__enumext_all_envs_clist
181 {
182   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
183   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
184 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

11.5 Some utility functions

`__enumext_at_begin_document:n`

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

```
185 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
186 {
187   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
188 }
```

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

`__enumext_after_env:nn`
`__enumext_before_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.

```
189 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
190 {
191   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
192 }
193 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
194 {
195   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
196 }
```

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

`__enumext_level:`

Function for check current level in `enumext`.

```
197 \cs_new:Nn \__enumext_level:
198 {
199   \int_to_roman:n { \l__enumext_level_int }
200 }
```

(End of definition for `__enumext_level:.`)

`__enumext_if_is_int:nT`
`__enumext_if_is_int:nF`
`__enumext_if_is_int:nTF`

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

```
201 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
202 {
203   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
204   { \prg_return_true: }
205   { \prg_return_false: }
206 }
```

(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.

```
207 \cs_new_protected:Nn \__enumext_regex_counter_style:
208 {
209   \tl_map_inline:Nn \c__enumext_counter_style_tl
210   {
211     \regex_replace_once:nnN { \c{##1}\* }
212     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
213   }
214 }
```

(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.

```
215 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
216 {
217   * ~ #2
218   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
219   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
220 }
```

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

11.5.1 Utilities for environments and levels

```

__enumext_is_not_nested:
  __enumext_is_on_first_level:

```

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

```

221 \cs_new_protected:Nn \__enumext_is_not_nested:
222 {
223   \str_case:en { \@currenvir }
224   {
225     {enumext}
226     {
227       \bool_lazy_and:nnT
228       { \bool_not_p:n { \g__enumext_standar_bool } }
229       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
230       {
231         \bool_gset_true:N \g__enumext_standar_bool
232       }
233     }
234     {enumext*}
235     {
236       \bool_lazy_and:nnT
237       { \bool_not_p:n { \g__enumext_starred_bool } }
238       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
239       {
240         \bool_gset_true:N \g__enumext_starred_bool
241       }
242     }
243   }
244 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§11.24.1), `\l__enumext_starred_first_bool` (§11.24.1) and `\l__enumext_anskey_env_bool` (§11.28) 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.

```

245 \cs_new_protected:Nn \__enumext_is_on_first_level:
246 {
247   \bool_lazy_all:nT
248   {
249     { \bool_if_p:N \g__enumext_standar_bool }
250     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
251     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
252   }
253   {
254     \bool_set_true:N \l__enumext_standar_first_bool
255     \bool_set_true:N \l__enumext_anskey_env_bool
256     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
257     \tl_gset:Nn \g__enumext_start_line_tl
258     {
259       on ~ line ~ \exp_not:V \inputlineno
260     }
261   }
262   \bool_lazy_all:nT
263   {
264     { \bool_if_p:N \g__enumext_starred_bool }
265     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
266     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
267   }
268   {
269     \bool_set_true:N \l__enumext_starred_first_bool
270     \bool_set_true:N \l__enumext_anskey_env_bool
271     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
272     \tl_gset:Nn \g__enumext_start_line_tl
273     {
274       on ~ line ~ \exp_not:V \inputlineno
275     }
276   }
277 }

```

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

__enumext_keyans_start_line:

The function __enumext_keyans_start_line: will save the start line number of the environments **keyans**, **keyans*** and **keyanspic** in the variable \l__enumext_check_start_line_env_tl to use in the __enumext_check_starred_cmd:n function.

```

278 \cs_new_protected:Nn \__enumext_keyans_start_line:
279 {
280   \str_case:en { \@currentenvir }
281   {
282     {keyans}
283     {
284       \tl_set:Nc \l__enumext_check_start_line_env_tl
285       {
286         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
287       }
288     }
289     {keyans*}
290     {
291       \tl_set:Nc \l__enumext_check_start_line_env_tl
292       {
293         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
294       }
295     }
296     {keyanspic}
297     {
298       \tl_set:Nc \l__enumext_check_start_line_env_tl
299       {
300         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
301       }
302     }
303   }
304 }

```

(End of definition for __enumext_keyans_start_line:.)

11.5.2 Utilities for log and terminal

__enumext_reset_global_vars:

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

__enumext_reset_global_int:

__enumext_reset_global_bool:

__enumext_reset_global_tl:

```

305 \cs_new_protected:Nn \__enumext_reset_global_vars:
306 {
307   \__enumext_reset_global_int:
308   \__enumext_reset_global_bool:
309   \__enumext_reset_global_tl:
310 }
311 \cs_new_protected:Nn \__enumext_reset_global_int:
312 {
313   \int_gzero:N \g__enumext_item_number_int
314   \int_gzero:N \g__enumext_item_anskey_int
315   \int_gzero:N \g__enumext_item_answer_diff_int
316 }
317 \cs_new_protected:Nn \__enumext_reset_global_bool:
318 {
319   \bool_gset_false:N \g__enumext_check_ans_key_bool
320   \bool_gset_false:N \g__enumext_standar_bool
321   \bool_gset_false:N \g__enumext_starred_bool
322 }
323 \cs_new_protected:Nn \__enumext_reset_global_tl:
324 {
325   \tl_gclear:N \g__enumext_store_name_tl
326   \tl_gclear:N \g__enumext_start_line_tl
327   \tl_gclear:N \g__enumext_envir_name_tl
328 }

```

(End of definition for __enumext_reset_global_vars: and others.)

__enumext_log_global_vars:

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.

__enumext_log_answer_vars:

```

329 \cs_new_protected:Nn \__enumext_log_global_vars:
330 {
331   \msg_log:nneeee { enumext } { prop-seq-int-hook }
332   { \g__enumext_store_name_tl }

```



```

333     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
334     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
335     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
336   }

```

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.

```

337 \cs_new_protected:Nn \__enumext_log_answer_vars:
338 {
339   \msg_log:nneee { enumext } { item-answer-hook }
340   { \int_use:N \g__enumext_item_number_int }
341   { \int_use:N \g__enumext_item_anskey_int }
342   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
343 }

```

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

11.6 Copying list and minipage environments

The `list` environment provided by L^AT_EX has the following plain form:

```

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

```

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

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w

```

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

```

344 \__enumext_at_begin_document:n
345 {
346   \cs_new_eq:NN \__enumext_start_list:nn \list
347   \cs_new_eq:NN \__enumext_stop_list: \endlist
348   \cs_new_eq:NN \__enumext_item_std:w \item
349 }

```

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

The `minipage` environment provided by L^AT_EX has the following (simplified) plain form:

```

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

```

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

```

\__enumext_minipage:w
\__enumext_endminipage:

```

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

```

350 \__enumext_at_begin_document:n
351 {
352   \cs_new_eq:NN \__enumext_minipage:w \minipage
353   \cs_new_eq:NN \__enumext_endminipage: \endminipage
354 }

```

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

11.7 The internal minipage environment

```

\__enumext_internal_mini_page:
  __enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` environment (*custom version* of `minipage`) setting the `\ifminipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\vspace{0pt}` to maintain alignment on “top”. 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 (§11.35) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.39)

```

355 \cs_new_protected:Nn \__enumext_internal_mini_page:
356 {
357   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
358   {
359     \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
360     {

```

```

361         \__enumext_minipage:w [ t ] { ##1 }
362         \legacy_if_gset_false:n { @minipage }
363         \vspace { 0pt }
364     }
365     { \__enumext_endminipage: }
366 }
367 }

```

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

11.8 Compatibility with hyperref and footnotehyper

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

```

368 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
369 \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”.

```

370 \cs_new_protected:Nn \__enumext_after_hyperref:
371 {
372     \IfPackageLoadedTF { hyperref }
373     {
374         \msg_info:nnn { enumext } { package-load } { hyperref }
375         \bool_set_true:N \l__enumext_hyperref_bool
376         \IfHyperBoolean{hyperfootnotes}
377         {
378             \typeout{hyperfootnotes=true}
379             \bool_set_true:N \l__enumext_footnotes_key_bool
380         }
381         { \typeout{hyperfootnotes=false} }
382     }
383     { }

```

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`.

```

384 \bool_if:NT \l__enumext_footnotes_key_bool
385 {
386     \IfPackageLoadedTF { footnotehyper }
387     {
388         \msg_info:nnn { enumext } { package-load } { footnotehyper }
389     }
390     {
391         \typeout{No ~ footnotehyper ~ load}
392         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
393         \bool_set_false:N \l__enumext_footnotes_key_bool
394     }
395 }

```

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.

```

396 \bool_if:NTF \l__enumext_hyperref_bool
397 {
398     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
399     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
400 }
401 {
402     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
403     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
404 }
405 }

```

(End of definition for `__enumext_after_hyperref:`, `__enumext_hypertarget:nn`, and `__enumext_phantomsection:`.)

```
\__enumext_newlabel:nn
```

The function `__enumext_newlabel:nn` write the information to the `.aux` file when using the `save-ref` key. The arguments taken by the function are:

#1: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

The trick here is to manage the number of arguments passed to `\newlabel{#1}{#2}` according to the presence of the `hyperref` package.

```

406 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
407 {
408   \protected@write \@auxout { }
409   {
410     \token_to_str:N \newlabel {#1}
411     {
412       {#2}
413       \bool_if:NT \l__enumext_hyperref_bool
414       { { \thepage } {#2} {#1} }
415       { }
416     }
417   }
418   \__enumext_hypertarget:nn {#1} { }
419   \__enumext_phantomsection:
420 }

```

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

11.9 Definition of counters

```

\__enumext_define_counters:Nn
\__enumext_define_counters:cn

```

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

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

```

421 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
422 {
423   \cs_if_exist:cTF { c@ #2 }
424   { \msg_fatal:nnn { enumext } { counters } { #2 } }
425   {
426     \tl_set:Nn #1 { #2 }
427     \newcounter { #2 }
428   }
429 }

```

(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   430 \__enumext_define_counters:Nn \l__enumext_counter_i_tl   { enumXi   }
enumXii  431 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl  { enumXii  }
enumXiii 432 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv  433 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl  { enumXiv  }
enumXvii 434 \__enumext_define_counters:Nn \l__enumext_counter_v_tl   { enumXv   }
enumXviii435 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl  { enumXvi  }
          436 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
          437 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

11.10 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*`, `\Alpha*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```
\__enumext_register_counter_style:Nn
```

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

```

438 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
439 {
440   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
441   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
442 }
443 \__enumext_register_counter_style:Nn \arabic { 0 }
444 \__enumext_register_counter_style:Nn \Alpha { M }
445 \__enumext_register_counter_style:Nn \alph { m }
446 \__enumext_register_counter_style:Nn \Roman { VIII }
447 \__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.

```
448 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
449 {
450   \hbox_set:Nn \__enumext_label_width_by_box {#2}
451   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
452 }
453 \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.

```
454 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
455 {
456   \tl_clear_new:N #1
457   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
458   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
459   \tl_map_inline:Nn \g__enumext_counter_styles_tl
460   {
461     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
462     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
463     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
464   }
465   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
466   { \tl_use:N \g__enumext_widest_label_tl }
467   \tl_set_eq:cN { the #2 } #1
468 }
469 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.11 Setting keys associated with label

`font`
`labelsep`
`labelwidth`
`wrap-label`
`wrap-label*`

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

```
470 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
471 {
472   \keys_define:nn { enumext / #1 }
473   {
474     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
475     font      .value_required:n = true,
476     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
477     labelsep  .initial:n   = {0.3333em},
478     labelsep  .value_required:n = true,
479     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
480     labelwidth .value_required:n = true,
481     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
482     wrap-label .initial:n   = {##1},
483     wrap-label .value_required:n = true,
484     wrap-label* .code:n = {
485       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
486       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
487     },
488     wrap-label* .value_required:n = true,
489   }
490 }
491 \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.

```

492 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
493 {
494   \keys_define:nn { enumext / #1 }
495   {
496     align .choice:,
497     align / left .code:n =
498       {
499         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
500         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
501       },
502     align / right .code:n =
503       {
504         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
505         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
506       },
507     align / center .code:n =
508       {
509         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
510         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
511       },
512     align / unknown .code:n =
513       \msg_error:nneee { enumext } { unknown-choice }
514       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
515     align .initial:n = left,
516     align .value_required:n = true,
517   }
518 }
519 \clist_map_inline:nn
520 {
521   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
522 }
523 { \__enumext_tmp:nn #1 }

524 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
525 {
526   \keys_define:nn { enumext / #1 }
527   {
528     align .choice:,
529     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
530     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
531     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
532     align / unknown .code:n =
533       \msg_error:nneee { enumext } { unknown-choice }
534       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
535     align .initial:n = left,
536     align .value_required:n = true,
537   }
538 }
539 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

11.12 Setting label and ref keys

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

11.12.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

540 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
541 {
542   \keys_define:nn { enumext / #1 }
543   {
544     label .code:n = {
545       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
546       { l__enumext_counter_#2_tl } {##1}
547       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
548       \l__enumext_current_widest_dim

```

```

549         },
550         label .initial:n = #3,
551         label .value_required:n = true,
552         ref .code:n = \__enumext_standar_ref:n {##1},
553         ref .value_required:n = true,
554     }
555 }
556 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
557 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
558 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
559 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

The `__enumext_standar_ref:n` first we will pass the key argument to `\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.

```

560 \cs_new_protected:Npn \__enumext_standar_ref:n #1
561 {
562     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
563     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
564     {
565         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
566     }
567     {
568         \tl_set_eq:Nc
569         \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
570         \__enumext_regex_counter_style:
571         \tl_set_eq:Nc
572         \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
573         \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
574         {
575             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
576             { \exp_not:V \l__enumext_ref_key_arg_tl }
577         }
578     }
579 }

```

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

```

580 \cs_new_protected:Npn \__enumext_standar_ref:
581 {
582     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
583     {
584         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
585     }
586 }

```

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

11.12.2 Define and set `label` and `ref` keys for `enumext*` and `keyans*` environments

`label` 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
587 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
588 {
589     \keys_define:nn { enumext / #1 }
590     {
591         label .code:n = {
592             \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
593             { \l__enumext_counter_#2_tl } {##1}
594             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
595             \l__enumext_current_widest_dim
596         },
597         label .initial:n = #3,
598         label .value_required:n = true,
599         ref .code:n = \__enumext_starred_ref:n {##1},
600         ref .value_required:n = true,
601     }

```



```

602 }
603 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
604 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*} }

```

(End of definition for label and others.)

```

\__enumext_starred_ref:n
\__enumext_starred_ref:

```

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

```

605 \cs_new_protected:Npn \__enumext_starred_ref:n #1
606 {
607   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
608   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
609   {
610     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
611     {
612       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
613     }
614     {
615       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
616       \__enumext_regex_counter_style:
617       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
618       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
619       {
620         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
621         { \exp_not:V \l__enumext_ref_key_arg_tl }
622       }
623     }
624   }
625   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
626   {
627     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
628     {
629       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
630     }
631     {
632       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
633       \__enumext_regex_counter_style:
634       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
635       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
636       {
637         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
638         { \exp_not:V \l__enumext_ref_key_arg_tl }
639       }
640     }
641   }
642 }

```

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.

```

643 \cs_new_protected:Nn \__enumext_starred_ref:
644 {
645   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
646   {
647     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
648     {
649       \tl_use:N \l__enumext_renew_the_count_vii_tl
650     }
651   }
652   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
653   {
654     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
655     {
656       \tl_use:N \l__enumext_renew_the_count_viii_tl
657     }
658   }
659 }

```

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

11.12.3 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.

```

label
ref
\__enumext_label_v_tl 660 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 661 {
662   label .code:n = {
663     \__enumext_label_style:cnv { \__enumext_label_v_tl }
664     { \__enumext_counter_v_tl } {#1}
665     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
666     \__enumext_current_widest_dim
667     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
668     { \__enumext_counter_vi_tl } {#1}
669     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
670     \__enumext_current_widest_dim
671   },
672   label .initial:n = \Alph*,
673   label .value_required:n = true,
674   ref .code:n = \__enumext_keyans_ref:n {#1},
675   ref .value_required:n = true,
676 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
677 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
678 {
679   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
680   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
681   {
682     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
683   }
684   {
685     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
686     \__enumext_regex_counter_style:
687     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
688     \tl_put_right:Ne \__enumext_renew_the_count_v_tl
689     {
690       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
691       { \exp_not:V \__enumext_ref_key_arg_tl }
692     }
693   }
694 }

```

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

```

695 \cs_new_protected:Nn \__enumext_keyans_ref:
696 {
697   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
698   {
699     \tl_use:N \__enumext_renew_the_count_v_tl
700   }
701 }

```

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

11.13 Setting start and widest keys

The function `__enumext_start_from:NNn` used by the **start** key take three arguments:

```

\__enumext_start_from:NNn
\__enumext_start_from:ccn
#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.

```

702 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
703 {
704   \__enumext_if_is_int:nTF { #3 }
705   {
706     \int_set:Nn #2 {#3}
707   }

```

```

708     {
709         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
710         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
711         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
712         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
713     }
714 }
715 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

(End of definition for __enumext_start_from:NNn.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

- #1: The counter associated with the environment level
- #2: \l__enumext_label_X_tl
- #3: \l__enumext_labelwidth_X_dim
- #4: *<integer or string>*

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

```

716 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
717 {
718     \__enumext_if_is_int:nTF {#4}
719     {
720         \setcounter{enumX#1} { #4 }
721     }
722     {
723         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
724         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
725         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
726         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
727     }
728     \__enumext_label_width_by_box:cv
729     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
730 }
731 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

(End of definition for __enumext_widest_from:nNNn.)

Now define and set `start` and `widest` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

start
widest
\l__enumext_start_X_int
732 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
733 {
734     \keys_define:nn { enumext / #1 }
735     {
736         start .code:n = {
737             \__enumext_start_from:ccn
738             { \l__enumext_label_#2_tl }
739             { \l__enumext_start_#2_int } {##1}
740         },
741         start .initial:n = 1,
742         widest .code:n = {
743             \__enumext_widest_from:nccn {#2}
744             { \l__enumext_label_#2_tl }
745             { \l__enumext_labelwidth_#2_dim } {##1}
746         },
747         widest .value_required:n = true,
748         start .value_required:n = true,
749     }
750 }
751 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.14 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
752 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
753 {
754     \keys_define:nn { enumext / #1 }

```

```

755 {
756   topsep .skip_set:c = { l__enumext_topsep_#2_skip },
757   topsep .initial:n = {#3},
758   topsep .value_required:n = true,
759   partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
760   partopsep .initial:n = {#4},
761   partopsep .value_required:n = true,
762   parsep .skip_set:c = { l__enumext_parsep_#2_skip },
763   parsep .initial:n = {#5},
764   parsep .value_required:n = true,
765   itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
766   itemsep .initial:n = {#6},
767   itemsep .value_required:n = true,
768   noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
769   noitemsep .value_forbidden:n = true,
770   nosep .meta:n = {
771     itemsep = 0pt, parsep = 0pt,
772     topsep = 0pt, partopsep = 0pt,
773   },
774   nosep .value_forbidden:n = true,
775 }
776 }

```

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

```

777 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
778 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
779 { 4.0pt plus 2.0pt minus 1.0pt }
780 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
781 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
782 { 2.0pt plus 1.0pt minus 1.0pt }
783 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
784 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
785 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
786 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
787 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
788 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
789 { 2.0pt plus 1.0pt minus 1.0pt }
790 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
791 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
792 { 4.0pt plus 2.0pt minus 1.0pt }
793 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
794 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
795 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

11.15 Setting base-fix key

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

```

base-fix
\__enumext_nested_base_line_fix:

```

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

```

796 \cs_set_protected:Npn \__enumext_tmp:n #1
797 {
798   \keys_define:nn { enumext / #1 }
799   {
800     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
801     base-fix .initial:n = false,
802     base-fix .value_forbidden:n = true,
803   }
804 }
805 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

The function `__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the `<keys>`. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§11.35) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39)

```

806 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
807 {
808   \bool_lazy_and:nnT

```

```

809 { \bool_if_p:N \__enumext_standar_first_bool }
810 { \bool_if_p:N \__enumext_base_line_fix_bool }
811 {
812   \mode_leave_vertical:
813   \vspace { -\baselineskip }
814   \keys_set:nn { enumext / level-1 }
815   {
816     topsep = 0pt, above = 0pt, above* = 0pt,
817   }
818 }
819 \bool_lazy_and:nnT
820 { \bool_if_p:N \__enumext_starred_first_bool }
821 { \bool_if_p:N \__enumext_base_line_fix_bool }
822 {
823   \mode_leave_vertical:
824   \vspace { -\baselineskip }
825   \keys_set:nn { enumext / enumext* }
826   {
827     topsep = 0pt, above = 0pt, above* = 0pt,
828   }
829 }
830 \bool_set_false:N \__enumext_base_line_fix_bool
831 }

```

🔗 This key is enabled by default in the command `\printkeyans` (§11.42).

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

11.16 Setting keys for horizontal spaces

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

```

832 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
833 {
834   \keys_define:nn { enumext / #1 }
835   {
836     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
837     itemindent .value_required:n = true,
838     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
839     rightmargin .value_required:n = true,
840     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
841     listparindent .value_required:n = true,
842     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
843     list-offset .value_required:n = true,
844     list-indent .code:n =
845       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
846       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
847     list-indent .value_required:n = true,
848   }
849 }
850 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__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.

```

851 \cs_set_protected:Npn \__enumext_tmp:n #1
852 {
853   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
854 }
855 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

11.16.1 Functions for setting the fake `itemindent`

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `0pt`. Here I will need to place `\mode_leave_vertical:` and the plain \TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

856 \cs_set_protected:Nn \__enumext_fake_item:
857 {
858   \dim_compare:nNnT
859     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
860     >
861     { \c_zero_dim }

```

```

862     {
863         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
864         {
865             \exp_not:N \mode_leave_vertical:
866             \exp_not:n { \skip_horizontal:n }
867             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
868             \ignorespaces
869         }
870     }
871 }
872 \cs_set_protected:Nn \__enumext_keyans_fake_item:
873 {
874     \dim_compare:nNnT
875     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
876     {
877         \tl_set:Nc \l__enumext_fake_item_indent_v_tl
878         {
879             \exp_not:N \mode_leave_vertical:
880             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
881         }
882     }
883 }
884 \cs_set_protected:Nn \__enumext_fake_item_vii:
885 {
886     \dim_compare:nNnT
887     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
888     {
889         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
890         {
891             \exp_not:N \mode_leave_vertical:
892             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
893         }
894     }
895 }
896 \cs_set_protected:Nn \__enumext_fake_item_viii:
897 {
898     \dim_compare:nNnT
899     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
900     {
901         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
902         {
903             \exp_not:N \mode_leave_vertical:
904             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
905         }
906     }
907 }

```

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

11.17 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.

```

908 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
909 {
910     \keys_define:nn { enumext / #1 }
911     {
912         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
913         show-length .initial:n = false,
914     }
915 }
916 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.18 Setting before, after and first keys

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

```

917 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2

```

`after`

`first`


```

918 {
919   \keys_define:nn { enumext / #1 }
920   {
921     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
922     before .value_required:n = true,
923     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
924     before* .value_required:n = true,
925     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
926     after .value_required:n = true,
927     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
928     first .value_required:n = true,
929   }
930 }
931 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

11.18.1 Functions for before, after and first keys in enumext

The function `__enumext_before_args_exec:` executes the $\{\langle code \rangle\}$ set by the *before** key “before” the *enumext* environment is started. The $\{\langle code \rangle\}$ is executed “without” knowing any definition of the *second argument* of the list.

```

932 \cs_new_protected:Nn \__enumext_before_args_exec:
933 {
934   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
935 }

```

The function `__enumext_before_keys_exec:` executes the $\{\langle code \rangle\}$ set by the *before* key “before” the *enumext* environment is started in *second argument* of the list. The $\{\langle code \rangle\}$ is executed “knowing” all definition and values provides by $\langle keys \rangle$.

```

936 \cs_new_protected:Nn \__enumext_before_keys_exec:
937 {
938   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
939 }

```

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

```

940 \cs_new_protected:Nn \__enumext_after_stop_list:
941 {
942   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
943 }

```

The function `__enumext_after_args_exec:` executes the $\{\langle code \rangle\}$ 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*.

```

944 \cs_new_protected:Nn \__enumext_after_args_exec:
945 {
946   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
947 }

```

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

11.18.2 Functions for before, after and first keys in keyans

The function `__enumext_before_args_exec_v:` executes the $\{\langle code \rangle\}$ set by the *before** key “before” the *keyans* environment is started. The $\{\langle code \rangle\}$ is executed “without” knowing any definition of the $\{\langle arg two \rangle\}$ of the list.

```

948 \cs_new_protected:Nn \__enumext_before_args_exec_v:
949 {
950   \tl_use:N \l__enumext_before_starred_key_v_tl
951 }

```

The function `__enumext_before_keys_exec_v:` executes the $\{\langle code \rangle\}$ set by the *before* key “before” the *keyans* environment is started in $\{\langle arg two \rangle\}$ of the list. The $\{\langle code \rangle\}$ is executed “knowing” all definition and values provides by $\langle keys \rangle$.

```

952 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
953 {
954   \tl_use:N \l__enumext_before_no_starred_key_v_tl
955 }

```

The function `__enumext_after_stop_list_v:` executes the `{⟨code⟩}` set by the `after` key “after” the `keyans` environment has finished.

```

956 \cs_new_protected:Nn \__enumext_after_stop_list_v:
957 {
958   \tl_use:N \l__enumext_after_stop_list_v_tl
959 }

```

The function `__enumext_after_args_exec_v:` executes the `{⟨code⟩}` set by the `first` key after the end of `{⟨arg two⟩}` of the list defining the `keyans` environment, just before the first occurrence of `\item`.

```

960 \cs_new_protected:Nn \__enumext_after_args_exec_v:
961 {
962   \tl_use:N \l__enumext_after_list_args_v_tl
963 }

```

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

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:

```

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

```

964 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
965 {
966   \tl_use:N \l__enumext_before_starred_key_vii_tl
967 }
968 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
969 {
970   \tl_use:N \l__enumext_before_starred_key_viii_tl
971 }

```

The functions `__enumext_before_keys_exec_vii:` and `__enumext_before_keys_exec_viii:` executes the `{⟨code⟩}` set by the `before` key “before” in `enumext*` and `keyans*` environments is started in `{⟨arg two⟩}` of the list. The `{⟨code⟩}` is executed “knowing” all definition and values provides by `⟨keys⟩`.

```

972 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
973 {
974   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
975 }
976 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
977 {
978   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
979 }

```

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

```

980 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
981 {
982   \tl_use:N \l__enumext_after_stop_list_vii_tl
983 }
984 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
985 {
986   \tl_use:N \l__enumext_after_stop_list_viii_tl
987 }

```

The function `__enumext_after_args_exec_v:` executes the `{⟨code⟩}` set by the `first` key after the end of `{⟨arg two⟩}` of the list defining the `keyans` environment, just before the first occurrence of `\item`.

```

988 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
989 {
990   \tl_use:N \l__enumext_after_list_args_vii_tl
991 }
992 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
993 {
994   \tl_use:N \l__enumext_after_list_args_viii_tl
995 }

```

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

11.19 Setting keys for multicols and minipage

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

```

996 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
997 {
998   \keys_define:nn { enumext / #1 }
999   {
1000     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
1001     mini-env .value_required:n = true,
1002     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
1003     mini-sep .initial:n = 0.3333em,
1004     mini-sep .value_required:n = true,
1005     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
1006     columns-sep .value_required:n = true,
1007     columns .int_set:c = { \__enumext_columns_#2_int },
1008     columns .initial:n = 1,
1009     columns .value_required:n = true,
1010   }
1011 }
1012 \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.

```

1013 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1014 {
1015   \keys_define:nn { enumext / #1 }
1016   {
1017     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1018     mini-right .value_required:n = true,
1019     mini-right* .code:n = {
1020       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1021       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1022     },
1023     mini-right* .value_required:n = true,
1024   }
1025 }
1026 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

11.20 Adjustment of vertical spaces for multicols

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

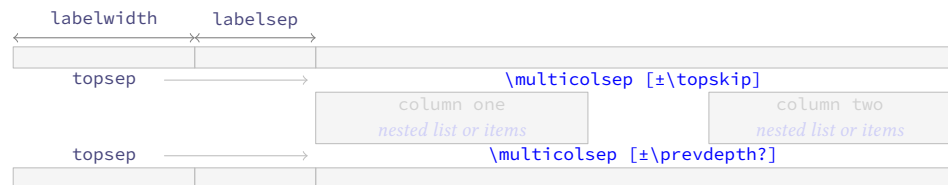


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

11.20.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

1027 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1028 {
1029   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1030   {
1031     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1032   }
1033   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1034   {
1035     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1036   }
1037   \__enumext_add_pre_parsep:
1038 }

```

(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*.

```

1039 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1040 {
1041   \int_case:nn { \l__enumext_level_int }
1042   {
1043     { 2 }{
1044       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1045       {
1046         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1047       }
1048     }
1049     { 3 }{
1050       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1051       {
1052         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1053       }
1054     }
1055     { 4 }{
1056       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1057       {
1058         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1059       }
1060     }
1061   }
1062 }

```

(End of definition for `__enumext_add_pre_parsep:.`)

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

```

1063 \cs_new_protected:Nn \__enumext_multi_addvspace:
1064 {
1065   \__enumext_multi_set_vskip:
1066   \mode_if_vertical:T
1067   {
1068     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1069     {
1070       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1071     }
1072     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1073     {
1074       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1075     }
1076   }
1077   \par\nopagebreak
1078   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1079 }

```

(End of definition for `__enumext_multi_addvspace:.`)

11.20.2 Adjustment of vertical spaces for multicol in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

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

```

1080 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1081 {
1082   \skip_set:Nn \l__enumext_multicol_above_v_skip
1083   {
1084     \l__enumext_topsep_v_skip
1085   }
1086   \skip_set:Nn \l__enumext_multicol_below_v_skip
1087   {
1088     \l__enumext_topsep_v_skip
1089   }
1090 }
1091 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1092 {
1093   \__enumext_keyans_multi_set_vskip:
1094   \mode_if_vertical:T
1095   {
1096     \skip_add:Nn \l__enumext_multicol_above_v_skip
1097     {
1098       \skip_use:N \l__enumext_partopsep_v_skip
1099     }
1100     \skip_add:Nn \l__enumext_multicol_below_v_skip
1101     {
1102       \skip_use:N \l__enumext_partopsep_v_skip
1103     }
1104   }
1105   \par\nopagebreak
1106   \addvspace{ \l__enumext_multicol_above_v_skip }
1107 }

```

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

11.21 Adjustment of vertical spaces for minipage

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

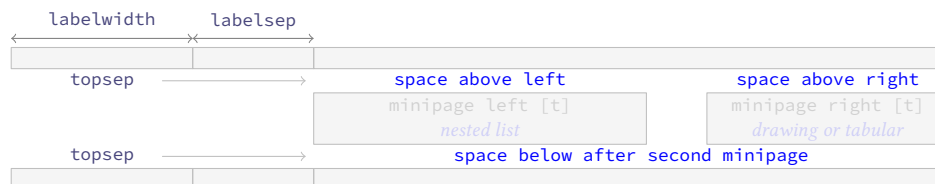


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

11.21.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_mini_set_vskip:`

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

First determine if the `multicol` environment is active by comparing the value of the `\l__enumext_columns_X_int` variable handled by the `columns` key, according to this comparison we set the adjusted

values for `\l__enumext_minipage_left_skip`, `\l__enumext_minipage_right_skip` and `\l__enumext_minipage_after_skip`.

```

1108 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1109 {
1110   \int_compare:nNnTF
1111     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1112     {

```

If `\multicols` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep` of the current level and the value of `\parsep` of the previous level, if these are zero we will use `\strutbox` as the basis for the calculations.

```

1113   \skip_if_eq:nnTF
1114     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1115   {
1116     \skip_set:Nn \l__enumext_minipage_left_skip
1117       {
1118         -0.150\box_dp:N \strutbox
1119       }
1120     \skip_set:Nn \l__enumext_minipage_right_skip
1121       {
1122         0.695\box_dp:N \strutbox
1123       }
1124     \skip_set:Nn \l__enumext_minipage_after_skip
1125       {
1126         \box_dp:N \strutbox
1127       }
1128     \__enumext_zero_parsep:
1129   }
1130   {
1131     \skip_set:Nn \l__enumext_minipage_left_skip
1132       {
1133         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1134       }
1135     \skip_set:Nn \l__enumext_minipage_right_skip
1136       {
1137         0.695\box_dp:N \strutbox
1138       }
1139     \skip_set:Nn \l__enumext_minipage_after_skip
1140       {
1141         1.85\box_dp:N \strutbox
1142         + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1143       }
1144   }
1145 }
1146 {

```

If only `\enumext` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep`, if this is zero we will use `\strutbox` as the basis for the calculations.

```

1147   \skip_if_eq:nnTF
1148     { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1149   {
1150     \skip_set:Nn \l__enumext_minipage_left_skip
1151       {
1152         0.5\box_dp:N \strutbox
1153         - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1154       }
1155     \skip_set:Nn \l__enumext_minipage_right_skip
1156       {
1157         \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1158       }
1159     \skip_set:Nn \l__enumext_minipage_after_skip
1160       {
1161         1.6\box_dp:N \strutbox
1162       }
1163   }
1164   {
1165     \skip_set:Nn \l__enumext_minipage_left_skip
1166       {
1167         0.5875\box_dp:N \strutbox
1168         - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }

```



```

1169         }
1170         \skip_set:Nn \l__enumext_minipage_right_skip
1171         {
1172             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1173             + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1174         }
1175         \skip_set:Nn \l__enumext_minipage_after_skip
1176         {
1177             0.325\box_dp:N \strutbox
1178             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1179         }
1180     }
1181 }
1182 }

```

(End of definition for `__enumext_mini_set_vskip:`)

`__enumext_zero_parsep:` The function `__enumext_zero_parsep:` “adjusted” the value of `\l__enumext_minipage_after_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```

1183 \cs_new_protected:Nn \__enumext_zero_parsep:
1184 {
1185     \int_case:nn { \l__enumext_level_int }
1186     {
1187         { 2 }{
1188             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1189             {
1190                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1191             }
1192         }
1193         { 3 }{
1194             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1195             {
1196                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1197             }
1198         }
1199         { 4 }{
1200             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1201             {
1202                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1203             }
1204         }
1205     }
1206 }

```

(End of definition for `__enumext_zero_parsep:`)

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

```

1207 \cs_new_protected:Nn \__enumext_mini_addvspace:
1208 {
1209     \__enumext_mini_set_vskip:
1210     \mode_if_vertical:T
1211     {
1212         \skip_add:Nn \l__enumext_minipage_left_skip
1213         {
1214             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1215         }
1216         \skip_add:Nn \l__enumext_minipage_after_skip
1217         {
1218             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1219         }
1220     }
1221     \par\nopagebreak
1222     \addvspace { \l__enumext_minipage_left_skip }
1223 }

```

(End of definition for `__enumext_mini_addvspace:`)

11.21.2 Adjustment of vertical spaces for minipage in keyans

`__enumext_keyans_mini_set_vskip:`

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

```

1224 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1225 {
1226   \skip_zero_new:N \l__enumext_minipage_after_skip
1227   \skip_zero_new:N \l__enumext_minipage_left_skip
1228   \skip_zero_new:N \l__enumext_minipage_right_skip
1229   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1230   {
1231     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1232     {
1233       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1234       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1235       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1236       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1237       {
1238         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1239       }
1240     }
1241     {
1242       \skip_set:Nn \l__enumext_minipage_left_skip
1243       {
1244         \skip_use:N \l__enumext_topsep_v_skip
1245       }
1246       \skip_set:Nn \l__enumext_minipage_right_skip
1247       {
1248         0.705\box_dp:N \strutbox
1249       }
1250       \skip_set:Nn \l__enumext_minipage_after_skip
1251       {
1252         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1253       }
1254     }
1255   }
1256   {
1257     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1258     {
1259       \skip_set:Nn \l__enumext_minipage_left_skip
1260       {
1261         0.5\box_dp:N \strutbox
1262         + \l__enumext_partopsep_v_skip
1263       }
1264       \skip_set:Nn \l__enumext_minipage_right_skip
1265       {
1266         \l__enumext_partopsep_v_skip
1267       }
1268       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1269     }
1270     {
1271       \skip_set:Nn \l__enumext_minipage_left_skip
1272       {
1273         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1274       }
1275       \skip_set:Nn \l__enumext_minipage_right_skip
1276       {
1277         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1278       }
1279       \skip_set:Nn \l__enumext_minipage_after_skip
1280       {
1281         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1282       }
1283     }
1284   }
1285 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

The function `__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `__enumext_mini_env*` environment in `keyans`, taking into account whether $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ is in

(*horizontal mode*) or (*vertical mode*). For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```

1286 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1287 {
1288   \__enumext_keyans_mini_set_vskip:
1289   \mode_if_vertical:T
1290   {
1291     \skip_add:Nn \l__enumext_minipage_left_skip
1292     {
1293       \l__enumext_partopsep_v_skip
1294     }
1295     \skip_add:Nn \l__enumext_minipage_after_skip
1296     {
1297       \l__enumext_partopsep_v_skip
1298     }
1299   }
1300   \par\nopagebreak
1301   \addvspace { \l__enumext_minipage_left_skip }
1302 }

```

(End of definition for `__enumext_keyans_mini_addvspace:.`)

11.21.3 Adjustment of vertical spaces for minipage in `enumext*` and `keyans*`

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1303 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1304 {
1305   \skip_zero_new:N \l__enumext_minipage_left_skip
1306   \skip_gzero_new:N \g__enumext_minipage_right_skip
1307   \skip_gzero_new:N \g__enumext_minipage_after_skip
1308   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1309   {
1310     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1311     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1312   }
1313   {
1314     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1315     \skip_gset:Nn \g__enumext_minipage_right_skip
1316     {
1317       \l__enumext_topsep_vii_skip
1318     }
1319     \skip_gset:Nn \g__enumext_minipage_after_skip
1320     {
1321       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1322     }
1323   }
1324 }
1325 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1326 {
1327   \skip_zero_new:N \l__enumext_minipage_after_skip
1328   \skip_zero_new:N \l__enumext_minipage_left_skip
1329   \skip_zero_new:N \l__enumext_minipage_right_skip
1330   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1331   {
1332     \skip_set:Nn \l__enumext_minipage_left_skip
1333     {
1334       0.5\box_dp:N \strutbox
1335     }
1336     \skip_set:Nn \l__enumext_minipage_right_skip
1337     {
1338       \l__enumext_partopsep_viii_skip
1339     }
1340     \skip_set:Nn \l__enumext_minipage_after_skip
1341     {
1342       1.6\box_dp:N \strutbox
1343     }
1344   }
1345 }

```

```

1346     \skip_set:Nn \l__enumext_minipage_left_skip
1347     {
1348         0.5875\box_dp:N \strutbox
1349     }
1350     \skip_set:Nn \l__enumext_minipage_right_skip
1351     {
1352         \l__enumext_topsep_viii_skip
1353     }
1354     \skip_set:Nn \l__enumext_minipage_after_skip
1355     {
1356         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1357     }
1358 }
1359 }

```

(End of definition for `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:`)

`\l__enumext_mini_addvspace_vii:`
`\l__enumext_mini_addvspace_viii:`

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

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

```

1360 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1361 {
1362     \l__enumext_mini_set_vskip_vii:
1363     \par\nopagebreak
1364     \addvspace { \l__enumext_minipage_left_skip }
1365 }
1366 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1367 {
1368     \l__enumext_mini_set_vskip_viii:
1369     \par\nopagebreak
1370     \addvspace { \l__enumext_minipage_left_skip }
1371 }

```

(End of definition for `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:`)

11.21.4 The command `\miniright`

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

`\miniright`

First we will perform some checks to prevent the command from being executed outside the `enumext` environment or from being executed inside the `keyanspic` environment, then we call the internal functions for the `enumext` and `keyans` environments.

```

1372 \NewDocumentCommand \miniright { s }
1373 {
1374     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1375     {
1376         \msg_error:nnn { enumext } { wrong-miniright-place }
1377     }
1378     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1379     {
1380         \msg_error:nnn { enumext } { wrong-miniright-place }
1381     }
1382     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1383     {
1384         \l__enumext_keyans_mini_right_cmd:n {#1}
1385     }
1386     { \l__enumext_mini_right_cmd:n {#1} }
1387 }

```

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

`\l__enumext_mini_right_cmd:n`

The function `\l__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 `\l__enumext_mini_env*` environment on the “left side”, then we open the `\l__enumext_mini_env*` environment on

the “*right side*”, apply our adjusted “*vertical spaces*”, followed by adding the `\centering` command when the starred argument ‘***’ is not present and set zero `\g__enumext_minipage_stat_int`, otherwise we return an error.

```

1388 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1389 {
1390     \dim_compare:nNnTF
1391     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1392     {
1393         \__enumext_multicols_stop:
1394         \end{__enumext_mini_env*}
1395         \hfill
1396         \begin{__enumext_mini_env*}
1397         { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1398         \par\addvspace { \l__enumext_minipage_right_skip }
1399         \bool_if:nF {#1}
1400         {
1401             \centering
1402         }
1403         \int_gzero:N \g__enumext_minipage_stat_int
1404     }
1405     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1406 }

```

(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.

```

1407 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1408 {
1409     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1410     {
1411         \__enumext_keyans_multicols_stop:
1412         \end{__enumext_mini_env*}
1413         \hfill
1414         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1415         \par\addvspace { \l__enumext_minipage_right_skip }
1416         \bool_if:nF {#1}
1417         {
1418             \centering
1419         }
1420         \int_gzero:N \g__enumext_minipage_stat_int
1421     }
1422     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1423 }

```

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

11.22 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.

```

above* 1424 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1425 {
below* 1426 \keys_define:nn { enumext / #1 }
1427 {
1428     above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1429     above .value_required:n = true,
1430     above* .code:n = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1431                 \keys_set:nn { enumext / #1 } { above = {##1} },
1432     above* .value_required:n = true,
1433     below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1434     below .value_required:n = true,
1435     below* .code:n = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1436                 \keys_set:nn { enumext / #1 } { below = {##1} },
1437     below* .value_required:n = true,

```

```

1438     }
1439 }
1440 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

11.22.1 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.

```

1441 \cs_new_protected:Nn \__enumext_vspace_above:
1442 {
1443     \skip_if_eq:nnF
1444     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1445     {
1446         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1447         {
1448             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1449         }
1450         {
1451             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1452         }
1453     }
1454 }

```

(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.

```

1455 \cs_new_protected:Nn \__enumext_vspace_below:
1456 {
1457     \skip_if_eq:nnF
1458     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1459     {
1460         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1461         {
1462             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1463         }
1464         {
1465             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1466         }
1467     }
1468 }

```

(End of definition for `__enumext_vspace_below:`.)

11.22.2 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.

```

1469 \cs_new_protected:Nn \__enumext_vspace_above_v:
1470 {
1471     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1472     {
1473         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1474         {
1475             \vspace*{ \l__enumext_vspace_above_v_skip }
1476         }
1477         { \vspace { \l__enumext_vspace_above_v_skip } }
1478     }
1479 }

```

(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.

```

1480 \cs_new_protected:Nn \__enumext_vspace_below_v:
1481 {
1482     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1483     {
1484         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1485         {
1486             \vspace*{ \l__enumext_vspace_below_v_skip }

```



```

1487     }
1488     { \vspace { \l__enumext_vspace_below_v_skip } }
1489   }
1490 }

```

(End of definition for `__enumext_vspace_below_v:`)

11.22.3 Functions for above and below keys in `enumext*` `keyans*`

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

```

\__enumext_vspace_above_vii:
\__enumext_vspace_above_viii:
1491 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1492 {
1493   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1494   {
1495     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1496     {
1497       \vspace*{ \l__enumext_vspace_above_vii_skip }
1498     }
1499     { \vspace { \l__enumext_vspace_above_vii_skip } }
1500   }
1501 }
1502 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1503 {
1504   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1505   {
1506     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1507     {
1508       \vspace*{ \l__enumext_vspace_above_viii_skip }
1509     }
1510     { \vspace { \l__enumext_vspace_above_viii_skip } }
1511   }
1512 }

```

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

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

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_viii:
1513 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1514 {
1515   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1516   {
1517     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1518     {
1519       \vspace*{ \l__enumext_vspace_below_vii_skip }
1520     }
1521     { \vspace { \l__enumext_vspace_below_vii_skip } }
1522   }
1523 }
1524 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1525 {
1526   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1527   {
1528     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1529     {
1530       \vspace*{ \l__enumext_vspace_below_viii_skip }
1531     }
1532     { \vspace { \l__enumext_vspace_below_viii_skip } }
1533   }
1534 }

```

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

11.23 Setting series, resume and resume* keys

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1535 \cs_set_protected:Npn \__enumext_tmp:n #1
1536 {
1537   \keys_define:nn { enumext / #1 }
1538   {
1539     series .str_set:N = \__enumext_series_str,
1540     series .value_required:n = true,
1541     resume .code:n = \__enumext_resume_series:n {##1},
1542     resume* .code:n = \__enumext_resume_starred:,
1543     resume* .value_forbidden:n = true,
1544   }
1545 }
1546 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

11.23.1 Internal functions for series key

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

```

1547 \cs_new:Npn \__enumext_filter_series:n #1
1548 {
1549   \use:e
1550   {
1551     \keyval_parse:NNn
1552     \__enumext_filter_series_key:n
1553     \__enumext_filter_series_pair:nn {#1}
1554   }
1555 }

```

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.

```

1556 \cs_new:Npn \__enumext_filter_series_key:n #1
1557 {
1558   \str_case:nnF {#1}
1559   {
1560     { resume } {}
1561     { resume* } {}
1562     { base-fix } {}
1563   }
1564   { , { \exp_not:n {#1} } }
1565 }

```

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`, `save-ans` and `save-key` keys.

```

1566 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1567 {
1568   \str_case:nnF {#1}
1569   {
1570     { series } {}
1571     { resume } {}
1572     { start } {}
1573     { save-ans } {}
1574     { save-key } {}
1575   }
1576   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
1577 }

```

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

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§11.35) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39).

```

1578 \cs_new_protected:Npn \__enumext_parse_series:n #1

```

```

1579 {
1580     \str_if_empty:NTF \l__enumext_series_str
1581     {
1582         \bool_if:NF \l__enumext_resume_active_bool
1583         {
1584             \__enumext_resume_last:n {#1}
1585         }
1586     }
1587     {
1588         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1589         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1590         { \__enumext_filter_series:n {#1} }
1591         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1592         {
1593             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1594         }
1595     }
1596 }

```

The function `__enumext_resume_last:n` will be in charge of saving the filtering *⟨keys⟩* when the *series* key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the `enumext` environment and in the variable `\g__enumext_starred_series_tl` for the `enumext*` environment. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the *series* key is not being used.

```

1597 \cs_new_protected:Npn \__enumext_resume_last:n #1
1598 {
1599     \bool_if:NT \l__enumext_standar_first_bool
1600     {
1601         \tl_gclear:N \g__enumext_standar_series_tl
1602         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1603     }
1604     \bool_if:NT \l__enumext_starred_first_bool
1605     {
1606         \tl_gclear:N \g__enumext_starred_series_tl
1607         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1608     }
1609 }

```

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

11.23.2 Internal function to save counter value

`__enumext_resume_save_counter:`

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same *⟨series name⟩* but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§11.35) and the `enumext*` environment definition (§11.39).

```

1610 \cs_new_protected:Nn \__enumext_resume_save_counter:
1611 {
1612     \bool_if:NT \g__enumext_standar_bool
1613     {
1614         \tl_if_empty:NF \l__enumext_series_str
1615         {
1616             \int_gset_eq:cN
1617             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1618         }
1619         \tl_if_empty:NTF \l__enumext_resume_name_tl
1620         {
1621             \str_if_empty:NF \l__enumext_series_str
1622             {
1623                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1624             }
1625         }
1626         {
1627             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1628             {

```

```

1629         \int_gset_eq:cN
1630         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1631     }
1632 }
1633 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1634 {
1635     \int_gset_eq:cN
1636     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1637 }
1638 }
1639 \bool_if:NT \g__enumext_starred_bool
1640 {
1641     \tl_if_empty:NF \l__enumext_series_str
1642     {
1643         \int_gset_eq:cN
1644         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1645     }
1646     \tl_if_empty:NTF \l__enumext_resume_name_tl
1647     {
1648         \str_if_empty:NT \l__enumext_series_str
1649         {
1650             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1651         }
1652     }
1653     {
1654         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1655         {
1656             \int_gset_eq:cN
1657             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1658         }
1659     }
1660     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1661     {
1662         \int_gset_eq:cN
1663         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1664     }
1665 }
1666 }

```

(End of definition for `__enumext_resume_save_counter:`.)

11.23.3 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.

```

1667 \cs_new_protected:Npn \__enumext_resume_series:n #1
1668 {
1669     \tl_if_empty:NTF {#1}
1670     {
1671         \__enumext_resume_counter:n { }
1672     }
1673     {
1674         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1675         {
1676             \__enumext_resume_counter:n {#1}
1677             \bool_if:NT \g__enumext_standar_bool
1678             {
1679                 \keys_set:nv { enumext / level-1 }
1680                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1681             }
1682             \bool_if:NT \g__enumext_starred_bool
1683             {
1684                 \keys_set:nv { enumext / enumext* }
1685                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1686             }
1687         }
1688     }

```

```

1689         \bool_if:NT \g__enumext_standar_bool
1690         {
1691             \msg_error:nnn { enumext } { unknown-series } {#1}
1692         }
1693         \bool_if:NT \g__enumext_starred_bool
1694         {
1695             \msg_error:nnn { enumext } { unknown-series } {#1}
1696         }
1697     }
1698 }
1699 }

```

(End of definition for __enumext_resume_series:n.)

```

\__enumext_resume_counter:n
\__enumext_resume_counter:
  \__enumext_resume_counter_series:
  \__enumext_resume_counter_save_ans:

```

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

```

1700 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1701 {
1702     \bool_set_true:N \l__enumext_resume_active_bool
1703     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1704     \tl_if_empty:NTF \l__enumext_resume_name_tl
1705     {
1706         \__enumext_resume_counter:
1707     }
1708     {
1709         \__enumext_resume_counter_series:
1710     }
1711     \__enumext_resume_counter_save_ans:
1712 }

```

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.

```

1713 \cs_new_protected:Nn \__enumext_resume_counter:
1714 {
1715     \bool_if:NT \g__enumext_standar_bool
1716     {
1717         \int_gincr:N \g__enumext_resume_int
1718         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1719     }
1720     \bool_if:NT \g__enumext_starred_bool
1721     {
1722         \int_gincr:N \g__enumext_resume_vii_int
1723         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1724     }
1725 }

```

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

```

1726 \cs_new_protected:Nn \__enumext_resume_counter_series:
1727 {
1728     \bool_if:NT \g__enumext_standar_bool
1729     {
1730         \int_set:Nn \l__enumext_start_i_int
1731         {
1732             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1733         }
1734     }
1735     \bool_if:NT \g__enumext_starred_bool
1736     {
1737         \int_set:Nn \l__enumext_start_vii_int
1738         {
1739             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1740         }
1741     }
1742 }

```

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.

```

1743 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1744 {
1745   \bool_lazy_and:nnT
1746     { \bool_if_p:N \__enumext_standar_first_bool }
1747     { \bool_if_p:N \__enumext_store_active_bool }
1748   {
1749     \int_set:Nn \__enumext_start_i_int
1750     {
1751       \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1752     }
1753   }
1754   \bool_lazy_and:nnT
1755     { \bool_if_p:N \__enumext_starred_first_bool }
1756     { \bool_if_p:N \__enumext_store_active_bool }
1757   {
1758     \int_set:Nn \__enumext_start_vii_int
1759     {
1760       \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1761     }
1762   }
1763 }

```

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

11.23.4 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.

```

1764 \cs_new_protected:Nn \__enumext_resume_starred:
1765 {
1766   \bool_if:NT \g__enumext_standar_bool
1767   {
1768     \tl_if_empty:NF \g__enumext_standar_series_tl
1769     {
1770       \__enumext_resume_counter:n { }
1771       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1772     }
1773   }
1774   \bool_if:NT \g__enumext_starred_bool
1775   {
1776     \tl_if_empty:NF \g__enumext_starred_series_tl
1777     {
1778       \__enumext_resume_counter:n { }
1779       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1780     }
1781   }
1782 }

```

(End of definition for `__enumext_resume_starred:`.)

11.24 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.

11.24.1 Setting `save-ans` key

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

```

1783 \cs_set_protected:Npn \__enumext_tmp:n #1
1784 {
1785   \keys_define:nn { enumext / #1 }
1786   {
1787     save-ans .code:n = \__enumext_storing_set:n {##1},
1788     save-ans .value_required:n = true,
1789   }
1790 }
1791 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

11.24.2 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:`.

```
1792 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1793 {
1794   \msg_term:nnVV { enumext } { save-ans-log }
1795   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1796 }
1797 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1798 {
1799   \msg_term:nnVV { enumext } { save-ans-log-hook }
1800   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1801 }
```

(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.

```
1802 \cs_new_protected:Npn \__enumext_storing_set:n #1
1803 {
1804   \tl_set:Nx \l__enumext_store_name_tl {#1}
1805   \tl_if_empty:NTF \l__enumext_store_name_tl
1806   {
1807     \bool_lazy_or:nnT
1808     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1809     {
1810       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1811     }
1812   }
1813   {
1814     \bool_lazy_or:nnT
1815     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1816     {
1817       \__enumext_start_save_ans_msg:
1818       \__enumext_storing_exec:
1819     }
1820   }
1821 }
```

The function `__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy `{⟨store name⟩}` into the global variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§11.28). 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*`.

```
1822 \cs_new_protected:Nn \__enumext_storing_exec:
1823 {
1824   \bool_set_true:N \l__enumext_store_active_bool
1825   \bool_set_true:N \l__enumext_check_answers_bool
1826   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1827   \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1828   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1829   {
1830     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1831     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1832   }
1833   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1834   {
1835     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1836     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
```



```

1837     }
1838     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1839     {
1840         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1841         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1842     }
1843 }

```

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

11.24.3 The check answer mechanism

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

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

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

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

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

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

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

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

11.24.4 Setting check-ans and no-store keys

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

```

check-ans 1844 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1845 {
1846     \keys_define:nn { enumext / #1 }
1847     {
1848         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1849         check-ans .initial:n = false,
1850         check-ans .value_required:n = true,
1851         no-store .code:n = {
1852             \bool_set_false:N \l__enumext_check_answers_bool
1853             \bool_set_false:N \l__enumext_check_ans_key_bool
1854         },
1855         no-store .value_forbidden:n = true,
1856     }
1857 }
1858 \clist_map_inline:nn
1859 {
1860     level-1, level-2, level-3, level-4, enumext*
1861 }
1862 { \__enumext_tmp:n {#1} }

```

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

11.24.5 Set-up check answer mechanism

`__enumext_check_ans_active:` 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.

```

1863 \cs_new_protected:Nn \__enumext_check_ans_active:
1864 {
1865   \tl_if_empty:NF \l__enumext_store_name_tl
1866   {
1867     \bool_if:NT \l__enumext_check_answers_bool
1868     {
1869       \__enumext_check_ans_level:
1870     }
1871   }
1872 }

```

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*”.

```

1873 \cs_new_protected:Nn \__enumext_check_ans_level:
1874 {
1875   \int_case:nn { \l__enumext_level_int }
1876   {
1877     { 1 }{
1878       \bool_lazy_all:nT
1879       {
1880         { \bool_if_p:N \g__enumext_starred_bool }
1881         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1882       }
1883       {
1884         \int_gdecr:N \g__enumext_item_number_int
1885         \bool_set_false:N \l__enumext_item_number_bool
1886       }
1887     }
1888     { 2 }{
1889       \int_gdecr:N \g__enumext_item_number_int
1890       \bool_set_false:N \l__enumext_item_number_bool
1891     }
1892     { 3 }{
1893       \int_gdecr:N \g__enumext_item_number_int
1894       \bool_set_false:N \l__enumext_item_number_bool
1895     }
1896     { 4 }{
1897       \int_gdecr:N \g__enumext_item_number_int
1898       \bool_set_false:N \l__enumext_item_number_bool
1899     }
1900   }

```

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.

```

1901   \int_case:nn { \l__enumext_level_h_int }
1902   {
1903     { 1 }{
1904       \bool_lazy_all:nT
1905       {
1906         { \bool_if_p:N \g__enumext_standar_bool }
1907         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1908       }
1909       {
1910         \int_gdecr:N \g__enumext_item_number_int
1911         \bool_set_false:N \l__enumext_item_number_bool
1912       }
1913     }
1914   }
1915 }

```

(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.

```

1916 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1917 {
1918   \bool_lazy_and:nnT
1919     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1920     { \bool_if_p:N \g__enumext_standar_bool }
1921   {
1922     \bool_gset_true:N \g__enumext_check_ans_key_bool
1923   }
1924   \bool_lazy_and:nnT
1925     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1926     { \bool_if_p:N \g__enumext_starred_bool }
1927   {
1928     \bool_gset_true:N \g__enumext_check_ans_key_bool
1929   }
1930 }

```

(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:.

```

1931 \cs_new_protected:Nn \__enumext_item_answer_diff:
1932 {
1933   \int_gset:Nn \g__enumext_item_answer_diff_int
1934   {
1935     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1936   }
1937 }

```

(End of definition for __enumext_item_answer_diff:.)

__enumext_check_ans_show:

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

```

1938 \cs_new_protected:Nn \__enumext_check_ans_show:
1939 {
1940   \int_case:nn { \g__enumext_item_answer_diff_int }
1941   {
1942     { -1 } { \__enumext_check_ans_msg_less: }
1943     { 0 } { \__enumext_check_ans_msg_same_ok: }
1944     { 1 } { \__enumext_check_ans_msg_greater: }
1945   }
1946 }
1947 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1948 {
1949   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1950   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1951 }
1952 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1953 {
1954   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1955   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1956 }
1957 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1958 {
1959   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1960   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1961 }

```

(End of definition for __enumext_check_ans_show: and others.)

__enumext_check_ans_log:

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

__enumext_check_ans_log_msg_less:

__enumext_check_ans_log_msg_same_ok:

__enumext_check_ans_log_msg_greater:

```

1962 \cs_new_protected:Nn \__enumext_check_ans_log:
1963 {
1964   \int_case:nn { \g__enumext_item_answer_diff_int }
1965   {
1966     { -1 } { \__enumext_check_ans_log_msg_less: }
1967     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1968     { 1 } { \__enumext_check_ans_log_msg_greater: }
1969   }
1970 }
1971 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1972 {
1973   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1974   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1975 }
1976 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1977 {
1978   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1979   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1980 }
1981 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1982 {
1983   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1984   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1985 }

```

(End of definition for __enumext_check_ans_log: and others.)

11.24.6 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.

```

1986 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1987 {
1988   \int_compare:nNnT
1989   { \g__enumext_check_starred_cmd_int } = { 0 }
1990   {
1991     \msg_warning:nnnV
1992     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1993   }
1994   \int_compare:nNnT
1995   { \g__enumext_check_starred_cmd_int } > { 1 }
1996   {
1997     \msg_warning:nnnV
1998     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1999   }
2000   \int_gzero:N \g__enumext_check_starred_cmd_int
2001   \tl_clear:N \l__enumext_check_start_line_env_tl
2002 }

```

(End of definition for __enumext_check_starred_cmd:n.)

11.25 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*` (§11.28) 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.

```

2003 \cs_new_protected:Nn \__enumext_execute_after_env:
2004 {
2005   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2006   {
2007     \tl_if_empty:NF \g__enumext_store_name_tl
2008     {
2009       \__enumext_stop_save_ans_msg:
2010       \__enumext_item_answer_diff:
2011       \__enumext_log_global_vars:
2012       \__enumext_log_answer_vars:

```

```

2013         \bool_if:NTF \g__enumext_check_ans_key_bool
2014         {
2015             \__enumext_check_ans_show:
2016         }
2017         { \__enumext_check_ans_log: }
2018         \__enumext_undefine_anskey_env:
2019     }
2020     \__enumext_reset_global_vars:
2021 }
2022 }

```

• This function is passed to the function `__enumext_after_env:nn` for the environments `enumext` (§11.35) and `enumext*` (§11.39) and it is executed only when the environments are not nested or at some level of these..

(End of definition for `__enumext_execute_after_env:.`)

11.26 Keys and functions associated with storage

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

```

2023 \cs_set_protected:Npn \__enumext_tmp:n #1
2024 {
2025     \keys_define:nn { enumext / #1 }
2026     {
2027         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2028         wrap-ans .initial:n = \fbox{##1},
2029         wrap-ans .value_required:n = true,
2030         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2031         wrap-opt .initial:n = [{##1}],
2032         wrap-opt .value_required:n = true,
2033         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2034         save-sep .initial:n = {, ~},
2035         save-sep .value_required:n = true,
2036         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2037         mark-ans .initial:n = \textasteriskcentered,
2038         mark-ans .value_required:n = true,
2039         mark-pos .choice:,
2040         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2041         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2042         mark-pos / unknown .code:n =
2043             \msg_error:nnee { enumext } { unknown-choice }
2044             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2045         mark-pos .initial:n = right,
2046         mark-pos .value_required:n = true,
2047         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2048         show-ans .initial:n = false,
2049         show-ans .value_required:n = true,
2050         show-pos .bool_set:N = \l__enumext_show_position_bool,
2051         show-pos .initial:n = false,
2052         show-pos .value_required:n = true,
2053         mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2054         mark-ref .initial:n = \textasteriskcentered,
2055         mark-ref .value_required:n = true,
2056         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2057         save-ref .initial:n = false,
2058         save-ref .value_required:n = true,
2059     }
2060 }
2061 \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`.

```

2062 \cs_set_protected:Npn \__enumext_tmp:n #1
2063 {
2064     \keys_define:nn { enumext / #1 }
2065     {
2066         mark-pos .choice:,
2067         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2068         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },

```

```

2069     mark-pos .initial:n = right,
2070     mark-pos .value_required:n = true,
2071     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2072     show-ans .initial:n = false,
2073     show-ans .value_required:n = true,
2074     show-pos .bool_set:N = \l__enumext_show_position_bool,
2075     show-pos .initial:n = false,
2076     show-pos .value_required:n = true,
2077   }
2078 }
2079 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

11.26.1 Store optional arguments of the environments

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

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

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

```

2080 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2081 {
2082   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2083   {
2084     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2085     \tl_set:ce
2086       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2087       { \__enumext_filter_save_key:n {#1} }
2088   }
2089 }
2090 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2091 {
2092   \bool_if:NF \l__enumext_store_save_key_vii_bool
2093   {
2094     \tl_clear:N \l__enumext_store_save_key_vii_tl
2095     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2096   }
2097 }

```

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

11.26.2 Setting save-key key

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

save-key

The values set by this key passed in the optional arguments of the `enumext` and `enumext*` environments will override the values of the `\l__enumext_store_save_key_X_tl` variable set by the functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n`.

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

```

2098 \cs_set_protected:Npn \__enumext_tmp:n #1
2099 {
2100   \keys_define:nn { enumext / enumext* }
2101   {
2102     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2103     save-key .value_required:n = true,
2104   }
2105   \keys_define:nn { enumext / #1 }
2106   {
2107     save-key .code:n = \__enumext_parse_save_key:n {##1},
2108     save-key .value_required:n = true,
2109   }
2110 }
2111 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \l__enumext_tmp:n {#1} }

```

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

```

\__enumext_parse_save_key:n
  \__enumext_parse_save_key_vii:n

```

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for storing the filtered $\langle keys \rangle$ in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2112 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2113 {
2114   \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2115   \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2116   \tl_set:ce
2117     { \l__enumext_store_save_key_ \__enumext_level: _tl }
2118     { \__enumext_filter_save_key:n {#1} }
2119 }
2120 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2121 {
2122   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2123   \tl_clear:N \l__enumext_store_save_key_vii_tl
2124   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2125 }

```

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

11.26.3 Internal functions to store optional arguments

```

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

```

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

```

2126 \cs_new:Npn \__enumext_filter_save_key:n #1
2127 {
2128   \use:e
2129   {
2130     \keyval_parse:NNn
2131       \__enumext_filter_save_key_key:n
2132       \__enumext_filter_save_key_pair:nn {#1}
2133   }
2134 }

```

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

```

2135 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2136 {
2137   \str_case:nnF {#1}
2138   {
2139     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2140   }
2141   { , { \exp_not:n {#1} } }
2142 }

```

The function `__enumext_filter_save_key_pair:nn` will be responsible for filtering the $\langle keys \rangle$ 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.

```

2143 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2144 {
2145   \str_case:nnF {#1}
2146   {
2147     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2148     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2149     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2150     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2151     { mini-right* } {}
2152   }
2153   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2154 }

```

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

11.26.4 Function for storing content in prop list

__enumext_store_addto_prop:n
 __enumext_store_addto_prop:V

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

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

```
2155 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2156 {
2157   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2158   {
2159     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2160   }
2161   { #1 }
2162 }
2163 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

(End of definition for __enumext_store_addto_prop:n.)

11.26.5 Function for storing content in sequence

__enumext_store_addto_seq:n
 __enumext_store_addto_seq:v
 __enumext_store_addto_seq:V

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

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

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

```
2164 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2165 {
2166   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2167 }
2168 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

(End of definition for __enumext_store_addto_seq:n.)

11.26.6 Functions for storing the list structure in the sequence

__enumext_store_level_open:
 __enumext_store_level_close:

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

```
2169 \cs_new_protected:Nn \__enumext_store_level_open:
2170 {
2171   \bool_if:NT \l__enumext_check_answers_bool
2172   {
2173     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2174     {
2175       \__enumext_store_addto_seq:n
2176       {
2177         \item \begin{enumext}
2178       }
2179     }
2180     {
2181       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2182       {
2183         \item \begin{enumext} [
2184         ]
2185       }
2186       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2187       {
2188         ]
2189       }
2190       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2191     }
2192   }
2193 \cs_new_protected:Nn \__enumext_store_level_close:
2194 {
2195   \bool_if:NT \l__enumext_check_answers_bool
2196   {
2197     \__enumext_store_addto_seq:n { \end{enumext} }
2198   }
2199 }
```

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

__enumext_store_level_open_vii:
 __enumext_store_level_close_vii:

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

```

2200 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2201 {
2202   \bool_if:NT \l__enumext_check_answers_bool
2203   {
2204     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2205     {
2206       \__enumext_store_addto_seq:n
2207       {
2208         \item \begin{enumext*}
2209       }
2210     }
2211     {
2212       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2213       {
2214         \item \begin{enumext*}[
2215       }
2216       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2217       {
2218         ]
2219       }
2220       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2221     }
2222   }
2223 }
2224 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2225 {
2226   \bool_if:NT \l__enumext_check_answers_bool
2227   {
2228     \__enumext_store_addto_seq:n { \end{enumext*} }
2229   }
2230 }

```

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

11.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
 __enumext_print_keyans_box:cc

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

#1: \l__enumext_labelwidth_X_dim
 #2: \l__enumext_labelsep_X_dim

```

2231 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2232 {
2233   \mode_leave_vertical:
2234   \skip_horizontal:n { -\dim_use:N #2 }
2235   \makebox[0pt][ r ]
2236   {
2237     \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2238     {
2239       \tl_use:N \l__enumext_mark_answer_sym_tl
2240     }
2241   }
2242   \skip_horizontal:n { \dim_use:N #2 }
2243 }
2244 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

11.27 The command \anskey and internal label and ref

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

The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows: `\anskey[⟨key = val⟩]{⟨content⟩}` so first we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```

2245 \keys_define:nn { enumext / anskey }
2246 {
2247   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2248   break-col .default:n = true,

```

```

2249     break-col .value_forbidden:n = true,
2250     item-join .int_set:N = \l__enumext_store_item_join_int,
2251     item-join .value_required:n = true,
2252     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2253     item-star .default:n = true,
2254     item-star .value_forbidden:n = true,
2255     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2256     item-sym* .value_required:n = true,
2257     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2258     item-pos* .value_required:n = true,
2259 }

```

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

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

```

2260 \NewDocumentCommand \anskey { o +m }
2261 {
2262     \__enumext_anskey_safe_outer:
2263     \group_begin:
2264         \bool_if:NT \l__enumext_check_answers_bool
2265         {
2266             \__enumext_anskey_safe_inner:n {#2}
2267             \tl_if_novalue:nF {#1}
2268             {
2269                 \keys_set:nn { enumext / anskey } {#1}
2270             }
2271             \int_gincr:N \g__enumext_item_anskey_int
2272             \__enumext_store_anskey_code:n {#2}
2273         }
2274     \group_end:
2275 }

```

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

11.27.1 Internal functions for the command

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

`__enumext_anskey_safe_inner:n`

```

2276 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2277 {
2278     \bool_if:NF \l__enumext_store_active_bool
2279     {
2280         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2281     }
2282     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2283     {
2284         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2285     }
2286     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2287     {
2288         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2289     }
2290     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2291     {
2292         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2293     }
2294 }

```

The `__enumext_anskey_safe_inner:n` function will first check to see if the passed argument is *empty* and then check to see if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2295 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2296 {
2297     \tl_if_empty:nT {#1}
2298     {
2299         \msg_error:nn { enumext } { anskey-empty-arg }

```

```

2300     }
2301     \int_incr:N \l__enumext_anskey_level_int
2302     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2303     {
2304         \msg_error:nn { enumext } { anskey-nested }
2305     }
2306     \bool_if:NF \l__enumext_item_number_bool
2307     {
2308         \msg_error:nn { enumext } { anskey-unnumber-item }
2309     }
2310     \mode_if_math:T
2311     {
2312         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2313     }
2314 }

```

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

`__enumext_store_anskey_code:n`

The internal function `__enumext_store_anskey_code:n` first we pass the *⟨argument⟩* 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” *⟨argument⟩*.

```

2315 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2316 {
2317     \__enumext_store_addto_prop:n {#1}
2318     \bool_if:NT \l__enumext_store_ref_key_bool
2319     {
2320         \__enumext_store_internal_ref:
2321     }
2322     \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2323     \tl_clear:N \l__enumext_store_anskey_arg_tl
2324     \bool_lazy_and:nnT
2325     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2326     { \bool_not_p:n { \l__enumext_starred_bool } }
2327     {
2328         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2329     }
2330     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

If the `item-join` key is present and the command is running under `enumext*` we will add (*⟨number⟩*) to `\l__enumext_store_anskey_arg_tl`.

```

2331     \bool_lazy_and:nnT
2332     { \bool_not_p:n { \l__enumext_starred_bool } }
2333     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2334     {
2335         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2336         {
2337             ( \exp_not:V \l__enumext_store_item_join_int )
2338         }
2339     }

```

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

```

2340     \bool_if:NTF \l__enumext_store_item_star_bool
2341     {
2342         \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2343         \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2344         {
2345             \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2346             {
2347                 [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2348             }
2349         }
2350         \dim_compare:nT
2351         {
2352             \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2353         }

```

```

2354     {
2355         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2356         {
2357             [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2358         }
2359     }
2360     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2361 }
2362 {
2363     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2364 }

```

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

```

2365     \bool_lazy_and:nnT
2366     { \bool_if_p:N \l__enumext_store_ref_key_bool }
2367     { \bool_if_p:N \l__enumext_hyperref_bool }
2368     {
2369         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2370         {
2371             \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2372             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2373         }
2374     }
2375     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2376 }

```

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

`__enumext_store_internal_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. 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`.

```

2377 \cs_new_protected:Nn \__enumext_store_internal_ref:
2378 {
2379     \cs_set_protected:Npn \__enumext_tmp:n ##1
2380     {
2381         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2382         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2383         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2384         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2385     }
2386     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2387     \cs_set:Npn \__enumext_tmp:n ##1
2388     { . \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.

```

2389     \bool_lazy_all:nT
2390     {
2391         { \bool_if_p:N \g__enumext_starred_bool }
2392         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2393     }
2394     {
2395         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2396         { \tl_use:N \l__enumext_label_copy_vii_tl }
2397     }
2398     \bool_lazy_all:nT
2399     {
2400         { \bool_if_p:N \l__enumext_standar_bool }
2401         { \bool_if_p:N \g__enumext_starred_bool }
2402         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2403     }
2404     {
2405         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2406         {
2407             \tl_use:N \l__enumext_label_copy_vii_tl
2408             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2409         }
2410     }

```

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.

```

2411 \bool_lazy_all:nT
2412 {
2413   { \bool_if_p:N \l__enumext_standar_bool }
2414   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2415   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2416   { \bool_not_p:n { \l__enumext_starred_bool } }
2417 }
2418 {
2419   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2420   {
2421     \tl_use:N \l__enumext_label_copy_i_tl
2422     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2423   }
2424 }
2425 \cs_set:Npn \l__enumext_tmp:n ##1
2426 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2427 \bool_lazy_all:nT
2428 {
2429   { \bool_if_p:N \l__enumext_standar_bool }
2430   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2431   { \bool_not_p:n { \g__enumext_starred_bool } }
2432   { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2433 }
2434 {
2435   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2436   {
2437     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2438     . \tl_use:N \l__enumext_label_copy_vii_tl
2439   }
2440 }

```

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

```

2441 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2442 {
2443   \l__enumext_store_name_tl \c_colon_str
2444   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2445 }

```

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

```

2446 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2447 {
2448   \l__enumext_newlabel:nn
2449   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2450   { \l__enumext_newlabel_arg_two_tl }
2451 }
2452 \l__enumext_write_aux_file_tl
2453 }

```

(End of definition for `\l__enumext_store_internal_ref:`)

`\l__enumext_anskey_show_wrap_arg:n`

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

```

2454 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_arg:n #1
2455 {
2456   \par
2457   \bool_if:NT \l__enumext_starred_bool
2458   {
2459     \cs_set:Nn \l__enumext_level: { vii }
2460   }
2461   \l__enumext_print_keyans_box:cc
2462   { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2463   { \l__enumext_labelsep_ \l__enumext_level: _dim }
2464   \l__enumext_anskey_wrapper:n { #1 }
2465 }

```

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

`__enumext_anskey_show_wrap_left:n`

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

```

2466 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2467 {
2468   \bool_if:NT \__enumext_show_answer_bool
2469   {
2470     \__enumext_anskey_show_wrap_arg:n { #1 }
2471   }
2472   \bool_if:NT \__enumext_show_position_bool
2473   {
2474     \tl_set:Nx \__enumext_mark_answer_sym_tl
2475     {
2476       \group_begin:
2477       \exp_not:N \normalfont
2478       \exp_not:N \footnotesize [ \int_eval:n
2479       {
2480         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
2481       }
2482       ]
2483       \group_end:
2484     }
2485     \__enumext_anskey_show_wrap_arg:n { #1 }
2486   }
2487 }

```

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

11.28 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:` (§11.25) which is executed after the environment in which the key `save-ans` is active.

```

2488 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2489 {
2490   \cs_undefine:c { anskey* }
2491   \cs_undefine:c { endanskey* }
2492   \cs_undefine:c { __scontents_anskey*_env_begin: }
2493   \cs_undefine:c { __scontents_anskey*_env_end: }
2494 }

```

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

```

2495 \__enumext_before_env:nn { enumext }
2496 {
2497   \bool_lazy_and:nnT
2498   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2499   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2500   {
2501     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2502     {
2503       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2504     }
2505   }
2506 }
2507 \__enumext_before_env:nn { enumext* }
2508 {

```



```

2509 \bool_lazy_and:nnT
2510 { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2511 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2512 {
2513   \cs_if_free:cF { __scontents_anskey*_env_begin: }
2514   {
2515     \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2516   }
2517 }
2518 }

```

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.

```

2519 \__enumext_before_env:nn { anskey* }
2520 {
2521   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2522   {
2523     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2524   }
2525   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2526   {
2527     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2528   }
2529   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2530   {
2531     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2532   }
2533   \bool_if:NF \l__enumext_item_number_bool
2534   {
2535     \msg_error:nn { enumext } { anskey-unnumber-item }
2536   }
2537   \mode_if_math:T
2538   {
2539     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2540   }
2541 }

```

(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:` (§11.24.1) and we will execute it only if the variable `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2542 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2543 {
2544   \bool_if:NT \l__enumext_anskey_env_bool
2545   {
2546     \newenvsc{anskey*}[store-env=#1,print-env=false]
2547     \__enumext_anskey_env_exec:
2548   }
2549 }
2550 \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`.

```

2551 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2552 {
2553   \keys_define:nn { scontents / scontents }
2554   {
2555     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2556     break-col .default:n = true,
2557     break-col .value_forbidden:n = true,
2558     item-join .int_gset:N = \g__enumext_store_item_join_int,
2559     item-join .value_required:n = true,
2560     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2561     item-star .default:n = true,

```

```

2562     item-star .value_forbidden:n = true,
2563     item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2564     item-sym* .value_required:n = true,
2565     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2566     item-pos* .value_required:n = true,
2567     print-env .undefine:,
2568     store-env .undefine:,
2569     write-out .undefine:,
2570   }
2571 }

```

The function `__enumext_anskey_env_undefine_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`.

```

2572 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2573 {
2574   \keys_define:nn { scontents / scontents }
2575   {
2576     break-col .undefine:,
2577     item-join .undefine:,
2578     item-star .undefine:,
2579     item-sym* .undefine:,
2580     item-pos* .undefine:,
2581     write-out .code:n = {
2582       \bool_set_false:N \l__scontents_storing_bool
2583       \bool_set_true:N \l__scontents_writing_bool
2584       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2585     },
2586     write-out .value_required:n = true,
2587     print-env .meta:nn = { scontents } { print-env = ##1 },
2588     print-env .default:n = true,
2589     store-env .meta:nn = { scontents } { store-env = ##1 },
2590     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} }
2591   }
2592 }

```

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*.

```

2593 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2594 {
2595   \group_begin:
2596     \int_set:Nn \tex_newlinechar:D { `^^^J }
2597     \__scontents_rescan_tokens:x
2598     {
2599       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2600       #1
2601     }
2602 }

```

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

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

```

2603 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2604 {
2605   \__enumext_before_env:nn { anskey* }
2606   {
2607     \__enumext_anskey_env_define_keys:
2608   }

```

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.

```

2609   \hook_if_empty:nF {env/anskey*/after}
2610   {
2611     \hook_gremove_code:nn {env/anskey*/after} { * }
2612   }
2613   \__enumext_after_env:nn { anskey* }
2614   {
2615     \tl_clear:N \l__enumext_store_anskey_env_tl
2616     \tl_clear:N \l__enumext_store_anskey_opt_tl

```

```

2617         \tl_gset:Ne \l__enumext_store_anskey_env_tl
2618         {
2619             \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2620         }
2621         \__enumext_anskey_env_keys:
2622         \__enumext_anskey_env_store:
2623         \__enumext_anskey_env_clean:
2624         \__enumext_anskey_env_undefine_keys:
2625     }
2626 }

```

- The use of `\hook_gremove_code:n` is necessary here, otherwise the `{\code}` passed to `__enumext_after_env:n` `{anskey*}` will be accumulated for each execution. The last function `__enumext_anskey_env_undefine_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_keys:` The function `__enumext_anskey__env_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`.

```

2627 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2628 {
2629     \bool_lazy_and:nnT
2630     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2631     { \bool_not_p:n { \l__enumext_starred_bool } }
2632     {
2633         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2634     }

```

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

```

2635     \bool_lazy_and:nnT
2636     { \bool_not_p:n { \l__enumext_starred_bool } }
2637     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2638     {
2639         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2640         {
2641             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2642         }
2643     }

```

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

```

2644     \bool_if:NT \g__enumext_store_item_star_bool
2645     {
2646         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2647         {
2648             ,item-star,
2649         }
2650         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2651         {
2652             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2653             {
2654                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2655             }
2656         }
2657         \dim_compare:nT
2658         {
2659             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2660         }
2661         {
2662             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2663             {
2664                 ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2665             }
2666         }
2667     }
2668 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment, we will execute and only if the variable `\l__enumext_store_anskey_env_tl` is not empty using the functions `__enumext_store_anskey_code:n` and `__enumext_rescan_anskey_env:n`.

```

2669 \cs_new_protected:Nn \__enumext_anskey_env_store:

```

```

2670 {
2671   \group_begin:
2672   \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2673   {
2674     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2675     {
2676       \exp_args:Ne
2677       \__enumext_store_anskey_code:n
2678       {
2679         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2680       }
2681     }
2682     {
2683       \keys_set:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2684       \exp_args:Ne
2685       \__enumext_store_anskey_code:n
2686       {
2687         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2688       }
2689     }
2690   }
2691   \group_end:
2692 }

```

The function `__enumext_anskey_env_clean:` will return the global variables used by the `⟨keys⟩` to their initial state.

```

2693 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2694 {
2695   \bool_gset_false:N \g__enumext_store_columns_break_bool
2696   \int_gzero:N \g__enumext_store_item_join_int
2697   \bool_gset_false:N \g__enumext_store_item_star_bool
2698   \tl_gclear:N \g__enumext_store_item_symbol_tl
2699   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2700 }

```

(End of definition for `__enumext_anskey_env_keys:`, `__enumext_anskey_env_store:`, and `__enumext_anskey_env_clean:`.)

11.29 Common functions for keyans, keyans* and keyanspic

11.29.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

2701 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2702 {
2703   \tl_clear:N \l__enumext_store_current_label_tl
2704   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2705   {
2706     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2707   }
2708   {
2709     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2710   }
2711   \tl_if_novalue:nF { #1 }
2712   {
2713     % Set save-sep
2714     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2715     {
2716       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2717     }
2718     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2719   }
2720   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2721 }

```

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

11.29.2 The save-ref key for keyans, keyans* and keyanspic

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

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

```

2722 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2723 {
2724   \bool_if:NT \l__enumext_store_ref_key_bool
2725   {
2726     \cs_set_protected:Npn \__enumext_tmp:n ##1
2727     {
2728       \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2729       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2730       \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2731       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2732     }
2733     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2734     \__enumext_keyans_store_ref_aux_i:
2735   }
2736 }

```

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`.

```

2737 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2738 {
2739   \bool_if:NT \g__enumext_starred_bool
2740   {
2741     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2742   }
2743   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2744   {
2745     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2746     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2747   }
2748   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2749   {
2750     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2751     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2752   }
2753   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2754   {
2755     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2756     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2757   }
2758   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2759   {
2760     \l__enumext_store_name_tl \c_colon_str
2761     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2762   }
2763   \__enumext_keyans_store_ref_aux_ii:
2764 }

```

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.

```

2765 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2766 {
2767   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2768   {
2769     \__enumext_newlabel:nn
2770     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2771     { \l__enumext_newlabel_arg_two_tl }
2772   }
2773   \l__enumext_write_aux_file_tl
2774 }

```

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

11.29.3 Storing content in sequence

```
\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:
```

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

```
2775 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2776 {
2777   \tl_clear:N \l__enumext_store_current_label_tl
2778   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2779   {
2780     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2781   }
2782   {
2783     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2784   }
2785   \tl_if_novalue:nF { #1 }
2786   {
2787     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2788     {
2789       \tl_put_right:Ne \l__enumext_store_current_label_tl
2790       {
2791         \l__enumext_store_keyans_item_opt_sep_tl
2792       }
2793     }
2794     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2795   }
2796   \__enumext_keyans_addto_seq_link:
2797 }
```

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.

```
2798 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2799 {
2800   \bool_lazy_and:nnT
2801   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2802   { \bool_if_p:N \l__enumext_hyperref_bool }
2803   {
2804     \tl_put_right:Ne \l__enumext_store_current_label_tl
2805     {
2806       \hfill \exp_not:N \hyperlink
2807       {
2808         \exp_not:V \l__enumext_newlabel_arg_one_tl
2809       }
2810       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2811     }
2812   }
2813   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2814   \bool_if:NT \l__enumext_check_answers_bool
2815   {
2816     \int_gincr:N \g__enumext_item_anskey_int
2817   }
2818 }
```

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

11.29.4 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.

```
2819 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2820 {
2821   \tl_if_novalue:nF { #1 }
2822   {
```

```

2823         \tl_set:Nc \l__enumext_store_current_opt_arg_tl { #1 }
2824     }
2825     \bool_if:NT \l__enumext_show_answer_bool
2826     {
2827         \__enumext_keyans_show_ans:
2828     }
2829     \bool_if:NT \l__enumext_show_position_bool
2830     {
2831         \__enumext_keyans_show_pos:
2832     }
2833 }
2834 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2835 {
2836     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2837     {
2838         \bool_lazy_or:nnT
2839         { \bool_if_p:N \l__enumext_show_answer_bool }
2840         { \bool_if_p:N \l__enumext_show_position_bool }
2841         {
2842             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2843         }
2844     }
2845 }
2846 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2847 {
2848     \tl_put_left:Nn \l__enumext_label_v_tl
2849     {
2850         \__enumext_print_keyans_box:NN
2851         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2852     }
2853 }
2854 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2855 {
2856     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2857     {
2858         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2859         {
2860             \group_begin:
2861             \exp_not:N \normalfont
2862             \exp_not:N \footnotesize [ \int_eval:n
2863             {
2864                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2865             }
2866             ]
2867             \group_end:
2868         }
2869     }
2870     {
2871         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2872         {
2873             \group_begin:
2874             \exp_not:N \normalfont
2875             \exp_not:N \footnotesize [ \int_eval:n
2876             {
2877                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2878             }
2879             ]
2880             \group_end:
2881         }
2882     }
2883     \tl_put_left:Nn \l__enumext_label_v_tl
2884     {
2885         \__enumext_print_keyans_box:NN
2886         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2887     }
2888 }

```

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

11.30 Setting item-sym* and item-pos* keys

In order to have a cleaner implementation of `\item*` it is best to define a couple of keys that allow us to control and set by default the `\symbol` and its `\offset`.

```

item-sym* Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
2889 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2890 {
2891   \keys_define:nn { enumext / #1 }
2892   {
2893     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
2894     item-sym* .value_required:n = true,
2895     item-sym* .initial:n = { $\star$ },
2896     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
2897     item-pos* .value_required:n = true,
2898   }
2899 }
2900 \clist_map_inline:nn
2901 {
2902   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2903 }
2904 { \__enumext_tmp:nn #1 }
```

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

11.31 Redefining \footnote command

`__enumext_footnotetext:nn` To keep the correct numbering of `\footnote` and to make it work correctly with the `mini-env` key and 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](#).

```

2905 \cs_new_protected:Nn \__enumext_footnotetext:nn
2906 {
2907   \footnotetext[#1]{#2}
2908 }
2909 \cs_new_protected:Nn \__enumext_renew_footnote:
2910 {
2911   \seq_gclear:N \g__enumext_footnote_arg_seq
2912   \seq_gclear:N \g__enumext_footnote_int_seq
2913   \RenewDocumentCommand \footnote { o +m }
2914   {
2915     \tl_if_novalue:nTF {##1}
2916     {
2917       \stepcounter{footnote}
2918       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2919     }
2920     {
2921       \int_gset:Nn \g__enumext_footnote_int { ##1 }
2922     }
2923     \footnotemark [ \g__enumext_footnote_int ]
2924     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2925     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2926   }
2927 }
2928 \cs_new_protected:Nn \__enumext_print_footnote:
2929 {
2930   \seq_if_empty:NF \g__enumext_footnote_int_seq
2931   {
2932     \seq_map_pairwise_function:NNN
2933     \g__enumext_footnote_int_seq
2934     \g__enumext_footnote_arg_seq
2935     \__enumext_footnotetext:nn
2936   }
2937 }
```

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

11.32 Redefining \item command

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.

11.32.1 The `\item` command in enumext

`__enumext_default_item:n`

The `\item` and `\item[custom]` commands work in the usual way on `enumext`.

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_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `__enumext_item_std:w`.

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.

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `__enumext_make_label: (§11.33)`.

```

2938 \cs_new_protected:Npn \__enumext_default_item:n #1
2939 {
2940   \tl_if_novalue:nTF {#1}
2941   {
2942     \bool_if:NT \l__enumext_check_answers_bool
2943     {
2944       \int_gincr:N \g__enumext_item_number_int
2945       \bool_set_true:N \l__enumext_item_number_bool
2946     }
2947     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2948     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2949   }
2950   {
2951     \bool_set_eq:cc
2952     { \l__enumext_wrap_label_ \__enumext_level: _bool }
2953     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2954     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2955   }
2956 }

```

(End of definition for `__enumext_default_item:n`)

`__enumext_starred_item:nn`

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 set by the `labelsep` key and can be *offset* using 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 optional argument in the global variable `\g__enumext_item_symbol_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_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `__enumext_item_std:w`.

In this function the optional argument of `__enumext_item_std:w` is omitted, we only want it to be numbered.

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `__enumext_make_label: (§11.33)`.

```

2957 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2958 {
2959   \tl_if_novalue:nF {#1}
2960   {
2961     \tl_set:cn { \l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2962   }
2963   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2964   \tl_if_novalue:nTF {#2}
2965   {
2966     \dim_set_eq:cc
2967     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2968     { \l__enumext_labelsep_ \__enumext_level: _dim }
2969   }
2970   {
2971     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2972   }
2973   \bool_if:NT \l__enumext_check_answers_bool
2974   {

```

```

2975         \int_gincr:N \g__enumext_item_number_int
2976         \bool_set_true:N \l__enumext_item_number_bool
2977     }
2978     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2979     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2980 }

```

(End of definition for __enumext_starred_item:nn.)

`__enumext_redefine_item:` The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment for the internal mechanism of check-answers for `check-ans` key and adding the starred `\item*` version.

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.34.2).

```

2981 \cs_new_protected:Nn \__enumext_redefine_item:
2982 {
2983     \RenewDocumentCommand \item { s o o }
2984     {
2985         \bool_if:nTF {##1}
2986         {
2987             \__enumext_starred_item:nn {##2} {##3}
2988         }
2989         { \__enumext_default_item:n {##2} }
2990     }
2991 }

```

(End of definition for __enumext_redefine_item:.)

11.32.2 The `\item` command in keyans

The `\item*` and `\item*[\langle content \rangle]` commands *store* the current `\label` next to the `[\langle content \rangle]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key.

`__enumext_keyans_default_item:n` The function `__enumext_keyans_default_item:n` executes the original behavior of the `\item`.

```

2992 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
2993 {
2994     \tl_if_novalue:nTF { #1 }
2995     {
2996         \bool_set_true:N \l__enumext_wrap_label_v_bool
2997         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
2998     }
2999     {
3000         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3001         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3002     }
3003 }

```

(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`.

```

3004 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3005 {
3006     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3007     \__enumext_keyans_show_left:n { #1 }
3008     \bool_set_true:N \l__enumext_wrap_label_v_bool
3009     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item:

```

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

```

3010     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3011     \__enumext_keyans_addto_prop:n { #1 }
3012     \__enumext_keyans_store_ref:
3013     \__enumext_keyans_addto_seq:n { #1 }
3014     \int_gincr:N \g__enumext_check_starred_cmd_int
3015 }

```

(End of definition for __enumext_keyans_starred_item:n.)

`\item*` The function `__enumext_keyans_redefine_item:` is responsible for adding the *starred* and *optional* argument by the `__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we need to use `\peek_remove_spaces:n` to prevent an unwanted space when using `\item*` in conjunction with the `itemindent` key.

This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.34.2).

```

3016 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3017 {
3018   \RenewDocumentCommand \item { s o }
3019   {
3020     \bool_if:nTF {##1}
3021     {
3022       \peek_remove_spaces:n
3023       {
3024         \__enumext_keyans_starred_item:n {##2}
3025       }
3026     }
3027     {
3028       \__enumext_keyans_default_item:n {##2}
3029     }
3030   }
3031 }

```

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

11.33 Redefining `\makeLabel` command

Redefine `\makeLabel` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` and `keyans` environments.

11.33.1 Redefining `\makeLabel` for `enumext`

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

```

3032 \cs_new_protected:Nn \__enumext_item_starred:
3033 {
3034   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3035   {
3036     \mode_leave_vertical:
3037     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3038     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
3039     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3040   }
3041 }

```

(End of definition for `__enumext_item_starred:`.)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment.

This function is passed to `__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.34.2).

```

3042 \cs_new_protected:Nn \__enumext_make_label:
3043 {
3044   \RenewDocumentCommand \makeLabel { m }
3045   {
3046     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
3047     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3048     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3049     {
3050       \__enumext_item_starred:
3051       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3052     }
3053     { ##1 }
3054     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
3055     \tl_gclear:N \g__enumext_item_symbol_tl
3056   }
3057 }

```

(End of definition for `__enumext_make_label:`.)

11.33.2 Redefining `\makeLabel` for `keyans`

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.34.2).

```

3058 \cs_new_protected:Nn \__enumext_keyans_make_label:
3059 {
3060   \RenewDocumentCommand \makeLabel { m }
3061   {
3062     \tl_use:N \l__enumext_label_fill_left_v_tl
3063     \tl_use:N \l__enumext_label_font_style_v_tl
3064     \bool_if:NTF \l__enumext_wrap_label_v_bool
3065     {
3066       \__enumext_wrapper_label_v:n { ##1 }
3067     }
3068     { ##1 }
3069     \tl_use:N \l__enumext_label_fill_right_v_tl
3070   }
3071 }

```

(End of definition for `__enumext_keyans_make_label:`.)

11.34 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

11.34.1 Calculation of `\leftmargin` and `\itemindent`

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

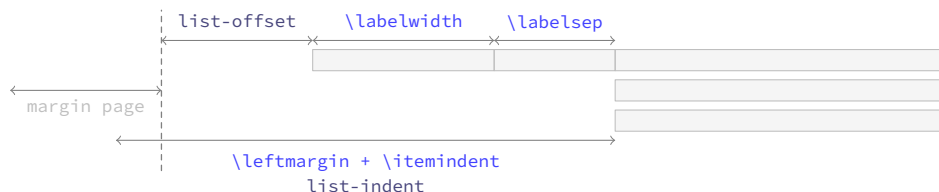


Figure 9: Representation of standard horizontal lengths in `list` environment.

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The key relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the label box is at `\leftmargin + \itemindent` minus `\labelwidth + \labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

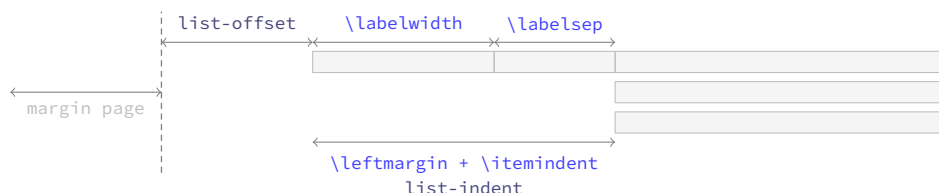


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

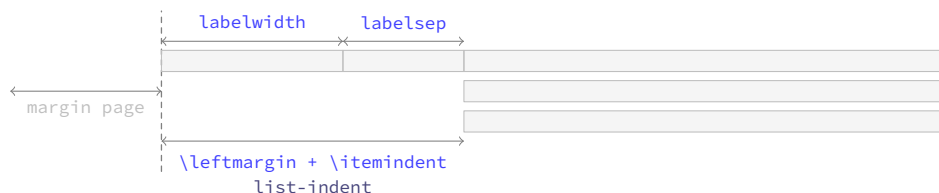


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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

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

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 (§11.34.2).

```

3072 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3073 {
3074   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3075   {
3076     \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3077     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3078   }
3079   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3080   {
3081     \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3082     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3083   }

```

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

```

3084   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3085   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3086   {
3087     \dim_set:Nn #6 { #1 + #2 - #4 }
3088     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3089   }
3090   {
3091     \dim_compare:nNnT { #4 } = { #1 + #2 }
3092     { \dim_set:Nn #6 { \c_zero_dim } }
3093     \dim_compare:nNnT { #4 } < { #1 + #2 }
3094     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3095     \dim_compare:nNnT { #4 } > { #1 + #2 }
3096     {
3097       \dim_set:Nn #6 { -#1 - #2 + #4 }
3098       \dim_set:Nn #6 { #6*-1 }
3099     }
3100     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3101   }
3102 }
3103 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

11.34.2 Setting second argument of the lists

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3104 \cs_set_protected:Npn \__enumext_tmp:n #1
3105 {
3106   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3107   {
3108     \__enumext_calc_hspace:ccccc
3109     { \l__enumext_labelwidth_#1_dim } { \l__enumext_labelsep_#1_dim }
3110     { \l__enumext_listoffset_#1_dim } { \l__enumext_leftmargin_tmp_#1_dim }
3111     { \l__enumext_leftmargin_#1_dim } { \l__enumext_itemindent_#1_dim }
3112     { \l__enumext_leftmargin_tmp_#1_bool }
3113     \clist_map_inline:nn
3114     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3115     { \dim_set_eq:cc {####1} { \l__enumext_####1_#1_dim } }
3116     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3117     { \skip_set_eq:cc {####1} { \l__enumext_####1_#1_skip } }
3118     \usecounter { enumX#1 }
3119     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \l__enumext_start_#1_int } - 1 } }
3120     \str_if_eq:nnTF {#1} { v }
3121     {
3122       \__enumext_keyans_redefine_item:
3123       \__enumext_keyans_make_label:
3124       \__enumext_keyans_ref:
3125       \__enumext_keyans_fake_item:
3126       \bool_if:cT { \l__enumext_show_length_#1_bool }
3127       {

```

```

3128         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3129     }
3130 }
3131 {
3132     \__enumext_redefine_item:
3133     \__enumext_make_label:
3134     \__enumext_standar_ref:
3135     \__enumext_fake_item:
3136     \bool_if:cT { l__enumext_show_length_#1_bool }
3137     {
3138         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3139     }
3140 }
3141 }
3142 }
3143 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_i: and others.)

```

\__enumext_list_arg_two_vii:
\__enumext_list_arg_two_viii:

```

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

```

3144 \cs_set_protected:Npn \__enumext_tmp:n #1
3145 {
3146     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3147     {
3148         \__enumext_calc_hspace:ccccc
3149         { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3150         { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3151         { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3152         { l__enumext_leftmargin_tmp_#1_bool }
3153         \clist_map_inline:nn
3154         { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3155         { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3156         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3157         { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3158         \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3159         \skip_zero:N \partopsep
3160         \usecounter { enumX#1 }
3161         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3162         \__enumext_starred_ref:
3163         \str_if_eq:nnTF {#1} { vii }
3164         {
3165             \__enumext_fake_item_vii:
3166             \bool_if:cT { l__enumext_show_length_vii_bool }
3167             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3168         }
3169         {
3170             \__enumext_fake_item_viii:
3171             \bool_if:cT { l__enumext_show_length_#1_bool }
3172             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3173         }
3174     }
3175 }
3176 \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:.)

11.35 The environment enumext

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

```

3177 \NewDocumentEnvironment{enumext}{0}{ }
3178 {
3179     \__enumext_safe_exec:
3180     \__enumext_parse_keys:n {#1}
3181     \__enumext_before_list:
3182     \__enumext_start_store_level:
3183     \__enumext_start_list:nn
3184     { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3185     {
3186         \use:c { __enumext_list_arg_two_ \__enumext_level: : }

```



```

3187     \__enumext_before_keys_exec:
3188     }
3189     \__enumext_after_args_exec:
3190     }
3191     {
3192     \__enumext_stop_list:
3193     \__enumext_stop_store_level:
3194     \__enumext_after_list:
3195     }

```

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

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first execute the function `__enumext_is_not_nested:` which will set the variable `\g__enumext_standar_bool` to “true” if the environment is not nested in `enumext*`, we increment the variable `\l__enumext_level_int` for the nesting levels and set the `\l__enumext_standar_bool` variable to “true”. Finally we set the variable `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level” of it using the function `__enumext_is_on_first_level:`.

```

3196 \cs_new_protected:Nn \__enumext_safe_exec:
3197 {
3198     \__enumext_internal_mini_page:
3199     \__enumext_is_not_nested:
3200     \int_incr:N \l__enumext_level_int
3201     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3202     { \msg_fatal:nn { enumext } { list-too-deep } }
3203     \bool_set_true:N \l__enumext_standar_bool
3204     \__enumext_is_on_first_level:
3205 }

```

(End of definition for `__enumext_safe_exec:`.)

`__enumext_parse_keys:n` The `__enumext_parse_store_keys:n` function will parse the `⟨keys⟩` passed to the optional environment argument `enumext` by levels only if present. First we clear the variable `\l__enumext_series_str` 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`, otherwise we will pass the `⟨keys⟩` to the inner levels of the environment and finally if the variable `\l__enumext_store_active_bool` established by the key `save-ans` is true we execute `__enumext_parse_store_keys:n` used by the key `save-key`.

```

3206 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3207 {
3208     \tl_if_novalue:nF {#1}
3209     {
3210         \str_clear:N \l__enumext_series_str
3211         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3212         {
3213             \keys_set:nn { enumext / level-1 } {#1}
3214             \__enumext_parse_series:n {#1}
3215             \__enumext_nested_base_line_fix:
3216         }
3217         {
3218             \exp_args:Ne \keys_set:nn
3219             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3220         }
3221         \__enumext_store_active_keys:n {#1}
3222     }
3223 }

```

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

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

`__enumext_stop_store_level:`

```

3224 \cs_new_protected:Nn \__enumext_start_store_level:
3225 {
3226     \bool_lazy_all:nT
3227     {
3228         { \bool_if_p:N \l__enumext_store_active_bool }
3229         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3230         { \bool_if_p:N \g__enumext_standar_bool }
3231     }
3232     {
3233         \int_compare:nNnT { \l__enumext_level_int } > { 1 }

```

```

3234         {
3235             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3236             \__enumext_store_level_open:
3237         }
3238     }

```

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

```

3239     \bool_lazy_all:nT
3240     {
3241         { \bool_if_p:N \l__enumext_store_active_bool }
3242         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3243         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3244     }
3245     {
3246         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3247         {
3248             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3249             \__enumext_store_level_open:
3250         }
3251     }
3252 }
3253 \cs_new_protected:Nn \__enumext_stop_store_level:
3254 {
3255     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3256     {
3257         \__enumext_store_level_close:
3258     }
3259 }

```

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

`__enumext_before_list:` The function `__enumext_before_list:` 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.

```

3260 \cs_new_protected:Nn \__enumext_before_list:
3261 {
3262     \__enumext_vspace_above:
3263     \__enumext_before_args_exec:

```

The function `__enumext_check_ans_active:` will handle the check answer mechanism, which will be activated with the `check-ans` key.

```

3264     \__enumext_check_ans_active:

```

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

```

3265     \dim_compare:nNnT
3266     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3267     {
3268         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3269         {
3270             \linewidth
3271             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3272             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3273         }

```

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

Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3274     \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3275     \int_gincr:N \g__enumext_minipage_stat_int
3276     \__enumext_mini_addvspace:
3277     \nointerlineskip\noindent
3278     \begin{\__enumext_mini_env*}
3279     { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }

```

```

3280     }
3281     \__enumext_multicols_start:
3282 }

```

(End of definition for __enumext_before_list:.)

`__enumext_multicols_start:` The function `__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=0pt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

3283 \cs_new_protected:Nn \__enumext_multicols_start:
3284 {
3285     \int_compare:nNnT
3286     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3287     {
3288         \dim_compare:nNnT
3289         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3290         {
3291             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3292             {
3293                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3294                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3295                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3296                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3297             }
3298         }
3299         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3300         \skip_zero:N \multicolsep
3301         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3302         {
3303             \dim_zero:N \columnseprule
3304         }

```

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.

```

3305     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3306     {
3307         \__enumext_multi_addvspace:
3308     }
3309     \raggedcolumns
3310     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3311 }
3312 }

```

(End of definition for __enumext_multicols_start:.)

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

```

3313 \cs_new_protected:Nn \__enumext_multicols_stop:
3314 {
3315     \int_compare:nNnT
3316     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3317     {
3318         \end{multicols}
3319         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3320         {
3321             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3322         }
3323     }
3324 }

```

(End of definition for __enumext_multicols_stop:.)

`__enumext_after_list:` The function `__enumext_after_list:` will check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3325 \cs_new_protected:Nn \__enumext_after_list:
3326 {
3327   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3328   {
3329     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3330     {
3331       \msg_warning:nn { enumext } { missing-miniright }
3332       \miniright
3333     }
3334     \int_gzero:N \g__enumext_minipage_stat_int
3335     \end{__enumext_mini_env*}
3336     \par\addvspace { \l__enumext_minipage_after_skip }
3337   }
3338   { \__enumext_multicols_stop: }

```

If the `check-ans` key is active, we set the boolean variable `\g__enumext_check_ans_show_bool` to true and copy the “*store name*” to the variable `\g__enumext_store_name_tl`.

```

3339   \__enumext_check_ans_key_hook:

```

Now apply the `{<code>}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present, set `\l__enumext_standar_bool` to false and save the *current value* of the counter for `series`, `resume` and `resume*` keys.

```

3340   \__enumext_after_stop_list:
3341   \__enumext_vspace_below:
3342   \bool_set_false:N \l__enumext_standar_bool
3343   \__enumext_resume_save_counter:
3344 }

```

(End of definition for `__enumext_after_list:`.)

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`.

```

3345 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

11.36 The environment `keyans`

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

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

```

3346 \NewDocumentEnvironment{keyans}{0}{ }
3347 {
3348   \__enumext_keyans_safe_exec:
3349   \__enumext_keyans_parse_keys:n {#1}
3350   \__enumext_before_list_v:
3351   \__enumext_start_list:nn
3352   { \tl_use:N \l__enumext_label_v_tl }
3353   {
3354     \__enumext_list_arg_two_v:
3355     \__enumext_before_keys_exec_v:
3356   }
3357   \__enumext_after_args_exec_v:
3358 }
3359 {
3360   \__enumext_check_starred_cmd:n { item }
3361   \__enumext_stop_list:
3362   \__enumext_after_list_v:
3363 }

```

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

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

```

3364 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3365 {
3366   \bool_if:NF \l__enumext_store_active_bool
3367   {
3368     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3369   }
3370   \int_incr:N \l__enumext_keyans_level_int

```

```

3371 \bool_set_true:N \__enumext_keyans_env_bool
3372 \__enumext_keyans_start_line:
3373 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3374 \bool_set_false:N \__enumext_store_active_bool
3375 \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
3376 {
3377   \msg_error:nn { enumext } { keyans-nested }
3378 }
3379 \int_compare:nNnT { \__enumext_level_int } > { 1 }
3380 {
3381   \msg_error:nn { enumext } { keyans-wrong-level }
3382 }
3383 }

```

(End of definition for __enumext_keyans_safe_exec:.)

```

\__enumext_keyans_parse_keys:n Parse [⟨key = val⟩] for keyans environment.
3384 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3385 {
3386   \keys_set:nn { enumext / keyans } {#1}
3387 }

```

(End of definition for __enumext_keyans_parse_keys:n.)

__enumext_before_list_v: The function __enumext_before_list_v: will add the *vertical spacing* above the environment if the *above* key is active next to the *⟨code⟩* defined by the *before* key if it is active.

```

3388 \cs_new_protected:Nn \__enumext_before_list_v:
3389 {
3390   \__enumext_vspace_above_v:
3391   \__enumext_before_args_exec_v:

```

When the *mini-env* key is active it will set the value of the \l__enumext_minipage_right_v_dim to be the *width* of the *__enumext_mini_env** environment on the *left side*, using this value together with the value of the \l__enumext_minipage_hsep_v_dim set by the *mini-sep* key, the value of \l__enumext_minipage_left_v_dim will be set, which will be the *width* of *__enumextt_mini_env** environment on the *right side*, always having \linewidth as the maximum width between them.

```

3392 \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3393 {
3394   \dim_set:Nn \l__enumext_minipage_left_v_dim
3395   {
3396     \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3397   }

```

The boolean variable \l__enumext_minipage_active_v_bool will be activated and the integer variable \g__enumext_minipage_stat_int used by the \mini_{right} command will be incremented, then the function __enumext_keyans_mini_addvspace: is called and the *__enumext_mini_env** environment on *left side* will be initialized followed by the *vertical spacing* \l__enumext_minipage_left_skip. Here we use the plain TeX macro \nointerlineskip to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3398 \bool_set_true:N \l__enumext_minipage_active_v_bool
3399 \int_gincr:N \g__enumext_minipage_stat_int
3400 \__enumext_keyans_mini_addvspace:
3401 \nointerlineskip\noindent
3402 \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3403 }

```

After these actions, the __enumext_keyans_multicols_start: function is called to handle the *multicols* environment.

```

3404 \__enumext_keyans_multicols_start:
3405 }

```

(End of definition for __enumext_before_list_v:.)

__enumext_keyans_multicols_start: The function __enumext_keyans_multicols_start: will start the *multicols* environment according to the value passed by the *columns* key.

```

3406 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3407 {
3408   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3409   {

```

Set the default value for `\columnsep` when `columns-sep` key is `opt`.

```

3410     \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3411     {
3412         \dim_set:Nn \l__enumext_columns_sep_v_dim
3413         {
3414             (
3415                 \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3416             ) / \l__enumext_columns_v_int
3417             - \l__enumext_listoffset_v_dim
3418         }
3419     }
3420     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

3421     \skip_zero:N \multicolsep
3422     \dim_zero:N \columnseprule

```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `__enumext_keyans_multi_addvspace:` and apply our “vertical adjust spacing”, then start the `multicols` environment.

```

3423     \bool_if:NF \l__enumext_minipage_active_v_bool
3424     {
3425         \__enumext_keyans_multi_addvspace:
3426     }
3427     \raggedcolumns
3428     \begin{multicols}{ \l__enumext_columns_v_int }
3429 }
3430 }

```

(End of definition for `__enumext_keyans_multicols_start:`)

`__enumext_keyans_multicols_stop:`

The function `__enumext_keyans_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3431 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3432 {
3433     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3434     {
3435         \end{multicols}
3436         \bool_if:NF \l__enumext_minipage_active_v_bool
3437         {
3438             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3439         }
3440     }
3441 }

```

(End of definition for `__enumext_keyans_multicols_stop:`)

`__enumext_after_list_v:`

The function `__enumext_after_list_v:` will check the state of the boolean variable `\l__enumext_minipage_active_v_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini-env*` environment has not been closed), then close `__enumext_mini-env*` and add the vertical adjustment space `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3442 \cs_new_protected:Nn \__enumext_after_list_v:
3443 {
3444     \bool_if:NTF \l__enumext_minipage_active_v_bool
3445     {
3446         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3447         {
3448             \msg_warning:nn { enumext } { missing-miniright }
3449             \miniright
3450         }
3451         \int_gzero:N \g__enumext_minipage_stat_int
3452         \end{__enumext_mini-env*}
3453         \par\addvspace{ \l__enumext_minipage_after_skip }
3454     }
3455     { \__enumext_keyans_multicols_stop: }

```

Finally we will apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

3456 \bool_set_false:N \l__enumext_keyans_env_bool
3457 \__enumext_after_stop_list_v:
3458 \__enumext_vspace_below_v:
3459 }

```

(End of definition for `__enumext_after_list_v:`)

11.37 The environment `keyanspic` and `\anspic`

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

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

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

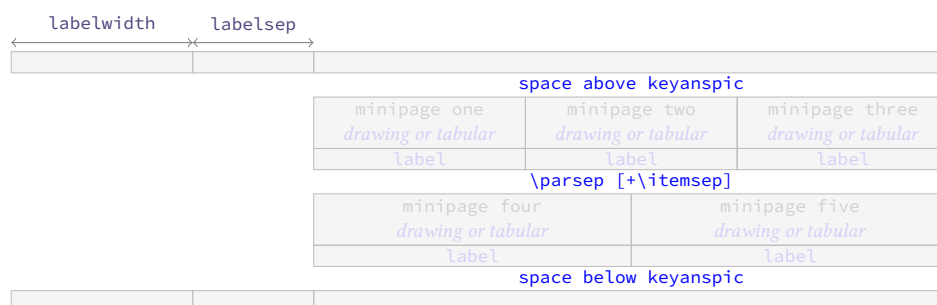


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

11.37.1 The command `\anspic`

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

```

3460 \NewDocumentCommand \anspic { s o +m }
3461 {

```

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

```

3462 \bool_if:NF \l__enumext_store_active_bool
3463 {
3464 \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3465 }
3466 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3467 {
3468 \msg_error:nn { enumext } { keyanspic-wrong-level }
3469 }
3470 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3471 {
3472 \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3473 }

```

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

```

3474 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3475 {
3476 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3477 }
3478 }

```

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

`__enumext_keyans_anspic_code:nnn`

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

```

3479 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3480 {
3481 \stepcounter { enumXvi }

```



```

3482 #3 \\
3483 \bool_if:nT { #1 }
3484 {
3485   \__enumext_keyans_addto_prop:n { #2 }
3486   \__enumext_keyans_store_ref:
3487   \__enumext_keyans_addto_seq:n { #2 }
3488   \int_gincr:N \g__enumext_check_starred_cmd_int
3489   \bool_lazy_or:nnT
3490   { \bool_if_p:N \l__enumext_show_answer_bool }
3491   { \bool_if_p:N \l__enumext_show_position_bool }
3492   {
3493     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3494     \__enumext_keyans_show_left:n { #2 }
3495     \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3496   }
3497 }
3498 \tl_use:N \l__enumext_label_font_style_v_tl
3499 \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3500 }

```

(End of definition for __enumext_keyans_anspic_code:nnn.)

11.37.2 The environment keyanspic

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

```

3501 \NewDocumentEnvironment{keyanspic}{o}
3502 {
3503   \__enumext_keyans_pic_safe_exec:
3504   \__enumext_start_list:nn
3505   { }
3506   {
3507     \__enumext_keyans_pic_arg_two:
3508   }

```

We apply the “adjusted” vertical spacing above the environment

```

3509   \vspace { \l__enumext_keyans_pic_above_skip }
3510 }

```

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

```

3511 {
3512   \tl_if_novalue:nTF { #1 }
3513   {
3514     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3515   }
3516   { \__enumext_keyans_pic_do:n { #1 } }
3517   \__enumext_stop_list:
3518   \__enumext_check_starred_cmd:n { anspic }
3519   \setcounter { enumXvi } { 0 }
3520   \vspace { \l__enumext_topsep_v_skip }
3521   %\bool_set_false:N \l__enumext_store_active_bool
3522 }

```

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

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

```

3523 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3524 {
3525   \int_incr:N \l__enumext_keyans_pic_level_int
3526   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3527   {
3528     \msg_error:nn { enumext } { keyanspic-nested }
3529   }
3530   \__enumext_keyans_start_line:
3531 }

```

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

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

```

3532 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3533 {
3534   \dim_compare:nNtT { #1 } < { 0pt }
3535   { \skip_set:Nn #1 { -#1 } }
3536 }

```

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

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

```

3537 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3538 {

```

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

```

3539   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3540   \__enumext_list_arg_two_v:

```

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

```

3541   \skip_add:Nn \parsep { \itemsep }
3542   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3543   \dim_zero:N \labelwidth
3544   \dim_zero:N \listparindent
3545   \dim_zero:N \labelsep
3546   \skip_zero:N \partopsep
3547   \skip_zero:N \itemsep

```

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

```

3548   \__enumext_keyans_pic_skip_abs:N \parsep
3549   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3550   {
3551     \box_dp:N \strutbox
3552     + \l__enumext_topsep_v_skip
3553     - \parsep
3554   }
3555   \__enumext_item_std:w \scan_stop:
3556 }

```

(End of definition for `__enumext_keyans_pic_arg_two:`.)

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

```

3557 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3558 {
3559   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3560 }
3561 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

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

```

3562 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3563 {
3564   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3565   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3566   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3567   \int_step_inline:nnn
3568   { \l__enumext_keyans_pic_above_int + 1 }
3569   { \l__enumext_keyans_pic_below_int }
3570   {
3571     \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3572     \centering

```

```

3573         \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3574     \__enumext_endminipage:
3575 }
3576 \par
3577 }

```

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

11.38 The horizontal environments

Generating horizontal list environments is NOT as simple as standard \TeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

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

11.38.1 Functions for item box width

We set the default value for the width of the box containing the content of the items and create `\itemwidth` in a public form.

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:

```

```

3578 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3579 {
3580     \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3581     {
3582         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3583         {
3584             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3585             / \l__enumext_columns_vii_int
3586         }
3587     }
3588     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3589     \dim_set:Nn \l__enumext_item_width_vii_dim
3590     {
3591         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3592         / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3593         - \l__enumext_labelsep_vii_dim
3594     }
3595     \dim_zero_new:N \itemwidth
3596 }
3597 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3598 {
3599     \dim_compare:nNt { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3600     {
3601         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3602         {
3603             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3604             / \l__enumext_columns_viii_int
3605         }
3606     }
3607     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3608     \dim_set:Nn \l__enumext_item_width_viii_dim
3609     {
3610         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3611         / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3612         - \l__enumext_labelsep_viii_dim
3613     }
3614     \dim_zero_new:N \itemwidth
3615 }

```

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

11.38.2 Functions for join item columns

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(\langle columns \rangle)` will be stored together with the value of `\itemwidth`.

```

\__enumext_starred_joined_item_vii:n
\__enumext_starred_joined_item_viii:n

```

```

3616 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3617 {

```

```

3618 \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3619 \int_compare:nNt { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3620 {
3621   \msg_warning:nnee { enumext } { item-joined }
3622   { \int_use:N \l__enumext_joined_item_vii_int }
3623   { \int_use:N \l__enumext_columns_vii_int }
3624   \int_set:Nn \l__enumext_joined_item_vii_int
3625   {
3626     \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3627   }
3628 }
3629 \int_compare:nNt
3630 { \l__enumext_joined_item_vii_int }
3631 >
3632 { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3633 {
3634   \msg_warning:nnee { enumext } { item-joined-columns }
3635   { \int_use:N \l__enumext_joined_item_vii_int }
3636   {
3637     \int_eval:n
3638     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3639   }
3640   \int_set:Nn \l__enumext_joined_item_vii_int
3641   {
3642     \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3643   }
3644 }
3645 \int_compare:nNtF { \l__enumext_joined_item_vii_int } > { 1 }
3646 {
3647   \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3648   \int_decr:N \l__enumext_joined_item_aux_vii_int
3649   \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3650   \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3651   \dim_set:Nn \l__enumext_joined_width_vii_dim
3652   {
3653     \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3654     + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3655         + \l__enumext_columns_sep_vii_dim
3656         ) * \l__enumext_joined_item_aux_vii_int
3657   }
3658   \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3659 }
3660 {
3661   \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3662   \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3663 }
3664 }
3665 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3666 {
3667   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3668   \int_compare:nNt { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3669   {
3670     \msg_warning:nnee { enumext } { item-joined }
3671     { \int_use:N \l__enumext_joined_item_viii_int }
3672     { \int_use:N \l__enumext_columns_viii_int }
3673     \int_set:Nn \l__enumext_joined_item_viii_int
3674     {
3675       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3676     }
3677   }
3678   \int_compare:nNt
3679   { \l__enumext_joined_item_viii_int }
3680   >
3681   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3682   {
3683     \msg_warning:nnee { enumext } { item-joined-columns }
3684     { \int_use:N \l__enumext_joined_item_viii_int }
3685     {
3686       \int_eval:n
3687       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3688     }
3689   }

```

```

3689     \int_set:Nn \l__enumext_joined_item_viii_int
3690     {
3691         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3692     }
3693 }
3694 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3695 {
3696     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3697     \int_decr:N \l__enumext_joined_item_aux_viii_int
3698     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3699     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3700     \dim_set:Nn \l__enumext_joined_width_viii_dim
3701     {
3702         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3703         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3704             + \l__enumext_columns_sep_viii_dim
3705             ) * \l__enumext_joined_item_aux_viii_int
3706     }
3707     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3708 }
3709 {
3710     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3711     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3712 }
3713 }

```

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

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

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

```

3714 \cs_new_protected:Nn \__enumext_start_mini_vii:
3715 {
3716     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3717     {
3718         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3719         {
3720             \linewidth
3721             - \l__enumext_minipage_right_vii_dim
3722             - \l__enumext_minipage_hsep_vii_dim
3723         }
3724         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3725         \dim_gset_eq:NN
3726             \g__enumext_minipage_right_vii_dim
3727             \l__enumext_minipage_right_vii_dim
3728         \__enumext_mini_addvspace_vii:
3729         \nointerlineskip\noindent
3730         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3731     }
3732 }

```

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

```

3733 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3734 {
3735     \bool_if:NT \l__enumext_minipage_active_vii_bool
3736     {
3737         \end{__enumext_mini_env*}
3738         \hfill
3739         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3740     }
3741 }

```

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

3742 \__enumext_after_env:nn {enumext*}
3743 {
3744   \bool_if:NT \g__enumext_minipage_active_vii_bool
3745   {
3746     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3747     \par\addvspace { \g__enumext_minipage_right_skip }
3748     \bool_if:NF \g__enumext_minipage_center_vii_bool
3749     {
3750       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3751       {
3752         \centering
3753       }
3754     }
3755     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3756     {
3757       \tl_use:N \g__enumext_miniright_code_vii_tl
3758     }
3759     \box_use_drop:N \l__enumext_miniright_code_vii_box
3760     \end{__enumext_mini_env*}
3761     \par\addvspace{ \g__enumext_minipage_after_skip }
3762   }
3763   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3764   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3765   \tl_gclear:N \g__enumext_miniright_code_vii_tl
3766   \dim_gzero:N \g__enumext_minipage_right_vii_dim
3767   \bool_gset_false:N \g__enumext_starred_bool
3768 }

```

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

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

```

3769 \cs_new_protected:Nn \__enumext_start_mini_viii:
3770 {
3771   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3772   {
3773     \dim_set:Nn \l__enumext_minipage_left_viii_dim
3774     {
3775       \linewidth
3776       - \l__enumext_minipage_right_viii_dim
3777       - \l__enumext_minipage_hsep_viii_dim
3778     }
3779     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3780     \dim_gset_eq:NN
3781       \g__enumext_minipage_right_viii_dim
3782       \l__enumext_minipage_right_viii_dim
3783     \__enumext_mini_addvspace_viii:
3784     \nointerlineskip\noindent
3785     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3786   }
3787 }
3788 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3789 {
3790   \bool_if:NT \l__enumext_minipage_active_viii_bool
3791   {
3792     \end{__enumext_mini_env*}
3793     \hfill
3794     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3795   }
3796 }
3797 \__enumext_after_env:nn {keyans*}
3798 {
3799   \bool_if:NT \g__enumext_minipage_active_viii_bool
3800   {
3801     \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3802     \par\addvspace { \g__enumext_minipage_right_skip }
3803     \bool_if:NF \g__enumext_minipage_center_viii_bool
3804     {
3805       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3806       {
3807         \centering

```

```

3808         }
3809     }
3810     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3811     {
3812         \tl_use:N \g__enumext_miniright_code_viii_tl
3813     }
3814     \box_use_drop:N \l__enumext_miniright_code_viii_box
3815     \end{__enumext_mini_env*}
3816     \par\addvspace{ \g__enumext_minipage_after_skip }
3817 }
3818 \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3819 \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3820 \tl_gc_clear:N \g__enumext_miniright_code_viii_tl
3821 \dim_gzero:N \g__enumext_minipage_right_viii_dim
3822 }

```

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

11.39 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to \noindent and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later.

```

3823 \NewDocumentEnvironment{enumext*}{ o }
3824 {
3825     \__enumext_safe_exec_vii:
3826     \__enumext_parse_keys_vii:n {#1}
3827     \__enumext_starred_columns_set_vii:
3828     \__enumext_before_list_vii:
3829     \__enumext_start_store_level_vii:
3830     \__enumext_start_list:nn { }
3831     {
3832         \__enumext_list_arg_two_vii:
3833         \__enumext_before_keys_exec_vii:
3834     }
3835     \item[] \scan_stop:
3836     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3837     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3838 }
3839 {
3840     \__enumext_stop_item_tmp_vii:
3841     \__enumext_remove_extra_parsep_vii:
3842     \__enumext_stop_list:
3843     \__enumext_stop_store_level_vii:
3844     \__enumext_after_list_vii:
3845 }

```

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

__enumext_safe_exec_vii: First check the maximum nesting level for the **enumext*** environment then set the vars \l__enumext_starred_bool and \g__enumext_starred_bool.

```

3846 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3847 {
3848     \__enumext_internal_mini_page:
3849     \__enumext_is_not_nested:
3850     \int_incr:N \l__enumext_level_h_int
3851     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3852     {
3853         \msg_error:nn { enumext } { nested }
3854     }
3855     \bool_set_true:N \l__enumext_starred_bool
3856     \__enumext_is_on_first_level:
3857 }

```

(End of definition for __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n Parse [*key* = *val*] for **enumext***. If the variable \l__enumext_store_active_bool is true it will call the functions __enumext_parse_serie:n and __enumext_store_active_keys_vii:n and reprocess the *keys* to pass them to the storage *sequence*.

```

3858 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3859 {

```



```

3860     \tl_if_novalue:nF {#1}
3861     {
3862         \str_clear:N \l__enumext_series_str
3863         \keys_set:nn { enumext / enumext* } {#1}
3864         \__enumext_parse_series:n {#1}
3865         \__enumext_store_active_keys_vii:n {#1}
3866         \__enumext_nested_base_line_fix:
3867     }
3868 }

```

(End of definition for __enumext_parse_keys_vii:n.)

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

```

3869 \cs_new_protected:Nn \__enumext_before_list_vii:
3870 {
3871     \__enumext_vspace_above_vii:
3872     \__enumext_check_ans_active:
3873     \__enumext_before_args_exec_vii:
3874     \__enumext_start_mini_vii:
3875 }

```

(End of definition for __enumext_before_list_vii:.)

`__enumext_after_list_vii:` The function `__enumext_after_list:` first call the function `__enumext_stop_mini_vii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present. Finally set false the vars `\g__enumext_starred_bool` and `\l__enumext_starred_bool`, save the *current value* of the counter in `\g__enumext_resume_vii_int` for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

3876 \cs_new_protected:Nn \__enumext_after_list_vii:
3877 {
3878     \__enumext_stop_mini_vii:
3879     \__enumext_after_stop_list_vii:
3880     \__enumext_check_ans_key_hook:
3881     \__enumext_vspace_below_vii:
3882     \bool_set_false:N \l__enumext_starred_bool
3883     \__enumext_resume_save_counter:
3884 }

```

(End of definition for __enumext_after_list_vii:.)

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

```

3885 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3886 {
3887     \bool_if:NT \l__enumext_store_active_bool
3888     {
3889         \int_compare:nNt { \l__enumext_level_int } > { 0 }
3890         {
3891             \__enumext_store_level_open_vii:
3892         }
3893     }
3894 }
3895 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3896 {
3897     \bool_if:NT \l__enumext_store_active_bool
3898     {
3899         \int_compare:nNt { \l__enumext_level_int } > { 0 }
3900         {
3891             \__enumext_store_level_close_vii:
3892         }
3893     }
3894 }

```

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

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

`__enumext_start_item_tmp_vii:`

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

```
3905 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3906 {
3907   \__enumext_stop_item_tmp_vii:
3908   \int_incr:N \l__enumext_item_column_pos_vii_int
3909   \int_gincr:N \g__enumext_item_count_all_vii_int
3910   \__enumext_item_peek_args_vii:
3911 }
```

(End of definition for `__enumext_start_item_tmp_vii:`)

`__enumext_item_peek_args_vii:`

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

```
3912 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3913 {
3914   \peek_meaning:NTF (
3915     { \__enumext_joined_item_vii:w }
3916     { \__enumext_joined_item_vii:w (1) }
3917 }
```

(End of definition for `__enumext_item_peek_args_vii:`)

`__enumext_joined_item_vii:w`

The function `__enumext_joined_item_vii:w` will first call the function `__enumext_starred_joined_item_vii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “*”, if it is present we will call the function `__enumext_starred_item_vii:w` otherwise we will call the function `__enumext_standar_item_vii:w`.

```
3918 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3919 {
3920   \__enumext_starred_joined_item_vii:n {#1}
3921   \peek_meaning_remove:NTF *
3922     { \__enumext_starred_item_vii:w }
3923     { \__enumext_standar_item_vii:w }
3924 }
```

(End of definition for `__enumext_joined_item_vii:w`)

`__enumext_standar_item_vii:w`

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

```
3925 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3926 {
3927   \bool_set_false:N \l__enumext_item_starred_vii_bool
3928   \peek_meaning:NTF [
3929     {
3930       \bool_set_eq:NN
3931       \l__enumext_wrap_label_vii_bool
3932       \l__enumext_wrap_label_opt_vii_bool
3933       \__enumext_start_item_vii:w
3934     }
3935     {
3936       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3937       \legacy_if_set_true:n { @noitemarg }
3938       \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ]
3939     }
3940 }
```

(End of definition for `__enumext_standar_item_vii:w`)

```

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

```

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

```

3941 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3942 {
3943   \bool_set_true:N \l__enumext_item_starred_vii_bool
3944   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3945   \peek_meaning:NTF [
3946     { \__enumext_starred_item_vii_aux_i:w }
3947     { \__enumext_starred_item_vii_aux_ii:w }
3948   }
3949   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3950   {
3951     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3952     \__enumext_starred_item_vii_aux_ii:w
3953   }
3954   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
3955   {
3956     \peek_meaning:NTF [
3957       { \__enumext_starred_item_vii_aux_iii:w }
3958       {
3959         \dim_set_eq:NN
3960         \l__enumext_item_symbol_sep_vii_dim
3961         \l__enumext_labelsep_vii_dim
3962         \legacy_if_set_true:n { @noitemarg }
3963         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3964       }
3965     }
3966     \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3967     {
3968       \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3969       \legacy_if_set_true:n { @noitemarg }
3970       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3971     }

```

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

11.39.2 Real definition of `\item` in `enumext*`

```
\__enumext_start_item_vii:w
```

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment.

The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to `__enumext_stop_item_vii:` which we will define later and add the `hyperref` compatible `enumXvii` counter, after that we will start capturing the item content in a box. Here need setting the `\if@hyper@item` switch to “true” for `hyperref` compatible. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier.

```

3972 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
3973 {
3974   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
3975   \legacy_if:nT { @noitemarg }
3976   {
3977     \legacy_if_set_false:n { @noitemarg }
3978     \legacy_if:nT { @nmbrrlist }
3979     {
3980       \bool_if:NT \l__enumext_hyperref_bool
3981       {
3982         \legacy_if_set_true:n { @hyper@item }
3983       }
3984       \refstepcounter{enumXvii}
3985       \bool_if:NT \l__enumext_check_answers_bool
3986       {
3987         \int_gincr:N \g__enumext_item_number_int
3988         \bool_set_true:N \l__enumext_item_number_bool
3989       }
3990     }
3991   }

```

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

environment and execute the `\listparindent` key which will be equal to `\parindent`, the `\parsep` key which will be equal to `\parskip` and the `\itemindent` key.

```

3992 \group_begin:
3993 \lrbox{ \l__enumext_item_text_vii_box }
3994 \bool_if:NF \l__enumext_footnotes_key_bool
3995 {
3996   \__enumext_renew_footnote:
3997 }
3998 \bool_if:NT \l__enumext_item_starred_vii_bool
3999 {
4000   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4001   {
4002     \tl_gset_eq:NN
4003       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4004   }
4005   \mode_leave_vertical:
4006   \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4007   \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4008   \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4009   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4010 }
4011 \group_begin:
4012 \tl_use:N \l__enumext_label_font_style_vii_tl
4013 \bool_if:NTF \l__enumext_wrap_label_vii_bool
4014 {
4015   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4016     { \__enumext_wrapper_label_vii:n {#1} }
4017 }
4018 {
4019   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4020 }
4021 \group_end:
4022 \skip_horizontal:N \l__enumext_labelsep_vii_dim
4023 \tl_use:N \l__enumext_after_list_args_vii_tl
4024 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4025   \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4026   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4027   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4028 }

```

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

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

```

4029 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4030 {
4031   \__enumext_endminipage:
4032   \endlrbox
4033   \group_end:
4034   \box_set_wd:Nn \l__enumext_item_text_vii_box
4035   {
4036     \l__enumext_joined_width_vii_dim
4037     + \l__enumext_labelwidth_vii_dim
4038     + \l__enumext_labelsep_vii_dim
4039   }
4040   \int_set:Nn \hbadness { 10000 }
4041   \box_use_drop:N \l__enumext_item_text_vii_box
4042   \bool_if:NF \l__enumext_footnotes_key_bool
4043   {
4044     \__enumext_print_footnote:
4045   }
4046   \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4047   {
4048     \par\noindent
4049     \int_zero:N \l__enumext_item_column_pos_vii_int
4050   }
4051   { \hspace{ \l__enumext_columns_sep_vii_dim } }
4052 }

```

(End of definition for `__enumext_stop_item_vii:`.)

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

```

4053 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4054 {
4055   \int_compare:nNtT
4056   {
4057     \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4058   }
4059   =
4060   { 0 }
4061   {
4062     \par
4063     \vspace{ -\l__enumext_itemsep_vii_skip }
4064     \int_gzero:N \g__enumext_item_count_all_vii_int
4065   }
4066 }

```

(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`.

```

4067 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

11.40 The environment `keyans*`

`keyans*` First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii:` equal to `\noindent` and next to `\item` equal to `__enumext_start_item_tmp_viii:` which we will redefine later.

```

4068 \NewDocumentEnvironment{keyans*}{o}{
4069 {
4070   \__enumext_safe_exec_viii:
4071   \__enumext_parse_keys_viii:n {#1}
4072   \__enumext_starred_columns_set_viii:
4073   \__enumext_before_list_viii:
4074   \__enumext_start_list:nn { }
4075   {
4076     \__enumext_list_arg_two_viii:
4077     \__enumext_before_keys_exec_viii:
4078   }
4079   \item[] \scan_stop:
4080   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4081   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4082 }
4083 {
4084   \__enumext_stop_item_tmp_viii:
4085   \__enumext_remove_extra_parsep_viii:
4086   \__enumext_check_starred_cmd:n { item }
4087   \__enumext_stop_list:
4088   \__enumext_after_list_viii:
4089 }

```

(End of definition for `keyans*`. This function is documented on page 13.)

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

```

4090 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4091 {
4092   \int_incr:N \l__enumext_keyans_level_h_int
4093   \int_compare:nNtT { \l__enumext_keyans_level_h_int } > { 1 }
4094   {
4095     \msg_error:nn { enumext } { nested }
4096   }
4097   \__enumext_keyans_start_line:
4098   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4099   \bool_set_false:N \l__enumext_store_active_bool
4100   \int_compare:nNtT { \l__enumext_level_int } > { 1 }
4101   {
4102     \msg_error:nn { enumext } { keyans-wrong-level }
4103   }
4104 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

4105 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4106 {
4107     \tl_if_novalue:nF {#1}
4108     {
4109         \keys_set:nn { enumext / keyans* } {#1}
4110     }
4111 }

```

(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*.

```

4112 \cs_new_protected:Nn \__enumext_before_list_viii:
4113 {
4114     \__enumext_vspace_above_viii:
4115     \__enumext_before_args_exec_viii:
4116     \__enumext_start_mini_viii:
4117 }

```

(End of definition for `__enumext_before_list_viii:`.)

`__enumext_after_list_viii:` The function `__enumext_after_list:` first call the function `__enumext_stop_mini_viii:`, then apply the `{code}` handled by the *after* key together with the *vertical space* handled by the *below* key if they are present.

```

4118 \cs_new_protected:Nn \__enumext_after_list_viii:
4119 {
4120     \__enumext_stop_mini_viii:
4121     \__enumext_after_stop_list_viii:
4122     \__enumext_vspace_below_viii:
4123 }

```

(End of definition for `__enumext_after_list_viii:`.)

11.40.1 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_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

4124 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4125 {
4126     \__enumext_stop_item_tmp_viii:
4127     \int_incr:N \l__enumext_item_column_pos_viii_int
4128     \int_gincr:N \g__enumext_item_count_all_viii_int
4129     \__enumext_item_peek_args_viii:
4130 }

```

(End of definition for `__enumext_start_item_tmp_viii:`.)

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

```

4131 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4132 {
4133     \peek_meaning:NTF (
4134         { \__enumext_joined_item_viii:w }
4135         { \__enumext_joined_item_viii:w (1) }
4136     }

```

(End of definition for `__enumext_item_peek_args_viii:`.)

`__enumext_joined_item_viii:w` The function `__enumext_joined_item_viii:w` will first call the function `__enumext_starred_joined_item_viii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument “*”, if it is present we will call the function `__enumext_starred_item_viii:w` otherwise we will call the function `__enumext_standar_item_viii:w`.

```

4137 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4138 {
4139     \__enumext_starred_joined_item_viii:n {#1}
4140     \peek_meaning_remove:NTF *
4141     { \__enumext_starred_item_viii:w }
4142     { \__enumext_standar_item_viii:w }
4143 }

```

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

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

```

4144 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4145 {
4146     \bool_set_false:N \l__enumext_item_starred_viii_bool
4147     \peek_meaning:NTF [
4148     {
4149         \bool_set_eq:NN
4150         \l__enumext_wrap_label_viii_bool
4151         \l__enumext_wrap_label_opt_viii_bool
4152         \__enumext_start_item_viii:w
4153     }
4154     {
4155         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4156         \legacy_if_set_true:n { @noitemarg }
4157         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4158     }
4159 }

```

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

`__enumext_starred_item_viii:w` The function `__enumext_starred_item_viii:w` together with the specified auxiliary functions `aux_i:w` and `aux_ii:w` execute `\item*` and `\item*[⟨content⟩]`.

```

\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
4160 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4161 {
4162     \bool_set_true:N \l__enumext_item_starred_viii_bool
4163     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4164     \peek_meaning:NTF [
4165     { \__enumext_starred_item_viii_aux_i:w }
4166     { \__enumext_starred_item_viii_aux_ii:w }
4167 }

```

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`.

```

4168 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4169 {
4170     \tl_clear:N \l__enumext_store_current_label_tl
4171     \tl_if_no_value:nF { #1 }
4172     {
4173         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4174         {
4175             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4176             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4177         }
4178         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4179     }
4180     \__enumext_starred_item_viii_aux_ii:w
4181 }

```



```

4182 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4183 {
4184   \legacy_if_set_true:n { @noitemarg }
4185   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4186 }

```

(End of definition for `__enumext_starred_item_viii:w`, `__enumext_starred_item_viii_aux_ii:w`, and `__enumext_starred_item_viii_aux_ii:w`.)

`__enumext_starred_item_exec:`

The function `__enumext_starred_item_exec:` will be in charge of storing the current *label* for `\item*` followed by the `[content]` for `\item*[content]` if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

4187 \cs_new_protected:Nn \__enumext_starred_item_exec:
4188 {
4189   \tl_put_left:Ne \__enumext_store_current_label_tl { \__enumext_label_viii_tl }
4190   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
4191   \__enumext_keyans_store_ref:
4192   \tl_put_left:Ne \__enumext_store_current_label_tl { \item }
4193   \__enumext_keyans_addto_seq_link:
4194   \int_gincr:N \g__enumext_check_starred_cmd_int
4195   \bool_if:NT \__enumext_show_answer_bool
4196   {
4197     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4198   }
4199   \bool_if:NT \__enumext_show_position_bool
4200   {
4201     \tl_set:Ne \__enumext_mark_answer_sym_tl
4202     {
4203       \group_begin:
4204       \exp_not:N \normalfont
4205       \exp_not:N \footnotesize [ \int_eval:n
4206         {
4207           \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
4208         }
4209       ]
4210       \group_end:
4211     }
4212     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4213   }
4214 }

```

(End of definition for `__enumext_starred_item_exec:`.)

Real definition of `\item in keyans*`

`__enumext_start_item_viii:w`

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

```

4215 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4216 {
4217   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4218   \legacy_if:nT { @noitemarg }
4219   {
4220     \legacy_if_set_false:n { @noitemarg }
4221     \legacy_if:nT { @nmbrlist }
4222     {
4223       \bool_if:NT \__enumext_hyperref_bool
4224       {
4225         \legacy_if_set_true:n { @hyper@item }
4226       }
4227       \refstepcounter{enumXviii}
4228     }
4229   }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment.

```

4230   \group_begin:
4231   \lrbox{ \__enumext_item_text_viii_box }
4232   \bool_if:NF \__enumext_footnotes_key_bool
4233   {
4234     \__enumext_renew_footnote:
4235   }
4236   \bool_if:NT \__enumext_item_starred_viii_bool
4237   {

```

```

4238         \__enumext_starred_item_exec:
4239     }
4240 \group_begin:
4241     \tl_use:N \__enumext_label_font_style_viii_tl
4242     \bool_if:NTF \__enumext_wrap_label_viii_bool
4243     {
4244         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
4245         { \__enumext_wrapper_label_viii:n {#1} }
4246     }
4247     {
4248         \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]{ #1
4249     }
4250 \group_end:
4251 \skip_horizontal:N \__enumext_labelsep_viii_dim
4252 \tl_use:N \__enumext_after_list_args_viii_tl
4253 \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
4254 \skip_set_eq:NN \parindent \__enumext_listparindent_viii_dim
4255 \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
4256 \bool_if:NT \__enumext_item_starred_viii_bool
4257     {
4258         \tl_use:N \__enumext_fake_item_indent_viii_tl
4259         \__enumext_keyans_show_item_opt:
4260         \skip_horizontal:n { -\__enumext_fake_item_indent_viii_dim - \__enumext_labelsep_viii_dim }
4261     }
4262     {
4263         \tl_use:N \__enumext_fake_item_indent_viii_tl
4264     }
4265 }

```

(End of definition for __enumext_start_item_viii:w.)

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

```

4266 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4267 {
4268     \__enumext_endminipage:
4269     \endlrbox
4270     \group_end:
4271     \box_set_wd:Nn \__enumext_item_text_viii_box
4272     {
4273         \__enumext_joined_width_viii_dim
4274         + \__enumext_labelwidth_viii_dim
4275         + \__enumext_labelsep_viii_dim
4276     }
4277     \int_set:Nn \hbadness { 10000 }
4278     \box_use_drop:N \__enumext_item_text_viii_box
4279     \bool_if:NF \__enumext_footnotes_key_bool
4280     {
4281         \__enumext_print_footnote:
4282     }
4283     \int_compare:nNnTF
4284     { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
4285     {
4286         \par\noindent
4287         \int_zero:N \__enumext_item_column_pos_viii_int
4288     }
4289     { \hspace{ \__enumext_columns_sep_viii_dim } }
4290 }

```

(End of definition for __enumext_stop_item_viii:.)

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

```

4291 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4292 {
4293     \int_compare:nNnT
4294     {
4295         \int_mod:nn
4296         { \__enumext_item_count_all_viii_int }
4297         { \__enumext_columns_viii_int }

```

```

4298     }
4299     =
4300     { 0 }
4301     {
4302         \par
4303         \vspace{ -\l__enumext_itemsep_viii_skip }
4304         \int_gzero:N \g__enumext_item_count_all_viii_int
4305     }
4306 }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`.)

11.41 The command `\getkeyans`

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

```

4307 \NewDocumentCommand \getkeyans { m }
4308 {
4309     \exp_args:Ne \__enumext_getkeyans_aux:n
4310     { \tl_to_str:e { \text_expand:n {#1} } }
4311 }

```

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

`__enumext_getkeyans_aux:n`

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

```

4312 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4313 {
4314     \str_if_in:nnTF {#1} { : }
4315     {
4316         \use:e
4317         {
4318             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4319             { {##1} {##2} }
4320         }
4321         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4322     }
4323     { \msg_error:nnn { enumext } { missing-colon } {#1} }
4324 }

```

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

`__enumext_getkeyans:nn`

The internal function `__enumext_getkeyans:nn` will check for the existence of the `⟨prop list⟩`, if it does not exist it will return an error message, then it will fetch the content specified by the second `⟨argument⟩` from `⟨prop list⟩`.

```

4325 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4326 {
4327     \prop_if_exist:cF { g__enumext_#1_prop }
4328     { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4329     \group_begin:
4330     \prop_item:cn { g__enumext_#1_prop }{#2}
4331     \group_end:
4332 }

```

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

11.42 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⟩]`

```

4333 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4334 \keys_define:nn { enumext / print }
4335 {
4336     print* .code:n = \keys_precompile:neN { enumext / enumext* }

```

```

4337         { \__enumext_filter_save_key:n {#1} }
4338         \__enumext_print_keyans_starred_tl, % starred cmd
4339     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4340     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4341         { \__enumext_filter_save_key:n {#1} }
4342         \__enumext_print_keyans_i_tl,
4343     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4344     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4345         { \__enumext_filter_save_key:n {#1} }
4346         \__enumext_print_keyans_ii_tl,
4347     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4348     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4349         { \__enumext_filter_save_key:n {#1} }
4350         \__enumext_print_keyans_iii_tl,
4351     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4352     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4353         { \__enumext_filter_save_key:n {#1} }
4354         \__enumext_print_keyans_iv_tl,
4355     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4356     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4357         { \__enumext_filter_save_key:n {#1} }
4358         \__enumext_print_keyans_vii_tl, % starred nested
4359     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4360 }

```

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

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

```

4361 \NewDocumentCommand \printkeyans { s O{} m }
4362 {
4363     \group_begin:
4364         \tl_use:N \__enumext_print_keyans_i_tl
4365         \tl_use:N \__enumext_print_keyans_ii_tl
4366         \tl_use:N \__enumext_print_keyans_iii_tl
4367         \tl_use:N \__enumext_print_keyans_iv_tl
4368         \tl_use:N \__enumext_print_keyans_vii_tl
4369         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4370     \group_end:
4371 }

```

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

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

```

4372 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4373 {
4374     \seq_if_exist:cTF { g__enumext_#3_seq }
4375     {
4376         \seq_if_empty:cF { g__enumext_#3_seq }
4377         {
4378             %%\seq_show:c { g__enumext_#3_seq }

```

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

```

4379         \bool_if:nTF {#1}
4380         {
4381             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4382             {
4383                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4384             }
4385             {
4386                 \tl_use:N \__enumext_print_keyans_starred_tl
4387                 \begin{enumext*}[#2]
4388                     \keys_set:nn { enumext / level-1 } { base-fix }
4389                     \seq_map_inline:cn { g__enumext_#3_seq } { #1 }

```

```

4390         \end{enumext*}
4391     }
4392 }

Otherwise it will open the environment enumext passing the optional argument to the “first level”, set the
key base-fix and then map the  $\langle sequence \rangle$ .

4393 {
4394     \begin{enumext}[#2]
4395         \keys_set:nn { enumext / enumext* }{ base-fix }
4396         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4397     \end{enumext}
4398 }
4399 }
4400 }
4401 {
4402     \msg_error:nnn { enumext } { undefined-storage-anskey } { #3 }
4403 }
4404 }

```

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

11.43 The command `\setenumext`

First we define a “*meta families*” of $\langle keys \rangle$ to access from `\setenumext`.

```

4405 \keys_define:nn { enumext / meta-families }
4406 {
4407     enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } { #1 } } ,
4408     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } { #1 } } ,
4409     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } { #1 } } ,
4410     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } { #1 } } ,
4411     keyans .code:n = { \keys_set:nn { enumext / keyans } { #1 } } ,
4412     enumext* .code:n = { \keys_set:nn { enumext / enumext* } { #1 } } ,
4413     keyans* .code:n = { \keys_set:nn { enumext / keyans* } { #1 } } ,
4414     print* .code:n = { \keys_set:nn { enumext / print } { print* = { #1 } } } ,
4415     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = { #1 } } } ,
4416     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = { #1 } } } ,
4417     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = { #1 } } } ,
4418     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = { #1 } } } ,
4419     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = { #1 } } } ,
4420     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4421 }

```

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

```

4422 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4423 {
4424     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4425     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4426 }

```

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

```

4427 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4428 {
4429     \tl_if_novalue:nTF { #1 }
4430     {
4431         \seq_map_inline:Nn \c__enumext_all_families_seq
4432     }
4433     {
4434         \seq_clear:N \l__enumext_setkey_tmpa_seq
4435         \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq { #1 }
4436         \int_set:Nn \l__enumext_setkey_tmpa_int
4437         {
4438             \seq_count:N \l__enumext_setkey_tmpb_seq
4439         }
4440         \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4441         {
4442             \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4443             \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4444             \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4445             {
4446                 \tl_use:N \l__enumext_setkey_tmpa_tl - #1
4447             }
4448         }
4449     }

```

```

4449         {
4450             \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4451         }
4452         \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4453         { \seq_map_inline:Nn \c__enumext_all_families_seq }
4454         { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4455     }
4456     {
4457         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4458     }
4459 }

```

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

```

__enumext_set_parse:n
__enumext_set_error:nn

```

Internal functions used by the `\setenumext` command.

```

4460 \cs_new_protected:Npn \__enumext_set_parse:n #1
4461 {
4462     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4463     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } {%<- max level
4464         { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4465     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4466     {
4467         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4468         { \tl_trim_spaces:n {#1} }
4469     }
4470     { \__enumext_set_error:nn {#1} { } }
4471 }
4472 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4473 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

11.44 Messages

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

```

4474 \msg_new:nnn { enumext } { package-load }
4475 {
4476     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4477 }
4478 \msg_new:nnn { enumext } { package-not-load }
4479 {
4480     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4481 }
4482 \msg_new:nnn { enumext } { package-load-foot }
4483 {
4484     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4485 }

```

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

```

4486 \msg_new:nnn { enumext } { counters }
4487 {
4488     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4489     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4490 }

```

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

```

4491 \msg_new:nnn { enumext } { unknown-choice }
4492 {
4493     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4494 }

```

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

```

4495 \msg_new:nnnn { enumext } { anskey-env-error }
4496 {
4497     The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4498     'enumext' ~ package, ~ It~ is~ already~ defined.
4499 }
4500 {
4501     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4502     for ~ the ~ 'save-ans' ~ key.\\
4503 }

```

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

```

4504 \msg_new:nnn { enumext } { store-prop }
4505 {
4506   * ~ Package ~ enumext: ~ Creating ~
4507   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4508 }
4509 \msg_new:nnn { enumext } { store-seq }
4510 {
4511   * ~ Package ~ enumext: ~ Creating ~
4512   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4513 }
4514 \msg_new:nnn { enumext } { store-int }
4515 {
4516   * ~ Package ~ enumext: ~ Creating ~
4517   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4518 }
4519 \msg_new:nnn { enumext } { prop-seq-int-hook }
4520 {
4521   * ~ Package ~ enumext: ~ Elements ~ in ~
4522   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4523   * ~ Package ~ enumext: ~ Elements ~ in ~
4524   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4525   * ~ Package ~ enumext: ~ Value ~ off ~
4526   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4527 }
4528 \msg_new:nnn { enumext } { item-answer-hook }
4529 {
4530   * ~ Package ~ enumext: ~ Value ~ off ~
4531   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4532   * ~ Package ~ enumext: ~ Value ~ off ~
4533   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4534   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4535 }

```

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

```

4536 \msg_new:nnn { enumext } { invalid-key }
4537 {
4538   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4539 }
4540 \msg_new:nnn { enumext } { unknown-key-family }
4541 {
4542   Unknown~key~family~`\l_keys_key_str'~for~enumext.
4543 }

```

Messages used in length calculation.

```

4544 \msg_new:nnn { enumext } { width-negative }
4545 {
4546   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4547   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
4548 }
4549 \msg_new:nnn { enumext } { width-zero }
4550 {
4551   Invalid ~ '#1=#2' ~ \msg_line_context:.\
4552   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
4553 }

```

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

```

4554 \msg_new:nnn { enumext } { list-lengths }
4555 {
4556   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4557   \__enumext_show_length:nnn { dim } { labelsep } {#1}
4558   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4559   \__enumext_show_length:nnn { dim } { itemindent } {#1}
4560   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4561   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4562   \__enumext_show_length:nnn { dim } { listparindent } {#1}
4563   \__enumext_show_length:nnn { skip } { topsep } {#1}
4564   \__enumext_show_length:nnn { skip } { parsep } {#1}
4565   \__enumext_show_length:nnn { skip } { partopsep } {#1}
4566   \__enumext_show_length:nnn { skip } { itemsep } {#1}
4567   ****~
4568 }

```


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

```
4569 \msg_new:nnn { enumext } { list-lengths-not-nested }
4570 {
4571     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4572     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4573     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4574     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4575     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4576     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4577     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4578     \__enumext_show_length:nnn { skip } { topsep } {#1}
4579     \__enumext_show_length:nnn { skip } { parsep } {#1}
4580     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4581     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4582     *****
4583 }
```

Messages used by `ref` key.

```
4584 \msg_new:nnn { enumext } { key-ref-empty }
4585 {
4586     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4587 }
```

Messages used by `save-ans` key.

```
4588 \msg_new:nnn { enumext } { save-ans-empty }
4589 {
4590     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4591 }
4592 \msg_new:nnn { enumext } { save-ans-log }
4593 {
4594     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4595 }
4596 \msg_new:nnn { enumext } { save-ans-log-hook }
4597 {
4598     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4599 }
4600 \msg_new:nnn { enumext } { save-ans-hook }
4601 {
4602     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4603 }
```

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

```
4604 \msg_new:nnn { enumext } { need-save-ans }
4605 {
4606     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4607 }
4608 \msg_new:nnn { enumext } { items-same-answer }
4609 {
4610     *****\\
4611     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4612     for ~ \c_left_brace_str #2 \c_right_brace_str\\
4613     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4614     'OK', ~ all ~ items ~ with ~ answer.\\
4615     *****
4616 }
4617 \msg_new:nnn { enumext } { item-greater-answer }
4618 {
4619     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4620     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4621     Items ~ > ~ Answers.
4622 }
4623 \msg_new:nnn { enumext } { item-less-answer }
4624 {
4625     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4626     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4627     Items ~ < ~ Answers.
4628 }
```

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

```
4629 \msg_new:nnn { enumext } { missing-starred }
4630 {
4631     Missing ~ '\c_backslash_str #1*' ~ #2.
4632 }
```

```

4633 \msg_new:nnn { enumext } { many-starred }
4634 {
4635   Many ~ '\c_backslash_str #1*' ~ #2.
4636 }

```

Messages used by `\printkeyans*` command.

```

4637 \msg_new:nnn { enumext } { print-starred }
4638 {
4639   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4640   #2 ~ environment ~ \msg_line_context:.
4641 }

```

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

```

4642 \msg_new:nnn { enumext } { list-too-deep }
4643 {
4644   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4645   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4646 }

```

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

```

4647 \msg_new:nnn { enumext } { anskey-unnumber-item }
4648 {
4649   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4650 }
4651 \msg_new:nnn { enumext } { anskey-already-stored }
4652 {
4653   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4654 }
4655 \msg_new:nnn { enumext } { anskey-empty-arg }
4656 {
4657   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4658 }
4659 \msg_new:nnn { enumext } { anskey-wrong-place }
4660 {
4661   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4662   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4663 }
4664 \msg_new:nnn { enumext } { anskey-nested }
4665 {
4666   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4667 }
4668 \msg_new:nnn { enumext } { anskey-math-mode }
4669 {
4670   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4671 }
4672 \msg_new:nnn { enumext } { anskey-env-wrong }
4673 {
4674   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4675 }
4676 \msg_new:nnn { enumext } { ansPIC-wrong-place }
4677 {
4678   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4679   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4680 }
4681 \msg_new:nnn { enumext } { command-wrong-place }
4682 {
4683   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4684   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4685 }

```

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

```

4686 \msg_new:nnn { enumext } { keyans-nested }
4687 {
4688   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
4689 }
4690 \msg_new:nnn { enumext } { keyans-wrong-level }
4691 {
4692   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
4693   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4694 }
4695 \msg_new:nnn { enumext } { wrong-place }
4696 {
4697   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\

```

```

4698     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
4699   }
4700   \msg_new:nnn { enumext } { keyanspic-nested }
4701   {
4702     The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
4703   }
4704   \msg_new:nnn { enumext } { keyanspic-wrong-level }
4705   {
4706     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
4707     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4708   }

```

Messages used by `\getkeyans` command.

```

4709   \msg_new:nnn { enumext } { undefined-storage-anskey }
4710   {
4711     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4712   }

```

Messages used by `\miniright` command.

```

4713   \msg_new:nnn { enumext } { missing-miniright }
4714   {
4715     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
4716     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4717   }
4718   \msg_new:nnn { enumext } { wrong-miniright-place }
4719   {
4720     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4721     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4722   }
4723   \msg_new:nnn { enumext } { wrong-miniright-use }
4724   {
4725     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
4726     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4727   }

```

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

```

4728   \msg_new:nnn { enumext } { nested }
4729   {
4730     The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4731   }
4732   \msg_new:nnn { enumext } { item-joined }
4733   {
4734     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
4735   }
4736   \msg_new:nnn { enumext } { item-joined-columns }
4737   {
4738     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4739   }

```

11.45 Finish package

Finish package implementation.

```

4740   \file_input_stop:
4741   </package>

```

12 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

*

.....

211

\+

.....

203

\-

.....

203

\\

219, 2599, 3482, 4488, 4497, 4502, 4522, 4524, 4531, 4533, 4546, 4551, 4556, 4571, 4610, 4612, 4614, 4619, 4620, 4625, 4626, 4644, 4661, 4678, 4683, 4692, 4697, 4706, 4715, 4720, 4725

A

above

.....

1424

above*

.....

1424

\addvspace

1078, 1106, 1222, 1301, 1364, 1370, 1398, 1415, 3321, 3336, 3438, 3453, 3747, 3761, 3802, 3816

after

.....

917

align

.....

492

\Alph

.....

36, 40, 41

\Alpha

.....

444, 559, 604, 672, 4355

\alph

.....

36, 40, 41

\alpha

.....

445, 557, 4347

\anskey

.....

12, 72, 2260

anskey*

.....

13, 2542

\anspic

.....

15, 98, 3460

\anspic*

.....

67

\arabic

.....

30, 36

\arabic

.....

443, 556, 603, 4339, 4343, 4359

B

base-fix

.....

796

\baselineskip

.....

49

\baselineskip

.....

813, 824

before

.....

917

before*

.....

917

below

.....

1424

below*

.....

1424

bool commands:

\bool_gset_false:N

319, 320, 321, 2695, 2697, 3763, 3767, 3818

\bool_gset_true:N

231, 240, 1020, 1922, 1928, 3739, 3764, 3794, 3819

\bool_if:NTF

384, 396, 413, 1446, 1460, 1473, 1484, 1495, 1506, 1517, 1528, 1582, 1599, 1604, 1612, 1639, 1677, 1682, 1689, 1693, 1715, 1720, 1728, 1735, 1766, 1774, 1867, 2013, 2082, 2092, 2171, 2195, 2202, 2226, 2264, 2278, 2306, 2318, 2340, 2457, 2468, 2472, 2533, 2544, 2644, 2724, 2739, 2814, 2825, 2829, 2942, 2973, 3048, 3064, 3126, 3136, 3166, 3171, 3255, 3305, 3319, 3327, 3366, 3423, 3436, 3444, 3462, 3735, 3744, 3748, 3790, 3799, 3803, 3887, 3897, 3980, 3985, 3994, 3998, 4013, 4042, 4195, 4199, 4223, 4232, 4236, 4242, 4256, 4279

\bool_if:nTF

1399, 1416, 2985, 3020, 3084, 3483, 4379

\bool_if_p:N

249, 264, 809, 810, 820, 821, 1746, 1747, 1755, 1756, 1880, 1906, 1919, 1920, 1925, 1926, 2325, 2366, 2367, 2391, 2400, 2401, 2413, 2429, 2630, 2801, 2802, 2839, 2840, 3228, 3230, 3241, 3490, 3491

\bool_lazy_all:nTF

247, 262, 1878, 1904, 2389, 2398, 2411, 2427, 3226, 3239

\bool_lazy_and:nnTF

227, 236, 808, 819, 1745, 1754, 1918, 1924, 2324, 2331, 2365, 2497, 2509, 2629, 2635, 2800

\bool_lazy_or:nnTF

.....

1807, 1814, 2838, 3489

\bool_new:N

34, 35, 36, 37, 38, 39, 40, 41, 63, 72, 93, 98, 99, 104, 105, 108, 129, 130, 138, 139, 144, 147, 148, 162, 173, 175

\bool_not_p:n

228, 237, 2326, 2332, 2416, 2431, 2631, 2636, 3229, 3242

\bool_set_eq:NN

.....

2951, 3000, 3930, 4149

\bool_set_false:N

393, 830, 1852, 1853, 1885, 1890, 1894, 1898, 1911, 2582, 3342, 3374, 3456, 3521, 3539, 3882, 3927, 4099, 4146

\bool_set_true:N

254, 255, 269, 270, 375, 379, 485, 845, 1430, 1435, 1702, 1824, 1825, 2114, 2122, 2583, 2945, 2947, 2976, 2978, 2996, 3008, 3203, 3235, 3248, 3274, 3371, 3398, 3724, 3779, 3855, 3936, 3943, 3944, 3988, 4155, 4162, 4163

box commands:

\box_dp:N

1118, 1122, 1126, 1137, 1141, 1152, 1161, 1167, 1177, 1190, 1196, 1202, 1233, 1234, 1235, 1238, 1248, 1252, 1261, 1268, 1273, 1281, 1310, 1311, 1314, 1321, 1334, 1342, 1348, 1356, 3551

\box_new:N

.....

69, 168, 174

\box_set_wd:Nn

.....

4034, 4271

\box_use_drop:N

.....

3759, 3814, 4041, 4278

\box_wd:N

.....

451

C

\c

.....

211, 212, 709, 711, 723, 725

\catcode

.....

2599

\cB

.....

212

\cE

.....

212

\centering

.....

1401, 1418, 3572, 3752, 3807

check-ans

.....

1844

Document class:

article

.....

42

clist commands:

\clist_const:Nn

.....

180

\clist_map_function:nN

.....

3559

\clist_map_inline:Nn

491, 751, 850, 916, 931, 1012, 1440

\clist_map_inline:nn

48, 59, 77, 83, 95, 107, 132, 156, 179, 519, 539, 805, 855, 1026, 1546, 1791, 1858, 2061, 2079, 2111, 2386, 2733, 2900, 3113, 3116, 3143, 3153, 3156, 3176, 4463

\columnbreak

.....

74

\columnbreak

.....

2328

columns

.....

996

columns-sep

.....

996

\columnsep

.....

94, 97

\columnsep

.....

3299, 3420

\columnseprule

.....

94, 97

\columnseprule

.....

3303, 3422

Commands provide by enumext:

\anskey

28, 63, 64, 69, 71-73, 75-77, 82, 83, 92, 106, 115, 116, 121

\anspic*

.....

28, 67, 71, 81-83, 98-100, 115, 116

\anspic

.....

71, 98-100, 121

\getkeyans

.....

71, 115, 122

©2024 by Pablo González L

123 / 136

`\item*` 28, 67, 71, 81–83, 86, 87, 108, 112, 113, 115, 116
`\itemwidth` 101
`\item` 86, 87, 101, 107, 108, 111, 112
`\miniright` 27, 47, 54, 55, 93, 94, 96, 97, 122
`\printkeyans*` 115
`\printkeyans` 28, 71, 115, 116
`\setenumext` 28, 116–119

Counters defined by `enumext`:

`enumXiii` 26, 35
`enumXii` 26, 35
`enumXiv` 26, 35
`enumXi` 26, 35
`enumXviii` 26, 35
`enumXvii` 26, 35, 108
`enumXvi` 26, 35
`enumXv` 26, 35

cs commands:

`\cs_generate_variant:Nn` 453, 469, 715, 731, 2163, 2168, 2244, 2550, 3103, 3561, 4333
`\cs_if_exist:NTF` 423
`\cs_if_free:NTF` 2501, 2513
`\cs_new:Nn` 197
`\cs_new:Npn` . 215, 1547, 1556, 1566, 2126, 2135, 2143
`\cs_new_eq:NN` 346, 347, 348, 352, 353, 398, 399, 402, 403
`\cs_new_protected:Nn` . 207, 221, 245, 278, 305, 311, 317, 323, 329, 337, 355, 370, 580, 643, 695, 806, 932, 936, 940, 944, 948, 952, 956, 960, 964, 968, 972, 976, 980, 984, 988, 992, 1027, 1039, 1063, 1080, 1091, 1108, 1183, 1207, 1224, 1286, 1303, 1325, 1360, 1366, 1441, 1455, 1469, 1480, 1491, 1502, 1513, 1524, 1610, 1713, 1726, 1743, 1764, 1792, 1797, 1822, 1863, 1873, 1916, 1931, 1938, 1947, 1952, 1957, 1962, 1971, 1976, 1981, 2003, 2169, 2193, 2200, 2224, 2231, 2276, 2377, 2488, 2551, 2572, 2603, 2627, 2669, 2693, 2722, 2737, 2765, 2798, 2834, 2846, 2854, 2905, 2909, 2928, 2981, 3016, 3032, 3042, 3058, 3196, 3224, 3253, 3260, 3283, 3313, 3325, 3364, 3388, 3406, 3431, 3442, 3479, 3523, 3537, 3557, 3562, 3578, 3597, 3714, 3733, 3769, 3788, 3846, 3869, 3876, 3885, 3895, 3912, 4053, 4090, 4112, 4118, 4131, 4187, 4291
`\cs_new_protected:Npn` 185, 189, 193, 406, 421, 438, 448, 454, 560, 605, 677, 702, 716, 1388, 1407, 1578, 1597, 1667, 1700, 1802, 1986, 2080, 2090, 2112, 2120, 2155, 2164, 2295, 2315, 2454, 2466, 2542, 2593, 2701, 2775, 2819, 2938, 2957, 2992, 3004, 3072, 3106, 3146, 3206, 3384, 3532, 3616, 3665, 3858, 3918, 3925, 3941, 3949, 3954, 3966, 4105, 4137, 4144, 4160, 4168, 4182, 4312, 4325, 4372, 4460, 4472
`\cs_new_protected_nopar:Nn` ... 3905, 4029, 4124, 4266
`\cs_new_protected_nopar:Npn` 3972, 4215
`\cs_set:Nn` 2459
`\cs_set:Npn` 2387, 2425, 4318
`\cs_set_eq:NN` . . 3836, 3837, 3974, 4080, 4081, 4217
`\cs_set_protected:Nn` 856, 872, 884, 896
`\cs_set_protected:Npn` . 44, 53, 70, 78, 90, 96, 125, 152, 160, 470, 492, 524, 540, 587, 732, 752, 796, 832, 851, 908, 917, 996, 1013, 1424, 1535, 1783, 1844, 2023, 2062, 2098, 2379, 2726, 2889, 3104, 3144
`\cs_to_str:N` 440, 463
`\cs_undefine:N` 2490, 2491, 2492, 2493

D

`\d` 203
`\DeclareDocumentEnvironment` 359

dim commands:

`\dim_abs:n` 3077, 3082
`\dim_add:Nn` 3542
`\dim_compare:nNnTF` . 858, 874, 886, 898, 1390, 1409, 3074, 3079, 3085, 3091, 3093, 3095, 3265, 3288, 3392, 3410, 3534, 3580, 3599, 3716, 3771
`\dim_compare:nTF` 2350, 2657
`\dim_gset_eq:NN` 3725, 3780
`\dim_gzero:N` 2699, 3766, 3821
`\dim_new:N` 66, 73, 74, 75, 92, 134, 169, 170, 176
`\dim_set:Nn` . . 451, 846, 2971, 3077, 3082, 3084, 3087, 3088, 3092, 3094, 3097, 3098, 3100, 3268, 3291, 3394, 3412, 3564, 3582, 3589, 3601, 3608, 3651, 3700, 3718, 3773, 3968
`\dim_set_eq:NN` 547, 594, 665, 669, 2966, 3115, 3155, 3299, 3420, 3658, 3661, 3662, 3707, 3710, 3711, 3959
`\dim_use:N` 859, 867, 1391, 1397, 2234, 2237, 2242, 3037, 3039, 3266, 3271, 3272, 3279, 3289, 3293, 3294, 3296
`\dim_zero:N` 3303, 3422, 3543, 3544, 3545
`\dim_zero_new:N` 3595, 3614
`\c_zero_dim` 861, 875, 887, 899, 1391, 1409, 2352, 2659, 3074, 3079, 3085, 3092, 3266, 3289, 3392, 3410, 3580, 3599, 3716, 3771

E

`\end` . . 1394, 1412, 2197, 2228, 3318, 3335, 3435, 3452, 3737, 3760, 3792, 3815, 4381, 4390, 4397
`\endgroup` 2599
`\endlist` 347
`\endlrbox` 4032, 4269
`\endminipage` 353
`enumext` 5, 3177
 enumext internal commands:
`\l__enumext_u_check_start_line_env_tl` ... 32
`\l__enumext_u_ref_the_count_tl` 38
`\l__enumext__resume_name_tl` 59
`__enumext_add_pre_parsep:` . 48, 1037, 1039, 1039
`__enumext_after_args_exec:` . 45, 932, 944, 3189
`__enumext_after_args_exec_v:` 46, 948, 960, 3357
`__enumext_after_args_exec_vii:` ... 964, 988
`__enumext_after_args_exec_viii:` 992
`__enumext_after_env:nn` . 68, 79, 80, 95, 103, 110, 189, 189, 2613, 3345, 3742, 3797, 4067
`__enumext_after_hyperref:` ... 34, 368, 370, 370
`__enumext_after_list:` . 94, 106, 111, 3194, 3325, 3325
`\l__enumext_after_list_args_v_tl` 962
`\l__enumext_after_list_args_vii_tl` 990, 4023
`\l__enumext_after_list_args_viii_tl` 994, 4252
`__enumext_after_list_v:` . . 97, 3362, 3442, 3442
`__enumext_after_list_vii:` ... 3844, 3876, 3876
`__enumext_after_list_viii:` . . 4088, 4118, 4118
`__enumext_after_stop_list:` ... 45, 46, 932, 940, 3340
`__enumext_after_stop_list_v:` 46, 948, 956, 3457
`\l__enumext_after_stop_list_v_tl` 958
`__enumext_after_stop_list_vii:` 964, 980, 3879
`\l__enumext_after_stop_list_vii_tl` ... 982
`__enumext_after_stop_list_viii:` . 984, 4121
`\l__enumext_after_stop_list_viii_tl` ... 986
`\l__enumext_align_label_vii_str` . . 4015, 4019
`\l__enumext_align_label_viii_str` . 4244, 4248
`\l__enumext_align_label_X_str` 160
`\c__enumext_all_envs_clist` . . 180, 491, 751, 850, 916, 931, 1012, 1440

```

\c__enumext_all_families_seq .. 117, 4422, 4431,
    4453
\__enumext_anskey__env_keys: ..... 80
\l__enumext_anskey_env_bool 31, 78, 34, 255, 270,
    2544
\__enumext_anskey_env_clean: .. 81, 2623, 2627,
    2693
\__enumext_anskey_env_define_keys: 78, 2542,
    2551, 2607
\__enumext_anskey_env_exec: 79, 2547, 2603, 2603
\__enumext_anskey_env_keys: .. 2621, 2627, 2627
\__enumext_anskey_env_make:n 63, 78, 1827, 2542,
    2542, 2550
\__enumext_anskey_env_store: .. 80, 2622, 2627,
    2669
\__enumext_anskey_env_undefine_keys: . 79, 80,
    2572, 2624
\__enumext_anskey_env_undefine_keys:\__-
    enumext_rescan_anskey_env:n ..... 2542
\l__enumext_anskey_level_int .. 28, 2301, 2302
\__enumext_anskey_safe_inner:n 73, 2266, 2276,
    2295
\__enumext_anskey_safe_outer: . 73, 2262, 2276,
    2276
\__enumext_anskey_show_wrap_arg:n . 76, 2454,
    2454, 2470, 2485
\__enumext_anskey_show_wrap_left:n 77, 2322,
    2466, 2466
\__enumext_anskey_wrapper:n ..... 2027, 2464
\__enumext_at_begin_document:n .. 33, 185, 185,
    344, 350
\l__enumext_base_line_fix_bool . 800, 810, 821,
    830
\__enumext_before_args_exec: 45, 932, 932, 3263
\__enumext_before_args_exec_v: 45, 46, 948, 948,
    3391
\__enumext_before_args_exec_vii: .. 964, 964,
    3873
\__enumext_before_args_exec_viii: 968, 4115
\__enumext_before_env:nn 78, 189, 193, 2495, 2507,
    2519, 2605
\__enumext_before_keys_exec: 45, 932, 936, 3187
\__enumext_before_keys_exec_v: .. 45, 948, 952,
    3355
\__enumext_before_keys_exec_vii ..... 964
\__enumext_before_keys_exec_vii: 46, 972, 3833
\__enumext_before_keys_exec_viii: .. 46, 976,
    4077
\__enumext_before_list: ... 93, 3181, 3260, 3260
\__enumext_before_list_v: . 96, 3350, 3388, 3388
\__enumext_before_list_vii: ... 106, 3828, 3869,
    3869
\__enumext_before_list_viii: .. 111, 4073, 4112,
    4112
\l__enumext_before_no_starred_key_v_tl 954
\l__enumext_before_no_starred_key_vii_-
    tl ..... 974
\l__enumext_before_no_starred_key_viii_-
    tl ..... 978
\l__enumext_before_starred_key_v_tl ... 950
\l__enumext_before_starred_key_vii_tl . 966
\l__enumext_before_starred_key_viii_tl 970
\__enumext_calc_hspace:NNNNNNN 89, 3072, 3072,
    3103, 3108, 3148
\__enumext_check_ans_active: 65, 93, 1863, 1863,
    3264, 3872
\g__enumext_check_ans_item_tl ..... 83
\g__enumext_check_ans_key_bool .. 66, 138, 319,
    1922, 1928, 2013
\l__enumext_check_ans_key_bool .. 66, 86, 1848,
    1853, 1919, 1925
\__enumext_check_ans_key_hook: 66, 1916, 1916,
    3339, 3880
\__enumext_check_ans_level: 65, 1863, 1869, 1873
\__enumext_check_ans_log: 66, 67, 1962, 1962, 2017
\__enumext_check_ans_log_msg_greater: 1962,
    1968, 1981
\__enumext_check_ans_log_msg_less: 1962, 1966,
    1971
\__enumext_check_ans_log_msg_same_ok: 1962,
    1967, 1976
\__enumext_check_ans_msg_greater: 1938, 1944,
    1957
\__enumext_check_ans_msg_less: 1938, 1942, 1947
\__enumext_check_ans_msg_same_ok: 1938, 1943,
    1952
\__enumext_check_ans_show: .. 66, 67, 1938, 1938,
    2015
\g__enumext_check_ans_show_bool ..... 95
\l__enumext_check_answers_bool 63, 65, 73, 138,
    1825, 1852, 1867, 2171, 2195, 2202, 2226, 2264, 2814,
    2942, 2973, 3985
\__enumext_check_starred_cmd:n 32, 67, 83, 1986,
    1986, 3360, 3518, 4086
\g__enumext_check_starred_cmd_int 138, 1989,
    1995, 2000, 3014, 3488, 4194
\l__enumext_check_start_line_env_tl 138, 284,
    291, 298, 1992, 1998, 2001
\l__enumext_columns_sep_v_dim 3410, 3412, 3420
\l__enumext_columns_sep_vii_dim .. 3580, 3582,
    3591, 3655, 4051
\l__enumext_columns_sep_viii_dim . 3599, 3601,
    3610, 3704, 4289
\l__enumext_columns_v_int 1229, 3408, 3416, 3428,
    3433
\l__enumext_columns_vii_int .. 3585, 3588, 3592,
    3619, 3623, 3626, 3632, 3638, 3642, 4046, 4057
\l__enumext_columns_viii_int . 3604, 3607, 3611,
    3668, 3672, 3675, 3681, 3687, 3691, 4284, 4297
\l__enumext_counter_i_tl ..... 44, 430
\l__enumext_counter_ii_tl ..... 44, 431
\l__enumext_counter_iii_tl ..... 44, 432
\l__enumext_counter_iv_tl ..... 44, 433
\c__enumext_counter_style_tl ..... 30, 49, 209
\g__enumext_counter_styles_tl . 27, 36, 66, 441,
    459
\l__enumext_counter_v_tl ..... 44, 434, 685
\l__enumext_counter_vi_tl ..... 44, 435
\l__enumext_counter_vii_tl ..... 44, 436, 615
\l__enumext_counter_viii_tl ..... 44, 437, 632
\l__enumext_current_widest_dim 27, 66, 465, 548,
    595, 666, 670
\__enumext_default_item:n ... 2938, 2938, 2989
\__enumext_define_counters:Nn 26, 421, 421, 430,
    431, 432, 433, 434, 435, 436, 437
\__enumext_endminipage: . 33, 350, 353, 365, 3574,
    4031, 4268
\g__enumext_envir_name_tl 31, 34, 256, 271, 327,
    1795, 1800, 1810, 1950, 1955, 1960, 1974, 1979, 1984

```


__enumext_execute_after_env: 32, 33, 63, 66, 67, 77, 2003, 2003, 3345, 4067
 __enumext_fake_item: 856, 856, 3135
 \l__enumext_fake_item_indent_v_dim 875, 880
 \l__enumext_fake_item_indent_v_tl 877, 2997, 3001, 3009
 \l__enumext_fake_item_indent_vii_dim 887, 892
 \l__enumext_fake_item_indent_vii_tl 889, 4027
 \l__enumext_fake_item_indent_viii_dim . 899, 904, 4260
 \l__enumext_fake_item_indent_viii_tl . 901, 4258, 4263
 \l__enumext_fake_item_indent_X_tl 96
 __enumext_fake_item_vii: 856, 884, 3165
 __enumext_fake_item_viii: 856, 896, 3170
 __enumext_filter_save_key:n . 70, 2087, 2095, 2118, 2124, 2126, 2126, 4337, 4341, 4345, 4349, 4353, 4357
 __enumext_filter_save_key_key:n . 70, 2126, 2131, 2135
 __enumext_filter_save_key_pair:nn 70, 2126, 2132, 2143
 __enumext_filter_series:n 58, 1547, 1547, 1590, 1602, 1607
 __enumext_filter_series_key:n 58, 1547, 1552, 1556
 __enumext_filter_series_pair:nn . 58, 1547, 1553, 1566
 \g__enumext_footnote_arg_seq . 157, 2911, 2924, 2934
 \g__enumext_footnote_int . 157, 2918, 2921, 2923, 2925
 \g__enumext_footnote_int_seq . 157, 2912, 2925, 2930, 2933
 __enumext_footnotes_key_bool 34
 \l__enumext_footnotes_key_bool 29, 34, 108, 147, 379, 384, 393, 3994, 4042, 4232, 4279
 __enumext_footnotetext:nn . 2905, 2905, 2935
 __enumext_getkeyans:nn . 115, 4321, 4325, 4325
 __enumext_getkeyans_aux:n 115, 4309, 4312, 4312
 \l__enumext_hyperref_bool 29, 34, 147, 375, 396, 413, 2367, 2802, 3980, 4223
 __enumext_hypertarget:nn 34, 370, 398, 402, 418
 __enumext_if_is_int:n 201
 __enumext_if_is_int:nTF 201, 704, 718
 __enumext_internal_mini_page: . 33, 355, 355, 3198, 3848
 __enumext_is_not_nested: . 26, 31, 92, 221, 221, 3199, 3849
 __enumext_is_on_first_level: . 26, 31, 92, 221, 245, 3204, 3856
 \g__enumext_item_anskey_int 73, 83, 138, 314, 341, 342, 1935, 2271, 2816
 __enumext_item_answer_diff: 66, 67, 1931, 1931, 2010
 \g__enumext_item_answer_diff_int 66, 138, 315, 1933, 1940, 1964
 \l__enumext_item_answer_diff_int 138
 \l__enumext_item_column_pos_vii_int 107, 3626, 3632, 3638, 3642, 3649, 3908, 4046, 4049
 \l__enumext_item_column_pos_viii_int . 111, 3675, 3681, 3687, 3691, 3698, 4127, 4284, 4287
 \l__enumext_item_column_pos_X_int 160
 \g__enumext_item_count_all_vii_int 107, 3650, 3909, 4057, 4064
 \g__enumext_item_count_all_viii_int 111, 3699, 4128, 4296, 4304
 \g__enumext_item_count_all_X_int 160
 \g__enumext_item_number_bool 138
 \l__enumext_item_number_bool 65, 144, 1885, 1890, 1894, 1898, 1911, 2306, 2533, 2945, 2976, 3988
 \g__enumext_item_number_int . 65, 138, 313, 340, 342, 1884, 1889, 1893, 1897, 1910, 1935, 2944, 2975, 3987
 __enumext_item_peek_args_vii: 107, 3910, 3912, 3912
 __enumext_item_peek_args_viii: . 111, 4129, 4131, 4131
 __enumext_item_starred: . 88, 3032, 3032, 3050
 \l__enumext_item_starred_vii_bool 3927, 3943, 3998
 \l__enumext_item_starred_viii_bool 4146, 4162, 4236, 4256
 \l__enumext_item_starred_X_bool 160
 __enumext_item_std:w . 33, 86, 87, 100, 344, 348, 2948, 2954, 2979, 2997, 3001, 3009, 3555
 \g__enumext_item_symbol_aux_vii_tl 3951, 4000, 4003, 4007, 4009
 \g__enumext_item_symbol_aux_X_tl 160
 \l__enumext_item_symbol_sep_vii_dim . 3960, 3968, 4006, 4008
 \g__enumext_item_symbol_tl . 86, 60, 122, 2963, 3038, 3055
 \l__enumext_item_symbol_vii_tl 4003
 \l__enumext_item_text_vii_box 3993, 4034, 4041
 \l__enumext_item_text_viii_box 4231, 4271, 4278
 \l__enumext_item_text_X_box 160
 \l__enumext_item_width_vii_dim . 3589, 3653, 3661, 3662
 \l__enumext_item_width_viii_dim . 3608, 3702, 3710, 3711
 \l__enumext_item_width_X_dim 160
 \l__enumext_itemindent_X_dim 70
 \l__enumext_itemsep_vii_skip 4063
 \l__enumext_itemsep_viii_skip 4303
 \l__enumext_joined_item_aux_vii_int . 3647, 3648, 3649, 3650, 3656
 \l__enumext_joined_item_aux_viii_int . 3696, 3697, 3698, 3699, 3705
 \l__enumext_joined_item_aux_X_int . 160
 __enumext_joined_item_vii:w . 107, 3915, 3916, 3918, 3918
 \l__enumext_joined_item_vii_int . 3618, 3619, 3622, 3624, 3630, 3635, 3640, 3645, 3647, 3653
 __enumext_joined_item_viii:w . 111, 112, 4134, 4135, 4137, 4137
 \l__enumext_joined_item_viii_int . 3667, 3668, 3671, 3673, 3679, 3684, 3689, 3694, 3696, 3702
 \l__enumext_joined_item_X_int 160
 \l__enumext_joined_width_vii_dim . 3651, 3658, 3661, 4024, 4036
 \l__enumext_joined_width_viii_dim 3700, 3707, 3710, 4253, 4273
 \l__enumext_joined_width_X_dim 160
 __enumext_keyans_addto_prop:n 81, 2701, 2701, 3011, 3485
 __enumext_keyans_addto_seq:n . 83, 2775, 2775, 3013, 3487

__enumext_keyans_addto_seq_link: [2775](#), [2796](#),
 [2798](#), [4193](#)
 __enumext_keyans_anspic_code:nnn . [98](#), [3476](#),
 [3479](#), [3479](#)
 __enumext_keyans_default_item:n . . [87](#), [2992](#),
 [2992](#), [3028](#)
 \l__enumext_keyans_env_bool [34](#), [3229](#), [3242](#), [3371](#),
 [3456](#)
 __enumext_keyans_fake_item: . . [856](#), [872](#), [3125](#)
 \l__enumext_keyans_level_h_int . . [28](#), [625](#), [652](#),
 [2286](#), [2525](#), [2753](#), [4092](#), [4093](#)
 \l__enumext_keyans_level_int . . [28](#), [1382](#), [2282](#),
 [2521](#), [2748](#), [3370](#), [3375](#), [3470](#)
 __enumext_keyans_make_label: [36](#), [89](#), [3058](#), [3058](#),
 [3123](#)
 __enumext_keyans_mini_addvspace: [52](#), [96](#), [1286](#),
 [1286](#), [3400](#)
 __enumext_keyans_mini_right_cmd:n [55](#), [1384](#),
 [1407](#), [1407](#)
 __enumext_keyans_mini_set_vskip: . [52](#), [1224](#),
 [1224](#), [1288](#)
 __enumext_keyans_multi_addvspace: [97](#), [1080](#),
 [1091](#), [3425](#)
 __enumext_keyans_multi_set_vskip: [49](#), [1080](#),
 [1080](#), [1093](#)
 __enumext_keyans_multicols_start: [96](#), [3404](#),
 [3406](#), [3406](#)
 __enumext_keyans_multicols_stop: . [97](#), [1411](#),
 [3431](#), [3431](#), [3455](#)
 __enumext_keyans_parse_keys:n [3349](#), [3384](#), [3384](#)
 \l__enumext_keyans_pic_above_int . [133](#), [3565](#),
 [3566](#), [3568](#)
 \l__enumext_keyans_pic_above_skip . [100](#), [133](#),
 [3509](#), [3549](#)
 __enumext_keyans_pic_arg_two: [100](#), [3507](#), [3537](#),
 [3537](#)
 \l__enumext_keyans_pic_below_int . [133](#), [3565](#),
 [3566](#), [3569](#)
 \l__enumext_keyans_pic_body_seq . [98–100](#), [133](#),
 [3474](#), [3514](#), [3573](#)
 __enumext_keyans_pic_do:n [100](#), [3514](#), [3516](#), [3557](#),
 [3557](#), [3561](#)
 \l__enumext_keyans_pic_level_int . . [28](#), [1374](#),
 [2290](#), [2529](#), [2704](#), [2743](#), [2778](#), [2856](#), [3525](#), [3526](#)
 __enumext_keyans_pic_row:n . . . [100](#), [3559](#), [3562](#),
 [3562](#)
 __enumext_keyans_pic_safe_exec: . . [99](#), [3503](#),
 [3523](#), [3523](#)
 __enumext_keyans_pic_skip_abs:N . [100](#), [3532](#),
 [3532](#), [3548](#)
 \l__enumext_keyans_pic_width_dim . [133](#), [3564](#),
 [3571](#)
 __enumext_keyans_redefine_item: . . [88](#), [3016](#),
 [3016](#), [3122](#)
 __enumext_keyans_ref: [40](#), [677](#), [695](#), [3124](#)
 __enumext_keyans_ref:n [40](#), [674](#), [677](#), [677](#)
 __enumext_keyans_safe_exec: . [3348](#), [3364](#), [3364](#)
 __enumext_keyans_show_ans: . . [2819](#), [2827](#), [2846](#)
 __enumext_keyans_show_item_opt: . [2819](#), [2834](#),
 [3009](#), [3499](#), [4259](#)
 __enumext_keyans_show_left:n . [87](#), [2819](#), [2819](#),
 [3007](#), [3494](#)
 __enumext_keyans_show_pos: . . [2819](#), [2831](#), [2854](#)
 __enumext_keyans_starred_item:n . . [87](#), [3004](#),
 [3004](#), [3024](#)
 __enumext_keyans_start_line: . [26](#), [32](#), [278](#), [278](#),
 [3372](#), [3530](#), [4097](#)
 __enumext_keyans_store_ref: . . [82](#), [2722](#), [2722](#),
 [3012](#), [3486](#), [4191](#)
 __enumext_keyans_store_ref_aux_i: [82](#), [2722](#),
 [2734](#), [2737](#)
 __enumext_keyans_store_ref_aux_ii: [82](#), [2722](#),
 [2763](#), [2765](#)
 __enumext_keyans_wrapper_opt:n . . [2030](#), [2842](#)
 \l__enumext_label_copy_i_tl . . [2421](#), [2741](#), [2746](#),
 [2751](#), [2756](#)
 \l__enumext_label_copy_v_tl [2751](#)
 \l__enumext_label_copy_vi_tl [2746](#)
 \l__enumext_label_copy_vii_tl [2396](#), [2407](#), [2438](#),
 [2741](#)
 \l__enumext_label_copy_viii_tl [2756](#)
 \l__enumext_label_copy_X_tl [149](#)
 \l__enumext_label_fill_left_v_tl [3062](#)
 \l__enumext_label_fill_left_X_tl [96](#)
 \l__enumext_label_fill_right_v_tl [3069](#)
 \l__enumext_label_fill_right_X_tl [96](#)
 \l__enumext_label_font_style_v_tl [3063](#), [3498](#)
 \l__enumext_label_font_style_vii_tl . . [4012](#)
 \l__enumext_label_font_style_viii_tl . . [4241](#)
 \l__enumext_label_i_tl [540](#)
 \l__enumext_label_ii_tl [540](#)
 \l__enumext_label_iii_tl [540](#)
 \l__enumext_label_iv_tl [540](#)
 __enumext_label_style:Nnn [26](#), [36](#), [454](#), [454](#), [469](#),
 [545](#), [592](#), [663](#), [667](#)
 \l__enumext_label_v_tl . . [81](#), [83](#), [660](#), [2709](#), [2783](#),
 [2848](#), [2883](#), [3006](#), [3010](#), [3352](#), [3493](#), [3495](#)
 \l__enumext_label_vi_tl . [81](#), [83](#), [660](#), [2706](#), [2780](#),
 [3493](#), [3495](#), [3499](#)
 \l__enumext_label_vii_tl . [587](#), [3938](#), [3963](#), [3970](#)
 \l__enumext_label_viii_tl [587](#), [4157](#), [4185](#), [4189](#)
 \l__enumext_label_width_by_box . . [66](#), [450](#), [451](#)
 __enumext_label_width_by_box:Nn [36](#), [448](#), [448](#),
 [453](#), [465](#), [728](#)
 \l__enumext_labelsep_i_dim . . . [2851](#), [2886](#), [4197](#),
 [4212](#)
 \l__enumext_labelsep_v_dim [3415](#)
 \l__enumext_labelsep_vii_dim . [3584](#), [3593](#), [3654](#),
 [3961](#), [4022](#), [4038](#)
 \l__enumext_labelsep_viii_dim [3603](#), [3612](#), [3703](#),
 [4251](#), [4260](#), [4275](#)
 \l__enumext_labelwidth_i_dim . [2851](#), [2886](#), [4197](#),
 [4212](#)
 \l__enumext_labelwidth_v_dim [3415](#)
 \l__enumext_labelwidth_vii_dim . . . [3584](#), [3592](#),
 [3654](#), [4015](#), [4019](#), [4037](#)
 \l__enumext_labelwidth_viii_dim . . [3603](#), [3611](#),
 [3703](#), [4244](#), [4248](#), [4274](#)
 \l__enumext_leftmargin_tmp_v_bool . [100](#), [3539](#)
 \l__enumext_leftmargin_tmp_X_bool [70](#)
 \l__enumext_leftmargin_tmp_X_dim [70](#)
 \l__enumext_leftmargin_X_dim [70](#)
 __enumext_level: [197](#), [197](#), [569](#), [572](#), [573](#), [582](#), [584](#),
 [859](#), [863](#), [867](#), [934](#), [938](#), [942](#), [946](#), [1029](#), [1031](#), [1033](#),
 [1035](#), [1068](#), [1070](#), [1072](#), [1074](#), [1078](#), [1111](#), [1114](#), [1133](#),
 [1142](#), [1148](#), [1153](#), [1157](#), [1168](#), [1172](#), [1173](#), [1178](#), [1214](#),
 [1218](#), [1391](#), [1397](#), [1444](#), [1446](#), [1448](#), [1451](#), [1458](#), [1460](#),
 [1462](#), [1465](#), [2082](#), [2084](#), [2086](#), [2114](#), [2115](#), [2117](#), [2173](#),

2181, 2185, 2189, 2459, 2462, 2463, 2947, 2948, 2952,
 2953, 2954, 2961, 2963, 2967, 2968, 2971, 2978, 2979,
 3034, 3037, 3039, 3046, 3047, 3048, 3051, 3054, 3184,
 3186, 3235, 3248, 3255, 3266, 3268, 3271, 3272, 3274,
 3279, 3286, 3289, 3291, 3293, 3294, 3295, 3296, 3299,
 3305, 3310, 3316, 3319, 3321, 3327
 \l__enumext_level_h_int 28, 229, 251, 265, 608, 645,
 1881, 1901, 2415, 2432, 2499, 2511, 3243, 3850, 3851
 \l__enumext_level_int . 92, 28, 199, 238, 250, 266,
 357, 1041, 1185, 1378, 1875, 1907, 2005, 2392, 2402,
 2408, 2414, 2422, 2430, 2437, 2498, 2510, 3138, 3200,
 3201, 3211, 3219, 3233, 3246, 3301, 3379, 3466, 3889,
 3899, 4100
 __enumext_list_arg_two_i: 3104
 __enumext_list_arg_two_ii: 3104
 __enumext_list_arg_two_iii: 3104
 __enumext_list_arg_two_iv: 3104
 __enumext_list_arg_two_v: . 88, 3104, 3354, 3540
 __enumext_list_arg_two_vii: 3144, 3832
 __enumext_list_arg_two_viii: 3144, 4076
 \l__enumext_listoffset_v_dim 3417
 \l__enumext_listparindent_vii_dim 4025
 \l__enumext_listparindent_viii_dim . . . 4254
 __enumext_log_answer_vars: . 33, 329, 337, 2012
 __enumext_log_global_vars: . 32, 329, 329, 2011
 __enumext_make_label: 36, 86, 88, 3042, 3042, 3133
 \l__enumext_mark_answer_sym_tl 72, 2036, 2239,
 2474, 2858, 2871, 4201
 \l__enumext_mark_position_str 122, 2040, 2041,
 2067, 2068, 2237
 \l__enumext_mark_ref_sym_tl . . 2053, 2372, 2810
 __enumext_mini_addvspace: . . 51, 93, 1207, 1207,
 3276
 __enumext_mini_addvspace_vii: 54, 1360, 1360,
 3728
 __enumext_mini_addvspace_viii: 54, 1360, 1366,
 3783
 __enumext_mini_env* 355
 __enumext_mini_right_cmd:n . 54, 55, 1386, 1388,
 1388
 __enumext_mini_set_vskip: . 49, 1108, 1108, 1209
 __enumext_mini_set_vskip_vii: 53, 1303, 1303,
 1362
 __enumext_mini_set_vskip_viii: 53, 1303, 1325,
 1368
 __enumext_minipage:w 33, 350, 352, 361, 3571, 4024,
 4253
 \l__enumext_minipage_active_v_bool . . 96, 97,
 3398, 3423, 3436, 3444
 \g__enumext_minipage_active_vii_bool . . 103,
 3739, 3744, 3763
 \l__enumext_minipage_active_vii_bool . 3724,
 3735
 \g__enumext_minipage_active_viii_bool 3794,
 3799, 3818
 \l__enumext_minipage_active_viii_bool 3779,
 3790
 \g__enumext_minipage_active_X_bool . . . 160
 \l__enumext_minipage_active_X_bool 84
 \g__enumext_minipage_after_skip 84, 1307, 1319,
 3761, 3816
 \l__enumext_minipage_after_skip 50, 51, 94, 97,
 84, 1124, 1139, 1159, 1175, 1190, 1196, 1202, 1216,
 1226, 1235, 1238, 1250, 1268, 1279, 1295, 1327, 1340,
 1354, 3336, 3453
 \g__enumext_minipage_center_vii_bool . 3748,
 3764
 \g__enumext_minipage_center_viii_bool 3803,
 3819
 \g__enumext_minipage_center_X_bool . . . 160
 \l__enumext_minipage_hsep_v_dim . . . 96, 3396
 \l__enumext_minipage_hsep_vii_dim 3722
 \l__enumext_minipage_hsep_viii_dim . . . 3777
 \l__enumext_minipage_left_skip 50, 96, 84, 1116,
 1131, 1150, 1165, 1212, 1222, 1227, 1233, 1242, 1259,
 1271, 1291, 1301, 1305, 1310, 1314, 1328, 1332, 1346,
 1364, 1370
 \l__enumext_minipage_left_v_dim 96, 3394, 3402
 \l__enumext_minipage_left_vii_dim 3718, 3730
 \l__enumext_minipage_left_viii_dim 3773, 3785
 \l__enumext_minipage_left_X_dim 84
 \g__enumext_minipage_right_skip 84, 1306, 1311,
 1315, 3747, 3802
 \l__enumext_minipage_right_skip . 50, 84, 1120,
 1135, 1155, 1170, 1228, 1234, 1246, 1264, 1275, 1329,
 1336, 1350, 1398, 1415
 \l__enumext_minipage_right_v_dim . . 96, 1409,
 1414, 3392, 3396
 \g__enumext_minipage_right_vii_dim 103, 3726,
 3746, 3766
 \l__enumext_minipage_right_vii_dim 103, 3716,
 3721, 3727
 \g__enumext_minipage_right_viii_dim . . 3781,
 3801, 3821
 \l__enumext_minipage_right_viii_dim . . 3771,
 3776, 3782
 \g__enumext_minipage_right_X_dim 160
 \g__enumext_minipage_right_X_skip 160
 \g__enumext_minipage_stat_int . 93, 96, 84, 1403,
 1420, 3275, 3329, 3334, 3399, 3446, 3451
 \l__enumext_miniright_code_vii_box 3755, 3759
 \g__enumext_miniright_code_vii_tl 103, 3750,
 3757, 3765
 \l__enumext_miniright_code_viii_box . . 3810,
 3814
 \g__enumext_miniright_code_viii_tl 3805, 3812,
 3820
 \l__enumext_miniright_code_X_box 160
 __enumext_multi_addvspace: . 48, 94, 1063, 1063,
 3307
 __enumext_multi_set_vskip: 47, 1027, 1027, 1065
 \l__enumext_multicols_above_ii_skip . . . 1046
 \l__enumext_multicols_above_iii_skip . . 1052
 \l__enumext_multicols_above_iv_skip . . . 1058
 \l__enumext_multicols_above_v_skip 1082, 1096,
 1106
 \l__enumext_multicols_above_X_skip 78
 \l__enumext_multicols_below_v_skip 1086, 1100,
 3438
 \l__enumext_multicols_below_X_skip 78
 __enumext_multicols_start: . 93, 94, 3281, 3283,
 3283
 __enumext_multicols_stop: 94, 1393, 3313, 3313,
 3338
 __enumext_nested_base_line_fix: 42, 796, 806,
 3215, 3866
 __enumext_newlabel:nn 29, 34, 76, 406, 406, 2448,
 2769
 \l__enumext_newlabel_arg_one_tl 29, 34, 76, 82,

[149](#), [2371](#), [2441](#), [2449](#), [2758](#), [2770](#), [2808](#)
`\l__enumext_newlabel_arg_two_tl` [29](#), [35](#), [75](#), [149](#),
[2395](#), [2405](#), [2419](#), [2435](#), [2450](#), [2745](#), [2750](#), [2755](#), [2771](#)
`__enumext_parse_keys:n` [42](#), [58](#), [3180](#), [3206](#), [3206](#)
`__enumext_parse_keys_vii:n` [42](#), [58](#), [3826](#), [3858](#),
[3858](#)
`__enumext_parse_keys_viii:n` [4071](#), [4105](#), [4105](#)
`__enumext_parse_save_key:n` [70](#), [2107](#), [2112](#), [2112](#)
`__enumext_parse_save_key_vii:n` [70](#), [2102](#), [2112](#),
[2120](#)
`__enumext_parse_serie:n` [105](#)
`__enumext_parse_series:n` [58](#), [92](#), [1578](#), [1578](#),
[3214](#), [3864](#)
`__enumext_parse_store_keys:n` [92](#)
`\l__enumext_parsep_i_skip` [1044](#), [1046](#), [1188](#), [1236](#)
`\l__enumext_parsep_ii_skip` [1050](#), [1052](#), [1194](#)
`\l__enumext_parsep_iii_skip` [1056](#), [1058](#), [1200](#)
`\l__enumext_parsep_vii_skip` [4026](#)
`\l__enumext_parsep_viii_skip` [4255](#)
`\l__enumext_partopsep_v_skip` [1098](#), [1102](#), [1262](#),
[1266](#), [1273](#), [1277](#), [1293](#), [1297](#)
`\l__enumext_partopsep_viii_skip` [1338](#)
`__enumext_phantomsection:` [34](#), [370](#), [399](#), [403](#), [419](#)
`__enumext_print_footnote:` [2905](#), [2928](#), [4044](#),
[4281](#)
`__enumext_print_keyans_box:NN` [72](#), [2231](#), [2231](#),
[2244](#), [2461](#), [2850](#), [2885](#), [4197](#), [4212](#)
`\l__enumext_print_keyans_i_tl` [4342](#), [4364](#)
`\l__enumext_print_keyans_ii_tl` [4346](#), [4365](#)
`\l__enumext_print_keyans_iii_tl` [4350](#), [4366](#)
`\l__enumext_print_keyans_iv_tl` [4354](#), [4367](#)
`\l__enumext_print_keyans_starred_tl` [115](#), [116](#),
[122](#), [4338](#), [4386](#)
`\l__enumext_print_keyans_vii_tl` [115](#), [4358](#), [4368](#)
`\l__enumext_print_keyans_X_tl` [122](#)
`__enumext_printkeyans:nnn` [116](#), [4369](#), [4372](#), [4372](#)
`__enumext_redefine_item:` [87](#), [2981](#), [2981](#), [3132](#)
`\l__enumext_ref_key_arg_tl` [38](#), [49](#), [212](#), [562](#), [563](#),
[576](#), [607](#), [610](#), [621](#), [627](#), [638](#), [679](#), [680](#), [691](#)
`\l__enumext_ref_the_count_tl` [38](#), [49](#), [569](#), [572](#),
[575](#), [615](#), [617](#), [620](#), [632](#), [634](#), [637](#), [685](#), [687](#), [690](#)
`__enumext_regex_counter_style:` [30](#), [38](#), [207](#),
[207](#), [570](#), [616](#), [633](#), [686](#)
`__enumext_register_counter_style:Nn` [438](#),
[438](#), [443](#), [444](#), [445](#), [446](#), [447](#)
`__enumext_remove_extra_parsep_vii:` [3841](#),
[4053](#), [4053](#)
`__enumext_remove_extra_parsep_viii:` [4085](#),
[4291](#), [4291](#)
`__enumext_renew_footnote:` [2905](#), [2909](#), [3996](#),
[4234](#)
`\l__enumext_renew_the_count_v_tl` [688](#), [697](#), [699](#)
`\l__enumext_renew_the_count_vii_tl` [618](#), [647](#),
[649](#)
`\l__enumext_renew_the_count_viii_tl` [635](#), [654](#),
[656](#)
`\l__enumext_renew_the_count_X_tl` [49](#)
`__enumext_rescan_anskey_env:n` [79](#), [80](#), [2593](#),
[2679](#), [2687](#)
`__enumext_reset_global_bool:` [305](#), [308](#), [317](#)
`__enumext_reset_global_int:` [305](#), [307](#), [311](#)
`__enumext_reset_global_tl:` [305](#), [309](#), [323](#)
`__enumext_reset_global_vars:` [32](#), [67](#), [305](#), [305](#),
[2020](#)
`\l__enumext_resume_active_bool` [58](#), [61](#), [60](#), [1582](#),
[1702](#)
`__enumext_resume_counter:` [60](#), [61](#), [1700](#), [1706](#),
[1713](#)
`__enumext_resume_counter:n` [58](#), [61](#), [1671](#), [1676](#),
[1700](#), [1700](#), [1770](#), [1778](#)
`__enumext_resume_counter_save_ans:` [61](#), [62](#),
[1700](#), [1711](#), [1743](#)
`__enumext_resume_counter_series:` [61](#), [1700](#),
[1709](#), [1726](#)
`\g__enumext_resume_int` [60](#), [1623](#), [1717](#), [1718](#)
`__enumext_resume_last:n` [58](#), [59](#), [1578](#), [1584](#), [1597](#)
`\l__enumext_resume_name_tl` [60](#), [1619](#), [1627](#), [1630](#),
[1646](#), [1654](#), [1657](#), [1703](#), [1704](#), [1732](#), [1739](#)
`__enumext_resume_save_counter:` [59](#), [1610](#), [1610](#),
[3343](#), [3883](#)
`__enumext_resume_series:n` [60](#), [1541](#), [1667](#), [1667](#)
`__enumext_resume_starred:` [62](#), [1542](#), [1764](#), [1764](#)
`\g__enumext_resume_vii_int` [106](#), [60](#), [1650](#), [1722](#),
[1723](#)
`__enumext_safe_exec:` [33](#), [92](#), [3179](#), [3196](#), [3196](#)
`__enumext_safe_exec_vii:` [33](#), [3825](#), [3846](#), [3846](#)
`__enumext_safe_exec_viii:` [4070](#), [4090](#), [4090](#)
`\l__enumext_series_name_tl` [61](#)
`\l__enumext_series_str` [59](#), [92](#), [1539](#), [1580](#), [1588](#),
[1589](#), [1591](#), [1593](#), [1614](#), [1617](#), [1621](#), [1641](#), [1644](#), [1648](#),
[3210](#), [3862](#)
`__enumext_set_error:nn` [4460](#), [4470](#), [4472](#)
`__enumext_set_parse:n` [4443](#), [4460](#), [4460](#)
`\l__enumext_setkey_tmpa_int` [117](#), [4436](#), [4440](#)
`\l__enumext_setkey_tmpa_seq` [117](#), [4434](#), [4444](#),
[4450](#), [4452](#), [4454](#), [4467](#)
`\l__enumext_setkey_tmpa_tl` [117](#), [4442](#), [4446](#)
`\l__enumext_setkey_tmpb_seq` [117](#), [4435](#), [4438](#),
[4442](#), [4443](#)
`\l__enumext_setkey_tmpb_tl` [117](#), [4462](#), [4464](#), [4465](#)
`\l__enumext_show_answer_bool` [2047](#), [2071](#), [2468](#),
[2825](#), [2839](#), [3490](#), [4195](#)
`__enumext_show_length:nnn` [44](#), [215](#), [215](#), [4557](#),
[4558](#), [4559](#), [4560](#), [4561](#), [4562](#), [4563](#), [4564](#), [4565](#), [4566](#),
[4572](#), [4573](#), [4574](#), [4575](#), [4576](#), [4577](#), [4578](#), [4579](#), [4580](#),
[4581](#)
`\l__enumext_show_position_bool` [2050](#), [2074](#),
[2472](#), [2829](#), [2840](#), [3491](#), [4199](#)
`\g__enumext_standar_bool` [31](#), [92](#), [34](#), [228](#), [231](#), [249](#),
[320](#), [1612](#), [1677](#), [1689](#), [1715](#), [1728](#), [1766](#), [1906](#), [1920](#),
[3230](#)
`\l__enumext_standar_bool` [92](#), [95](#), [34](#), [2400](#), [2413](#),
[2429](#), [3203](#), [3342](#)
`\l__enumext_standar_first_bool` [31](#), [92](#), [34](#), [254](#),
[809](#), [1599](#), [1746](#), [1808](#), [1815](#)
`__enumext_standar_item_vii:w` [107](#), [3923](#), [3925](#),
[3925](#)
`__enumext_standar_item_viii:w` [112](#), [4142](#), [4144](#),
[4144](#)
`__enumext_standar_ref:` [38](#), [560](#), [580](#), [3134](#)
`__enumext_standar_ref:n` [38](#), [552](#), [560](#), [560](#)
`\g__enumext_standar_series_tl` [60](#), [1601](#), [1602](#),
[1768](#), [1771](#)
`\g__enumext_starred_bool` [31](#), [105](#), [106](#), [34](#), [237](#), [240](#),
[264](#), [321](#), [1639](#), [1682](#), [1693](#), [1720](#), [1735](#), [1774](#), [1880](#),
[1926](#), [2391](#), [2401](#), [2431](#), [2739](#), [3767](#)
`\l__enumext_starred_bool` [105](#), [106](#), [34](#), [2326](#), [2332](#),
[2416](#), [2457](#), [2631](#), [2636](#), [3855](#), [3882](#)

```

__enumext_starred_columns_set_vii: .. 3578,
    3578, 3827
__enumext_starred_columns_set_viii: . 3578,
    3597, 4072
\l__enumext_starred_first_bool 31, 34, 269, 820,
    1604, 1755, 1808, 1815
__enumext_starred_item:nn ... 2957, 2957, 2987
__enumext_starred_item_exec: . 113, 4187, 4187,
    4238
__enumext_starred_item_vii:w . 107, 108, 3922,
    3941, 3941
__enumext_starred_item_vii_aux_i:w .. 3941,
    3946, 3949
__enumext_starred_item_vii_aux_ii:w . 3941,
    3947, 3952, 3954
__enumext_starred_item_vii_aux_iii:w 3941,
    3957, 3966
__enumext_starred_item_viii:w 112, 4141, 4160,
    4160
__enumext_starred_item_viii_aux_i:w .. 112,
    4160, 4165, 4168
__enumext_starred_item_viii_aux_ii:w . 112,
    4160, 4166, 4180, 4182
__enumext_starred_joined_item_vii:n 101, 107,
    3616, 3616, 3920
__enumext_starred_joined_item_viii:n . 101,
    112, 3616, 3665, 4139
__enumext_starred_ref: .... 39, 605, 643, 3162
__enumext_starred_ref:n .... 39, 599, 605, 605
\g__enumext_starred_series_tl . 60, 1606, 1607,
    1776, 1779
__enumext_start_from:NNn 40, 702, 702, 715, 737
\l__enumext_start_i_int ..... 1718, 1730, 1749
__enumext_start_item_tmp_vii: 105, 3837, 3905,
    3905
__enumext_start_item_tmp_viii: .. 110, 4081,
    4124, 4124
__enumext_start_item_vii:w 107, 108, 3933, 3938,
    3963, 3970, 3972, 3972
__enumext_start_item_viii:w .. 112, 4152, 4157,
    4185, 4215, 4215
\g__enumext_start_line_tl 31, 34, 257, 272, 326,
    1950, 1955, 1960, 1974, 1979, 1984
__enumext_start_list:nn .. 33, 89, 100, 344, 346,
    3183, 3351, 3504, 3830, 4074
__enumext_start_mini_vii: 106, 3714, 3714, 3874
__enumext_start_mini_viii: ... 111, 3769, 3769,
    4116
__enumext_start_save_ans_msg: 63, 1792, 1792,
    1817
__enumext_start_store_level: . 92, 3182, 3224,
    3224
__enumext_start_store_level_vii: 106, 3829,
    3885, 3885
\l__enumext_start_vii_int ... 1723, 1737, 1758
\l__enumext_start_X_int ..... 96, 732
__enumext_stop_item_tmp_vii: .. 105, 107, 108,
    3836, 3840, 3907, 3974
__enumext_stop_item_tmp_viii: 110, 111, 4080,
    4084, 4126, 4217
__enumext_stop_item_vii: 108, 109, 3974, 4029,
    4029
__enumext_stop_item_viii: 114, 4217, 4266, 4266
__enumext_stop_list: .. 33, 344, 347, 3192, 3361,
    3517, 3842, 4087
__enumext_stop_mini_vii: 103, 106, 3714, 3733,
    3878
__enumext_stop_mini_viii: 111, 3769, 3788, 4120
__enumext_stop_save_ans_msg: . 63, 1792, 1797,
    2009
__enumext_stop_store_level: .. 92, 3193, 3224,
    3253
__enumext_stop_store_level_vii: . 106, 3843,
    3885, 3895
\l__enumext_store_active_bool . 28, 63, 92, 105,
    108, 1747, 1756, 1824, 2278, 3228, 3241, 3366, 3374,
    3462, 3521, 3887, 3897, 4099
__enumext_store_active_keys:n 69, 2080, 2080,
    3221
__enumext_store_active_keys_vii:n . 69, 105,
    2080, 2090, 3865
__enumext_store_addto_prop:n 71, 81, 2155, 2155,
    2163, 2317, 2720, 4190
__enumext_store_addto_seq:n 71, 83, 2164, 2164,
    2168, 2175, 2189, 2197, 2206, 2220, 2228, 2375, 2813
\l__enumext_store_anskey_arg_tl .. 28, 74, 108,
    2323, 2328, 2330, 2335, 2342, 2345, 2355, 2360, 2363,
    2369, 2375
__enumext_store_anskey_code:n 73, 74, 80, 2272,
    2315, 2315, 2677, 2685
\l__enumext_store_anskey_env_tl 28, 79, 80, 108,
    2615, 2617, 2672, 2679, 2687
\l__enumext_store_anskey_opt_tl .. 28, 80, 108,
    2616, 2633, 2639, 2646, 2652, 2662, 2674, 2683
__enumext_store_anskey_safe_outer: .... 73
\g__enumext_store_columns_break_bool . 2555,
    2630, 2695
\l__enumext_store_columns_break_bool . 2247,
    2325
\l__enumext_store_current_label_tl 28, 81, 83,
    112, 108, 2703, 2706, 2709, 2716, 2718, 2720, 2777,
    2780, 2783, 2789, 2794, 2804, 2813, 4170, 4175, 4176,
    4189, 4190, 4192
\l__enumext_store_current_label_tmp_tl . 28,
    108, 3006, 3010
\l__enumext_store_current_opt_arg_tl 28, 112,
    108, 2823, 2836, 2842, 4178
__enumext_store_internal_ref: .. 74, 75, 2320,
    2377, 2377
\g__enumext_store_item_join_int .. 2558, 2637,
    2641, 2696
\l__enumext_store_item_join_int .. 2250, 2333,
    2337
\g__enumext_store_item_star_bool . 2560, 2644,
    2697
\l__enumext_store_item_star_bool . 2252, 2340
\g__enumext_store_item_symbol_sep_dim 2565,
    2659, 2664, 2699
\l__enumext_store_item_symbol_sep_dim 2257,
    2352, 2357
\g__enumext_store_item_symbol_tl . 2563, 2650,
    2654, 2698
\l__enumext_store_item_symbol_tl . 2255, 2343,
    2347
\l__enumext_store_keyans_item_opt_sep_
    tl .... 2033, 2714, 2716, 2787, 2791, 4173, 4175
__enumext_store_level_close: . 71, 2169, 2193,
    3257
__enumext_store_level_close_vii: . 72, 2200,

```


2224, 3901
 __enumext_store_level_open: .. 71, 2169, 2169, 3236, 3249
 __enumext_store_level_open_vii: .. 72, 2200, 2200, 3891
 \g__enumext_store_name_tl . 28, 63, 95, 108, 325, 332, 333, 334, 335, 1800, 1826, 1949, 1954, 1959, 1973, 1978, 1983, 2007
 \l__enumext_store_name_tl 28, 63, 65, 108, 1633, 1636, 1660, 1663, 1751, 1760, 1795, 1804, 1805, 1826, 1827, 1828, 1830, 1831, 1833, 1835, 1836, 1838, 1840, 1841, 1865, 2157, 2159, 2166, 2443, 2444, 2480, 2619, 2760, 2761, 2864, 2877, 4207
 \l__enumext_store_ref_key_bool 74, 2056, 2318, 2366, 2724, 2801
 \l__enumext_store_save_key_vii_bool .. 2092, 2122
 \l__enumext_store_save_key_vii_tl 2094, 2095, 2123, 2124, 2204, 2212, 2216, 2220
 \l__enumext_store_save_key_X_bool .. 69, 122
 \l__enumext_store_save_key_X_tl 69, 122
 \l__enumext_store_upper_level_X_bool .. 122
 __enumext_storing_exec: 63, 78, 1802, 1818, 1822
 __enumext_storing_set:n .. 63, 1787, 1802, 1802
 \l__enumext_the_counter_v_tl 687
 \l__enumext_the_counter_vii_tl 617
 \l__enumext_the_counter_viii_tl 634
 \l__enumext_the_counter_X_tl 49
 __enumext_tmp:n 44, 48, 53, 59, 70, 77, 78, 83, 90, 95, 96, 107, 125, 132, 152, 156, 160, 179, 796, 805, 851, 855, 1535, 1546, 1783, 1791, 1844, 1862, 2023, 2061, 2062, 2079, 2098, 2111, 2379, 2386, 2387, 2408, 2422, 2425, 2437, 2726, 2733, 3104, 3143, 3144, 3176
 __enumext_tmp:nn 470, 491, 492, 523, 524, 539, 732, 751, 832, 850, 908, 916, 917, 931, 996, 1012, 1013, 1026, 1424, 1440, 2889, 2904
 __enumext_tmp:nnn 540, 556, 557, 558, 559, 587, 603, 604
 __enumext_tmp:nnnnn 752, 777, 780, 783, 785, 787, 790, 793
 __enumext_tmp:w 4318, 4321
 \l__enumext_tmpa_vii_int 3588, 3591
 \l__enumext_tmpa_viii_int 3607, 3610
 \l__enumext_tmpa_X_int 160
 \l__enumext_topsep_v_skip 1084, 1088, 1231, 1244, 1252, 1257, 1277, 1281, 3520, 3552
 \l__enumext_topsep_vii_skip .. 1308, 1317, 1321
 \l__enumext_topsep_viii_skip . 1330, 1352, 1356
 __enumext_undefine_anskey_env: . 67, 77, 2018, 2488, 2488
 \l__enumext_vspace_a_star_v_bool 1473
 \l__enumext_vspace_a_star_vii_bool ... 1495
 \l__enumext_vspace_a_star_viii_bool ... 1506
 \l__enumext_vspace_a_star_X_bool 96
 __enumext_vspace_above: .. 56, 1441, 1441, 3262
 __enumext_vspace_above_v: . 56, 1469, 1469, 3390
 \l__enumext_vspace_above_v_skip .. 1471, 1475, 1477
 __enumext_vspace_above_vii: .. 57, 1491, 1491, 3871
 \l__enumext_vspace_above_vii_skip 1493, 1497, 1499
 __enumext_vspace_above_viii: . 57, 1491, 1502, 4114
 \l__enumext_vspace_above_viii_skip 1504, 1508, 1510
 \l__enumext_vspace_b_star_v_bool 1484
 \l__enumext_vspace_b_star_vii_bool ... 1517
 \l__enumext_vspace_b_star_viii_bool ... 1528
 \l__enumext_vspace_b_star_X_bool 96
 __enumext_vspace_below: .. 56, 1455, 1455, 3341
 __enumext_vspace_below_v: . 56, 1480, 1480, 3458
 \l__enumext_vspace_below_v_skip .. 1482, 1486, 1488
 __enumext_vspace_below_vii: .. 57, 1513, 1513, 3881
 \l__enumext_vspace_below_vii_skip 1515, 1519, 1521
 __enumext_vspace_below_viii: . 57, 1513, 1524, 4122
 \l__enumext_vspace_below_viii_skip 1526, 1530, 1532
 __enumext_widest_from:nnn .. 41, 716, 716, 731, 743
 \g__enumext_widest_label_tl 27, 36, 66, 458, 462, 466
 \l__enumext_wrap_label_opt_v_bool 3000
 \l__enumext_wrap_label_opt_vii_bool 107, 3932
 \l__enumext_wrap_label_opt_viii_bool .. 112, 4151
 \l__enumext_wrap_label_opt_X_bool 96
 \l__enumext_wrap_label_v_bool 2996, 3000, 3008, 3064
 \l__enumext_wrap_label_vii_bool .. 107, 3931, 3936, 3944, 4013
 \l__enumext_wrap_label_viii_bool . 112, 4150, 4155, 4163, 4242
 \l__enumext_wrap_label_X_bool 96
 __enumext_wrapper_label_v:n 3066, 3499
 __enumext_wrapper_label_vii:n 4016
 __enumext_wrapper_label_viii:n 4245
 \l__enumext_write_aux_file_tl . 29, 76, 82, 149, 2446, 2452, 2767, 2773
 __enumext_zero_parsep: ... 51, 1128, 1183, 1183
 enumext* 5, 3823
 enumXi 430
 enumXii 430
 enumXiii 430
 enumXiv 430
 enumXv 430
 enumXvi 430
 enumXvii 430
 enumXviii 430
 Environments provide by enumext:
 anskey* 28, 63, 67, 75–79, 92, 116, 118, 121
 enumext* 25, 26, 29–31, 33, 35, 38, 39, 41–44, 46, 47, 53, 54, 57–60, 62–65, 68–77, 80, 82, 85, 91–93, 104–106, 108, 110, 111, 113, 115, 116, 120, 122
 enumext 25, 26, 30, 31, 33, 35–45, 47–60, 62–65, 68–77, 80, 82, 85–93, 95, 99, 101, 103, 106, 115, 117, 119, 121
 keyans* 25, 26, 28–30, 32, 35, 38–41, 43, 44, 46, 47, 53, 54, 57, 63, 64, 67, 68, 71, 78, 82, 85, 91, 110, 111, 120, 122
 keyanspic 25, 26, 28, 32, 35, 36, 40, 54, 63, 64, 67, 71, 78, 81–83, 98–100, 121
 keyans .. 25, 26, 28, 30, 32, 35, 36, 40, 41, 43–47, 49, 52, 54–56, 63, 64, 67, 68, 71, 78, 81–83, 88–90, 95, 96, 98, 100, 103, 111, 120, 121
 Environments:
 list 30, 33, 89, 91

©2024 by Pablo González L

before*	44-46, 93, 106, 111
before	44-46, 96
below*	27, 55-57
below	27, 55-57, 95, 98, 106, 111
check-ans	28, 30, 31, 62-64, 66, 67, 70, 83, 86, 87, 93, 95, 110, 120
columns-sep	47, 94, 97
columns	27, 47, 49, 55, 94, 96
first	44-46, 108
font	36, 88, 108
item-pos*	85, 86
item-sym*	28, 85, 86
itemindent	27, 43, 88, 109
itemsep	41, 91
labelsep	36, 86, 90, 108
labelwidth	35-38, 40, 41, 90
label	26, 27, 35-37, 40, 41, 101
lisparindent	91
list-indent	27, 43, 100
list-offset	43
listparindent	43, 109
mark-ans	68, 70, 77
mark-pos	68, 118
mark-ref	68, 70, 75
mini-env	27, 33, 47, 54, 55, 70, 85, 93, 96, 103, 104, 106, 111
mini-right*	27, 30, 47, 70, 103, 104
mini-right	27, 30, 47, 54, 70, 103, 104
mini-sep	27, 47, 70, 93, 96
no-store	28, 62-65, 70, 73
noitemsep	41, 51
nosep	41, 51
parindent	91
parsep	41, 91, 109
partopsep	41
ref	26, 30, 37, 38, 40, 120
resume*	26, 57, 58, 62, 63, 70, 95
resume	26, 32, 57-63, 70, 95, 106
rightmargin	43
save-ans	28, 32, 58-63, 65-67, 69-71, 73, 77, 78, 81, 83, 87, 92, 95, 98, 106, 111, 113, 115, 120
save-key	28, 58, 69, 92
save-pos	70
save-ref	29, 34, 68, 70, 74, 75, 82, 83, 87, 113
save-sep	68, 70, 112
series	26, 57-62, 70, 92, 95
show-ans	68, 70, 72, 74, 77, 87, 113
show-length	30, 44, 119, 120
show-pos	28, 68, 72, 74, 77, 83, 87, 113
start	27, 30, 40, 41, 58
store-key	69
topsep	41
widest	27, 30, 41
wrap-ans	68, 70, 72, 76
wrap-label*	36, 86, 88, 107, 108, 112
wrap-label	36, 88, 107, 108, 112
wrap-opt	68, 70
keys commands:	
\keys_define:nn	472, 494, 526, 542, 589, 660, 734, 754, 798, 834, 853, 910, 919, 998, 1015, 1426, 1537, 1785, 1846, 2025, 2064, 2100, 2105, 2245, 2553, 2574, 2891, 4334, 4405
\l_keys_key_str	4542
\keys_precompile:nnN	116, 4333, 4336, 4340, 4344, 4348, 4352, 4356
\keys_set:nn	486, 814, 825, 1021, 1431, 1436, 1679, 1684, 1771, 1779, 2269, 2683, 3213, 3218, 3386, 3863, 4109, 4388, 4395, 4407, 4408, 4409, 4410, 4411, 4412, 4413, 4414, 4415, 4416, 4417, 4418, 4419, 4457
keyval commands:	
\keyval_parse:NNn	1551, 2130
L	
label	540, 587, 660
Labels provide by enumext:	
\Alph*	35, 36
\Roman*	35, 36
\alph*	35, 36
\arabic*	30, 35, 36
\roman*	35, 36
\labelsep	100
\labelsep	3542, 3545
labelsep	470
\labelwidth	36, 100
\labelwidth	3542, 3543
labelwidth	470
\leftmargin	90
\leftmargin	89, 3542
legacy commands:	
\legacy_if:nTF	3975, 3978, 4218, 4221
\legacy_if_gset_false:n	362
\legacy_if_set_false:n	3977, 4220
\legacy_if_set_true:n	3937, 3962, 3969, 3982, 4156, 4184, 4225
\linewidth	93, 96
\linewidth	3270, 3396, 3564, 3591, 3610, 3720, 3775
\list	346
list-indent	832
list-offset	832
\listparindent	3544
listparindent	832
\lrbox	3993, 4231
M	
\makebox	101
\makebox	2235, 2237, 3038, 4007, 4015, 4019, 4244, 4248
\makelabel	85, 88, 89, 101
\makelabel	88, 89, 3044, 3060
\makesavenoteenv	392
mark-ans	2023
mark-pos	2023, 2062
mark-ref	2023
mini-env	996
mini-sep	996
\minipage	352
\miniright	10, 54, 1372, 3332, 3449
mode commands:	
\mode_if_math:TF	2310, 2537
\mode_if_vertical:TF	1066, 1094, 1210, 1289
\mode_leave_vertical:	812, 823, 865, 879, 891, 903, 2233, 3036, 4005
msg commands:	
\msg_error:nn	2299, 2304, 2308, 2535, 3377, 3381, 3468, 3528, 3853, 4095, 4102, 4420
\msg_error:nnn	565, 612, 629, 682, 1376, 1380, 1405, 1422, 1691, 1695, 1810, 2312, 2503, 2515, 2523, 2527, 2531, 2539, 4323, 4328, 4402, 4473
\msg_error:nnnn	2280, 2284, 2288, 2292, 3368, 3464, 3472, 4383
\msg_error:nnnnn	513, 533, 2043

`\msg_fatal:nn` 3202
`\msg_fatal:nnn` 424
`\msg_info:nnn` 13, 16, 21, 24, 374, 388
`\msg_line_context:` .. 4507, 4512, 4517, 4546, 4551, 4556, 4571, 4586, 4590, 4594, 4598, 4602, 4606, 4613, 4620, 4626, 4640, 4644, 4649, 4653, 4657, 4661, 4666, 4670, 4674, 4678, 4683, 4688, 4692, 4697, 4702, 4706, 4711, 4715, 4720, 4725, 4730, 4734, 4738
`\msg_log:nnn` 1830, 1835, 1840
`\msg_log:nnnnn` 339, 1973, 1978, 1983
`\msg_log:nnnnnn` 331
`\msg_new:nnn` 4474, 4478, 4482, 4486, 4491, 4504, 4509, 4514, 4519, 4528, 4536, 4540, 4544, 4549, 4554, 4569, 4584, 4588, 4592, 4596, 4600, 4604, 4608, 4617, 4623, 4629, 4633, 4637, 4642, 4647, 4651, 4655, 4659, 4664, 4668, 4672, 4676, 4681, 4686, 4690, 4695, 4700, 4704, 4709, 4713, 4718, 4723, 4728, 4732, 4736
`\msg_new:nnnn` 4495
`\msg_term:nnnn` . 1794, 1799, 3128, 3138, 3167, 3172
`\msg_term:nnnnn` 1954
`\msg_warning:nn` 3331, 3448
`\msg_warning:nnnn` 1991, 1997, 3076, 3081, 3621, 3634, 3670, 3683
`\msg_warning:nnnnn` 1949, 1959
`\multicolsep` 94, 97
`\multicolsep` 3300, 3421

N

`\NeedsTeXFormat` 3
`\newcounter` 427
`\NewDocumentCommand` 1372, 2260, 3460, 4307, 4361, 4427
`\NewDocumentEnvironment` . 3177, 3346, 3501, 3823, 4068
`\newenvsc` 2546
`\newlabel` 35
`\newlabel` 410
`no-store` 1844
`\noindent` 105, 110
`\noindent` . 3277, 3401, 3729, 3784, 3836, 4048, 4080, 4286
`\nointerlineskip` 3277, 3401, 3729, 3784
`noitemsep` 752
`\nopagebreak` 1077, 1105, 1221, 1300, 1363, 1369
`\normalfont` 2477, 2861, 2874, 4204
`nosep` 752

P

Packages:
`caption` 103
`enumext` 25, 37, 62, 89, 98, 118, 119
`enumitem` 35
`expl3` 101
`footnotehyper` 34
`hyperref` 29, 30, 34, 35, 75, 83, 108, 118
`lua-visual-debug` 49
`multicol` 25, 118
`scontents` 25, 77, 78
`shortlst` 101
`\par` .. 1077, 1105, 1221, 1300, 1363, 1369, 1398, 1415, 2456, 3321, 3336, 3438, 3453, 3576, 3747, 3761, 3802, 3816, 4048, 4062, 4286, 4302
`\parindent` 4025, 4254
`\parsep` 48, 51, 99, 100
`\parsep` 3158, 3541, 3548, 3553
`parsep` 752
`\parskip` 4026, 4255
`\partopsep` 100
`\partopsep` 3159, 3546

`partopsep` 752
peek commands:
`\peek_meaning:NTF` 3914, 3928, 3945, 3956, 4133, 4147, 4164
`\peek_meaning_remove:NTF` 3921, 4140
`\peek_remove_spaces:n` 3022
`\phantomsection` 34
`\phantomsection` 399
prg commands:
`\prg_do_nothing:` 403
`\prg_new_protected_conditional:Npnn` ... 201
`\prg_replicate:nn` 218
`\prg_return_false:` 205
`\prg_return_true:` 204
`\printkeyans` 16, 115, 4361
prop commands:
`\prop_count:N` 333, 2159, 2444, 2480, 2761, 2864, 2877, 4207
`\prop_gput_if_not_in:Nnn` 2157
`\prop_if_exist:NTF` 1828, 4327
`\prop_item:Nn` 4330
`\prop_new:N` 1831
`\ProvidesExplPackage` 4

R

`\raggedcolumns` 3309, 3427
`\ref` 75, 82
`ref` 540, 587, 660
`\refstepcounter` 3984, 4227
regex commands:
`\regex_match:nnTF` 203, 709, 711, 723, 725
`\regex_replace_once:nnN` 211
`\renewcommand` 575, 620, 637, 690
`\RenewDocumentCommand` ... 2913, 2983, 3018, 3044, 3060
`\RequirePackage` 17, 25
`resume` 1535
`resume*` 1535
`rightmargin` 832
`\Roman` 36, 40, 41
`\Roman` 446
`\roman` 36, 40, 41
`\roman` 447, 558, 4351

S

`save-ans` 1783
`save-key` 2098
`save-ref` 2023
`save-sep` 2023
scan commands:
`\scan_stop:` 100, 3555, 3835, 4079, 4318, 4321
scontents internal commands:
`\l_scontents_fname_out_tl` 2584
`__scontents_parse_environment_keys:n` 2590
`__scontents_rescan_tokens:n` 2597
`\l_scontents_storing_bool` 2582
`\l_scontents_writing_bool` 2583
seq commands:
`\seq_clear:N` 4434
`\seq_const_from_clist:Nn` 4422
`\seq_count:N` 334, 3514, 4438
`\seq_gclear:N` 2911, 2912
`\seq_gput_right:Nn` 2166, 2924, 2925
`\seq_if_empty:NTF` 2930, 4376, 4452
`\seq_if_exist:NTF` 1833, 4374
`\seq_if_in:NnTF` 4381

©2024 by Pablo González L

\tl_trim_spaces:n	457, 4450, 4462, 4468	\use_none:nn	402
\tl_use:N	463, 466, 584, 649, 656, 699, 934, 938, 942, 946, 950, 954, 958, 962, 966, 970, 974, 978, 982, 986, 990, 994, 2239, 2388, 2396, 2407, 2421, 2426, 2438, 2948, 2954, 2979, 2997, 3001, 3009, 3046, 3047, 3054, 3062, 3063, 3069, 3184, 3352, 3498, 3757, 3812, 4012, 4023, 4027, 4241, 4252, 4258, 4263, 4364, 4365, 4366, 4367, 4368, 4386, 4446	\usecounter	3118, 3160
token commands:		V	
\token_to_str:N	410	\value	1617, 1623, 1630, 1636, 1644, 1650, 1657, 1663
topsep	752	vbox commands:	
\typeout	378, 381, 391, 392	\vbox_set_top:Nn	3755, 3810
U		\vspace	363, 813, 824, 1448, 1451, 1462, 1465, 1475, 1477, 1486, 1488, 1497, 1499, 1508, 1510, 1519, 1521, 1530, 1532, 3509, 3520, 4063, 4303
\u	212	W	
use commands:		widest	732
\use:N	219, 3051, 3186	wrap-ans	2023
\use:n	1549, 2128, 4316	wrap-label	470
		wrap-label*	470
		wrap-opt	2023